Vladimir Gorodetsky
Igor Kotenko
Victor A. Skormin (Eds.)

Communications in Computer and Information Science 1

Computer Network Security

Fourth International Conference
on Mathematical Methods, Models, and Architectures
for Computer Network Security, MMM-ACNS 2007
St. Petersburg, Russia, September 2007, Proceedings

Springer

Communications
in Computer and Information Science 1

Vladimir Gorodetsky Igor Kotenko
Victor A. Skormin (Eds.)

Computer Network Security

Fourth International Conference
on Mathematical Methods, Models, and Architectures
for Computer Network Security, MMM-ACNS 2007
St. Petersburg, Russia, September 13-15, 2007
Proceedings

Springer

Volume Editors

Vladimir Gorodetsky
Igor Kotenko
St. Petersburg Institute for Informatics and Automation
39, 14th Liniya, St. Petersburg, 199178, Russia
E-mail: {gor, ivkote}@mail.iias.spb.su

Victor A. Skormin
US Air Force, Binghamton University (SUNY)
Binghamton, NY 13902, USA
E-mail: vskormin@binghamton.edu

Library of Congress Control Number: 2007933896

CR Subject Classification (1998): C.2, D.4.6, K.4.4, H.2.8

ISSN 1865-0929
ISBN-10 3-540-73985-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-73985-2 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12101467 06/3180 5 4 3 2 1 0

Preface

This volume contains papers presented at the Fourth International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS 2007) held in St. Petersburg, Russia, during September 13–15, 2007. The workshop was organized by the St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS) in cooperation with Binghamton University (SUNY, USA).

The organizers are proud that the MMM-ACNS workshops hosted by the St. Petersburg Institute for Informatics and Automation in 2001, 2003 and 2005 evolved into a bi-annual series recognized in the professional community. These events not only demonstrated the keen interest of the participating researchers in the subject matter and the opportunity to present and disseminate individual achievements, but also promoted the spirit of cooperation, camaraderie, free exchange of ideas, and intellectually stimulating interaction between colleagues.

Again, MMM-ACNS 2007 provided an international forum for sharing original research results among specialists in fundamental and applied problems of computer network security. An important distinction of the conference was its focus on mathematical aspects of information and computer network security addressing the ever-increasing demands for secure computing and highly dependable computer networks.

A total of 56 papers from 18 countries related to significant aspects of both theory and applications of computer network and information security were submitted to MMM-ACNS 2007. In total, 18 papers were selected for regular presentations and 12 for short presentations (32 % of acceptance for full papers and 53 % for all papers).

The MMM-ACNS 2007 program was enriched by invited papers presented by six distinguished invited speakers: Christian Collberg (University of Arizona, USA), Angelos D. Keromytis (Columbia University, USA), Paulo Verissimo (University of Lisbon, Portugal), Jean-Daniel Aussel (Gemalto, France), Mauricio Sanchez (ProCurve Networking, HP, USA) and Victor Serdiouk (DialogueScience, Inc., Russia) addressing important theoretical aspects and advanced applications.

The success of the workshop was assured by the team efforts of sponsors, organizers, reviewers and participants. We would like to acknowledge the contributions of the individual Program Committee members and thank the paper reviewers.

Our sincere gratitude goes to the participants of the workshop and all authors of the submitted papers. We are grateful to our sponsors: European Office of Aerospace Research and Development (EOARD) of the U.S. Air Force and the U.S. Office of Naval Research Global (ONRGlobal) for their generous support.

We also wish to express our gratitude to the Springer LNCS team managed by Alfred Hofmann for their help and cooperation.

September 2007

Vladimir Gorodetsky
Igor Kotenko
Victor Skormin

Organization

Workshop Chairmen

General Chairs

Rafael M. Yusupov	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
Robert L. Herklotz	US Air Force Office of Scientific Research, USA

Program Committee Co-chairs

Vladimir Gorodetsky	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
Igor Kotenko	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS), Russia
Victor Skormin	Binghamton University, State University of New York; National Research Council's Senior Research Associate with the Air Force, USA

Program Committee

Julien Bourgeois	University of Franche-Comte, France
David Chadwick	University of Kent, UK
Shiu-Kai Chin	Syracuse University, USA
Howard Chivers	Cranfield University, UK
Christian Collberg	University of Arizona, USA
Dipankar Dasgupta	University of Memphis, USA
Naranker Dulay	Imperial College London, UK
Dieter Gollmann	Hamburg University of Technology, Germany
Dimitris Gritzalis	Athens University of Economics and Business, Greece
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Ming-Yuh Huang	The Boeing Company, USA
Sushil Jajodia	George Mason University, USA
Angelos Keromytis	Columbia University, USA
Victor Korneev	Federal Enterprise "R&D Institute "Kvant", Russia
Klaus-Peter Kossakowski	Presecure Consulting GmbH, Germany

Christopher Kruegel	Technical University of Vienna, Austria
Antonio Lioy	Politecnico di Torino, Italy
Javier Lopez	University of Malaga, Spain
Fabio Martinelli	CNR/IIT, Italy
Catherine Meadows	Naval Research Laboratory, USA
Nasir Memon	Polytechnic University Brooklyn, USA
Ann Miller	University of Missouri - Rolla, USA
Nikolay Moldovyan	Specialized Center of Program Systems "SPECTR", Russia
Wojciech Molisz	Gdansk University of Technology, Poland
David Nicol	University of Illinois at Urbana-Champaign, USA
Yoram Ofek	University of Trento, Italy
Monika Oit	Cybernetica, Estonia
Udo Prayer	IAIK, Austria
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany
Andrei Sabelfeld	Chalmers University of Technology, Sweden)
Ravi Sandhu	George Mason University and NSD Security, USA
Antonio Gomez Skarmeta	University of Murcia, Spain
Anatol Slissenko	University Paris-12, France
Michael Smirnov	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Igor Sokolov	The Institute of Informatics Problems of the Russian Academy of Sciences, Russia
Douglas Summerville	Binghamton University, USA
Shambhu Upadhyaya	University at Buffalo, USA
Alfonso Valdes	SRI International, USA
Vijay Varadharajaran	Macquarie University, Australia
Valery Vasenin	Moscow State University, Russia
Paulo Verissimo	University of Lisbon, Portugal
Diego Zamboni	IBM, Switzerland
Peter Zegzhda	St. Petersburg Polytechnical University, Russia

Reviewers

Elli Androulaki	Columbia University, USA
Daniele Beauquier	University Paris-12, France
Julien Bourgeois	University of Franche-Comte, France
David Chadwick	University of Kent, UK
Nikolaos Chatzis	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Shiu-Kai Chin	Syracuse University, USA
Howard Chivers	Cranfield University, UK
Christian Collberg	University of Arizona, USA

Dipankar Dasgupta	University of Memphis, USA
Catalin Dima	University Paris-12, France
Naranker Dulay	Imperial College London, UK
Dieter Gollmann	Hamburg University of Technology, Germany
Vladimir Gorodetsky	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, Russia
Dimitris Gritzalis	Athens University of Economics and Business, Greece
Stefanos Gritzalis	University of the Aegean, Greece
Alexander Grusho	Moscow State University, Russia
Thomas Hirsch	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Ming-Yuh Huang	The Boeing Company, USA
Sushil Jajodia	George Mason University, USA
Angelos Keromytis	Columbia University, USA
Victor Korneev	Federal Enterprise “R&D Institute “Kvant”, Russia
Klaus-Peter Kossakowski	Presecure Consulting GmbH, Germany
Igor Kotenko	St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, Russia
Christopher Kruegel	Technical University of Vienna, Austria
Regine Laleau	University Paris-12, France
Wei-Jen Li	Columbia University, USA
Antonio Lioy	Politecnico di Torino, Italy
Javier Lopez	University of Malaga, Spain
Fabio Martinelli	CNR/IIT, Italy
Catherine Meadows	Naval Research Laboratory, USA
Nasir Memon	Polytechnic University Brooklyn, USA
Ann Miller	University of Missouri - Rolla, USA
Nikolay Moldovyan	Specialized Center of Program Systems “SPECTR”, Russia
Wojciech Molisz	Gdansk University of Technology, Poland
David Nicol	University of Illinois at Urbana-Champaign, USA
Peter Ochsenschleger	Fraunhofer Institute for Secure Information Technology SIT, Germany
Yoram Ofek	University of Trento, Italy
Monika Oit	Cybernetica, Estonia
Carla Piazza	University of Udine, Italy
Udo Prayer	IAIK, Austria
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Roland Rieke	Fraunhofer Institute for Secure Information Technology SIT, Germany

Carsten Rudolph	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Alejandro Russo	Chalmers University of Technology, Sweden
Andrei Sabelfeld	Chalmers University of Technology, Sweden
Antonio Gomez Skarmeta	University of Murcia, Spain
Anatol Slissenko	University Paris-12, France
Dieter Gollmann	Hamburg University of Technology, Germany
Michael Smirnov	Fraunhofer-Gesellschaft Institute FOKUS, Germany
Yingbo Song	Columbia University, USA
Douglas Summerville	Binghamton University, USA
Shambhu Upadhyaya	University at Buffalo, USA
Alfonso Valdes	SRI International, USA
Vijay Varadharajaran	Macquarie University, Australia
Frederik Vercauteren	Katholieke Universiteit Leuven, Belgium
Paulo Verissimo	University of Lisbon, Portugal
Diego Zamboni	IBM, Switzerland
Peter Zegzhda	St. Petersburg Polytechnical University, Russia
Hang Zhao	Columbia University, USA

Table of Contents

Surreptitious Software: Models from Biology and History

Christian Collberg[1,⋆], Jasvir Nagra[2,⋆⋆], and Fei-Yue Wang[3]

[1] Department of Computer Science, University of Arizona, Tucson, AZ 85721, USA
christian@collberg.com

[2] Dipartimento di Informatica e Telecomunicazioni, University of Trento, Via Sommarive 14, 38050 Povo (Trento), Italy
jas@nagras.com

[3] Key Lab for Complex Systems and Intelligence Science, Institute of Automation, Chinese Academy of Sciences, ZhongGuanCun East Road 95, Beijing, Haidian, People's Republic of China
feiyue@gmail.com

Abstract. Over the last decade a bewildering array of techniques have been proposed to protect software from *piracy*, *malicious reverse engineering*, and *tampering*. While we can broadly classify these techniques as *obfuscation*, *watermarking/fingerprinting*, *birthmarking*, and *tamperproofing* there is a need for a more constructive taxonomy. In this paper we present a model of *Surreptitious Software* techniques inspired by defense mechanisms found in other areas: we will look at the way humans have historically protected themselves from each other and from the elements, how plants and animals have evolved to protect themselves from predators, and how secure software systems have been architected to protect against malicious attacks. In this model we identify a set of primitives which underlie many protection schemes. We propose that these primitives can be used to characterize existing techniques and can be combined to construct novel schemes which address a specific set of protective requirements.

Keywords: Software protection, defense mechanisms, taxonomy.

1 Introduction

Your computer program can contain many different kinds of secrets that you may feel need to be protected. For example, you may want to prevent a competitor from learning about a particularly elegant algorithm. You therefore *obfuscate* our program, i.e. make it so convoluted and complex that reverse engineering it becomes a daunting task. Or, you may want to bind the copy sold to each person who buys it to prevent them from illegally reselling it. You therefore *fingerprint* the program, i.e. embed a unique identifier into each copy you sell,

⋆ Supported in part by the Institute of Automation, Chinese Academy of Sciences.

⋆⋆ Supported by the European Commission, contract N° 021186-2, RE-TRUST project.

allowing you to trace a pirated copy back to the original purchaser. Or, you may want to prevent a user from running a program after he has manipulated it, for example by removing a license check. You therefore *tamperproof* the program, i.e. make it unexecutable/self-destructing/self-repairing if it detects that its code has changed. Or, you may want to detect if part of your program has been incorporated into your competitor's program. You therefore check for *birthmarks*, unique characteristics of your code, within your competitor's code.

These techniques have collectively been referred to as *intellectual property protection of software*, or *software protection*, or *whitebox cryptography.* However, we will henceforth refer to the area as *Surreptitious Software.*

Over the last decade many algorithms have been proposed to protect secrets in programs. Seeing as the area has been (and is still) in a great deal of flux, a core set of ideas and techniques on which these algorithms are built has yet to be identified. It is the purpose of this paper to serve as a starting point for constructing such a classification scheme. Our goal is to identify a set of primitives which can be used to build algorithms protecting secrets in programs, and to use these primitives to model and classify software protection schemes that have been proposed in the literature. It is our hope that this model will provide a uniform language for researchers and practitioners, making it easier to discuss existing protection schemes and to invent new ones.

In software engineering, researchers have developed the concept of "design patterns" [1] to capture the rules-of-thumb that regularly occur during the development of large pieces of software. Garfinkel [2] also describes user-interface design patterns for security applications. The models of attacks and defenses we will describe in this paper are similar. Our motivation for identifying and classifying software protection schemes is to eliminate the need to develop new schemes from first principles. Instead we seek to model attacks and defenses that occur repeatedly so experiences and solutions can be reused. We hope that as a result, the insights gained from defending against any one instance of an attack can be generalized to the entire class of defenses.

We will seek inspiration for this model from defense mechanisms found in nature, from the way humans have protected themselves from each other and from the elements, and from protection schemes found in software systems. We will see how, since the dawn of time, plants, animals, and human societies have used surreptition to protect themselves against attackers, and then see how (or if) these ideas can be applied to the intellectual property protection of software.

The model we present here is still in its infancy. In particular, to complement our model of the techniques used by the *defender* we're still working to model the techniques used by the *adversary.* Our ultimate goal is a model which will allow us to classify a proposed new software protection scheme as

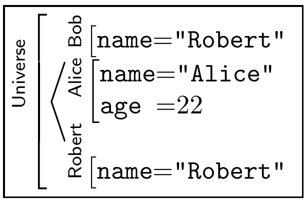
1. a simple variant of another, previously published scheme, or,
2. a novel combination of two known schemes, which we can predict will have certain properties, or
3. a novel scheme not fitting the model, forcing us to reconsider the model itself.

Ideally, we would like the model to *predict* new ways of protecting software, i.e. we should be able to deduce from the model if there are combinations of techniques that will produce a protection scheme with a particular set of required properties.

The remainder of the paper is organized as follows: In Section 2 we describe our notation and in Section 3 the intents of an attacker. In the main section, Section 4, we present the protection primitives available to the defender. In Section 5 we summarize our model and point out directions for future research.

2 Notation

Our model consists of a universe of objects. The attacks we are concerned with and defenses against these attacks are modeled as a series of transformations on these objects. We represent all objects as *frames*. Frames are knowledge representation devices well-known from the AI community. A frame consists of a unique identifier and one or more *slots*, which are *name=value*-pairs. We use a graphical notation to illustrate frames and frame transformations:



Each frame is introduced by a left square bracket ([), and the object's unique identifier is written vertically to the left of the bracket.[1]

The slots of a frame describe properties of an object. A value can be a scalar such as a string or an integer, a list of objects, another frame, or a reference to another frame. A special list slot is the *contains* slot, which, when present, indicates that a particular object contains other objects. The *contains* slot is indicated by a left angle bracket (⟨). In the example above the universe itself is such a frame, where the *contains* slot has three objects, `Alice`, `Bob`, and `Robert`.

In our model, protection strategies are functions which map frames to frames. We start out with a universe of unprotected objects and proceed by a sequence of protective transformations that make the objects progressively safer from attack. Layering protection schemes is modeled by function composition of the frame transformations. *Attacks* are also modeled by frame-to-frame transformations: they try to subvert protection strategies by *undoing* layers of protection.

As an example, consider the Leatherback turtle (Dermochelys coriacea) which buries its eggs in the sand to protect them from predators and the elements. In Figure 3 (a), the initial, unprotected scenario is a frame where the universe contains a *a turtle*, *sand*, and *eggs*. Then, we apply the **cover** protection primitive,

[1] In AI, frames can be of various types called "abstractions". For example, "Bob" can have "human" and "attacker" abstractions. However, for our purposes of this paper we do not require this additional classification.

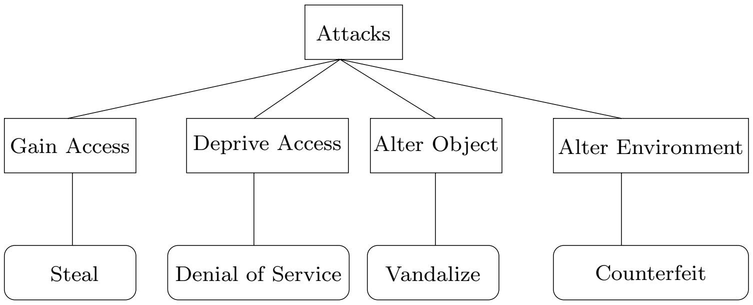


Fig. 1. Types of attacks

the sand frame is given a *contains* slot, and this slot now holds the turtle-eggs frame. This indicates that a predator can no longer see the turtle-eggs, them now being covered by sand. In other words, an object can only sense what's in its own environment—it cannot look inside other objects.

Each slot can also have associated with it a set of *demons*. These are actions which fire under certain circumstances. For example, to protect its young the Cane toad (Bufo marinus) lays eggs that are poisonous. We would represent this form of protection with a demon attached to a toad egg saying "**when** *eaten* **then** *cause harm to attacker*".

The model uses seven *basic operations* to construct frames and frame transformations. The `new` and `delete` operators create and destroy frames from the universe respectively. As in the real universe objects are never actually created or destroyed, but manufactured from existing material. We model this by giving `new` and `delete` access to a special *source* object which provides an infinite supply of raw-materials and a *sink* object into which excess material can be disposed. The `getprop`, `setprop` and `delprop` operators get, set and delete properties of a given frame. The `move` operator relocates an object represented by a frame from a particular position in one object to a different position within a possibly different object. Our final operator, `apply` is *function application*.

These seven basic operations allow us to build the universe of unprotected objects that we are interested in as well as the protective primitives that we describe. However, we do not give their explicit formulations in this paper.

3 Attacks

In order to describe protection schemes we require a corresponding model of the attacker. Protection is an act of defense against a perceived attack. The type of protection a scheme provides depends heavily on which types of attack are of interest. For example, in the case of the Leatherback turtle hiding its eggs, the threat being defended against is a predator stealing and eating the eggs, depriving the parent turtle of its offspring. In other cases, (like the Common Cuckoo, (Cuculus canorus)), an attacker may be interested in adding their own eggs to the brood which appear similar to the nest owner's eggs. The intent of this attacker is to pass on their genes with minimal labor, by mimicking the object of value.

Given an object of value O, we can taxonomize the attacks by considering the different ways an attacker may derive value from O. The intent of an attacker may be to: gain unauthorized access to O, deprive legitimate users access to O, alter O, or alter the environment in which O occurs.

In Figure 1, a variety of attacks are shown classified into four different types. In a stealing attack, an attacker simply takes the object of value, while in a denial-of-service attack, an attacker deprives legitimate access to an object overwhelming it. Vandalizing a piece of property is an example of devaluing an object by altering it. Counterfeiting alters the environment of an object by filling it with imitations which in turn can reduce the value of the original.

The primitives that we will build in the next section are designed to prevent each of these types of attacks, irrespective of the context in which they occur. For example, in the next section we will introduce a **cover** operator. In the biological context, a Leatherback turtle can defend unauthorized access to its eggs by hiding them in sand using the **cover** operator. Similarly, in the historical context, pirates would hide their treasure by burying their treasure on a remote island using the same **cover** operator. In software, we may embed a picture inside a document to hide it from prying eyes. The document serves as a **cover** for the picture. In all three instances, the pattern of defense remains the same — hide the object of value using another object — even though the context may have changed.

4 Defenses

In Figure 2 we illustrate the eleven primitive operations of the defense model. Each primitive is a transformation from an unprotected universe of objects to a protected one. In the remainder of this section we will present each primitive in turn and illustrate with scenarios from biology, human history, computer security, and surreptitious software.

4.1 The Cover Primitive

The most fundamental way of defending against all four types of attacks is to hide an object by **covering** it with another object. If the attacker does not have access to an object, it becomes difficult for him to attack it. This may seem like a trivial observation, but **covering** is an operation that occurs frequently in the natural, human, and computer domains. In Figure 3 (a), we illustrate our earlier biological example of the Leatherback turtle. In Figure 3 (b), we show an historical example of **covering** occurring in in 499 BC, when, to instigate a revolt against the Persians, Histiaeus tattooed a message on the shaved head of a slave, waited for the hair to grow back, and sent him to Axristagoras who shaved the slave's head again to read the message.

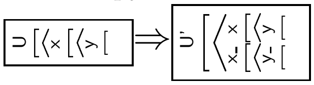
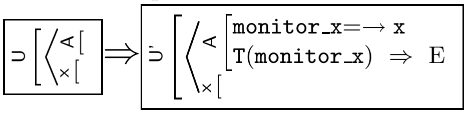
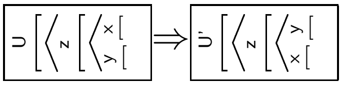
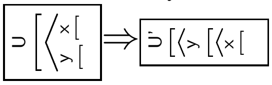
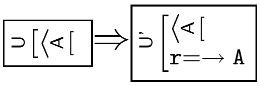
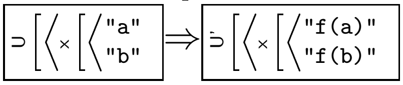
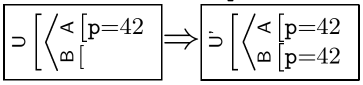
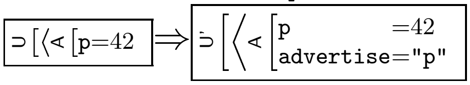
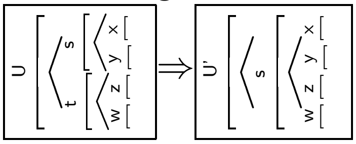
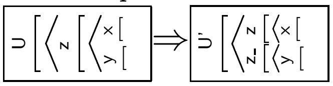
compose f g $(f \circ g)(x) = f(g(x))$ Compose primitives f and g.	**dynamic** x f U $x \Rightarrow fx \Rightarrow f(fx) \Rightarrow f(f(fx)) \ldots$ Repeatedly apply primitive f to object x in U.
copy x "_" U  Make of deep copy of x, adding "_" to keep names unique.	**tamperProof** A T x E U  When T happens to x, A takes action E.
reorder z f U  Reorder the elements of frame z according to function f.	**cover** x y U  Move x from its environment into/under y.
indirect A r U  r is an indirect reference to A.	**map** x f U  Apply function f to the components of x.
mimic A B p U  Copy property p from A to B.	**advertise** A p U  Make A's property p public.
merge s t U  Merge t into s, removing t from U.	**split** z f U  Split z into two parts, z and z_. Function f returns True for elements which go in z.

Fig. 2. Primitives of the protection model

In the software world, we can cover a file or some data by putting it inside another file. For example, when mail systems initially began scanning attachments for viruses, virus writers responded by zipping (and occasionally encrypting) the virus before emailing them. The zip file served as **cover** for the virus itself (Figure 3 (c)). In order to counter this cover, mail servers today have to "look under" every possible cover by unpacking and scanning every archive (including archives containing archives) to find potential viruses.

A more obvious instance of **covering** occurs in systems that use hardware to protect software. For example, the military uses hardened boxes to contain

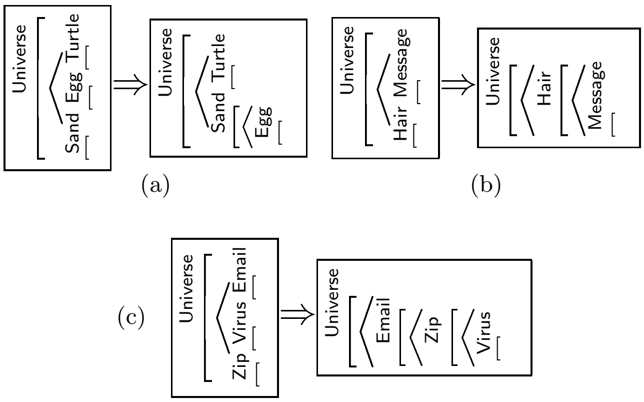


Fig. 3. Applying the **cover** primitive in the biological, historical, and software domains

computers on which sensitive software is run. These boxes are also **tamperproofed** so that they will explode (known by the military as "rapid disassembly") if someone unauthorized tries to open them. **Tamperproofing** is in fact one of the primitives in our model, and we will discuss this further in Section 4.9.

4.2 The Copy Primitives

The **decoy** primitive makes the environment larger to force an attacker to consider more items while searching for a secret or to draw attention away from the real secret. The **clone** primitive adds a copy of the secret itself to the universe in order to force an attacker to destroy both copies. The difference between **clone** and **decoy** is merely one of intent, and in Figure 2 we therefore merge these primitives into one, **copy**.

In the animal kingdom, **decoying** and **cloning** are common protection strategies. An animal low on the food-chain will often use **cloning** to protect itself, or, rather it's own DNA. For example, the Pufferfish (Arothron Meleagris) spawns 200,000 offspring in the hope that at least *some* will survive and carry on its parents' legacy.

The California newt (Taricha torosa) combines **cloning**, **tamperproofing**, and **covering** to protect its progeny. The newt lays 7-30 eggs, each covered by a gel-like membrane containing tarichatoxin for which no known antidote exists.

Scientists are still unsure exactly why zebras have stripes, but one favorite theory suggests that they are used as part of a **decoying** strategy: when a zebra is running in a herd of similarly-striped animals it's difficult for a lion to pick out one individual to attack. As we'll see later, **mimicking** is another possible explanation.

Decoys are, of course, common in human warfare: "In World War II, the U.S. created an entire dummy army unit in southern Britain to convince the Germans that the Normandy invasion would be directed toward the Pas-de-Calais" and

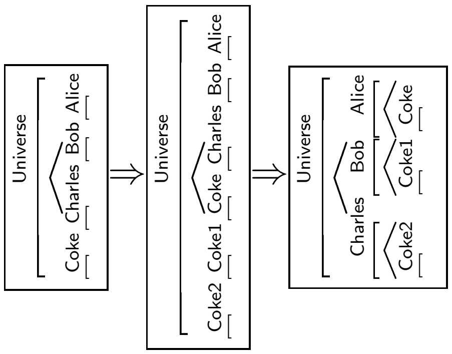


Fig. 4. Applying the **cloning** primitive to protect the Coca-Cola recipe. We've **cloned** twice and **covered** three times, resulting in a situation where the three executives are each holding a copy of the recipe in their pockets.

during Gulf War I, the Iraqis "employed mock tanks, airplanes, bunkers and artillery" made up of "plywood, aluminum and fiber glass" [3].

A well-known legend states that the recipe for Coca-Cola is protected by **cloning** it over three people, none of whom may ever travel on the same plane (Figure 4).

In the recent TV mini-series *Traffic*, terrorists attempt to smuggle smallpox virus into the United States. The authorities are conned into believing that the smallpox will enter the country inside a shipping container with a particular identification number, but are forced into a nationwide hunt once they realize that the terrorists have shipped several containers with *the same number*! This is a classic use of **decoying**, but later in this section you will see how the terrorists combined this ruse with the **advertise** primitive to further confuse the DEA.

Decoys can be used by software watermarking algorithms to protect the mark. In the very simplest case you just add a variable with a conspicuous name, like this:

```
int THE_WATERMARK_IS_HERE = 42;
```

This draws attention away from the actual watermark, if only for a short while. You could also add a number of fake watermarks, some really obvious, some less so, to force the attacker to spend valuable time examining them all, and having to decide which one is real and which ones are fakes. The **cloning** primitive can also be used to embed multiple copies of the mark (possibly by using different embedding algorithms), thus forcing the attacker to locate and destroy all of them.

In code obfuscation, **cloning** is also a common operation. For example, a function f could be cloned into f', the clone obfuscated into f'', and different call sites could be modified to either call f or f'' [4]. This gives the appearance that calls to f and f'' actually call different functions, both of which the attacker needs to analyze in order to gain an understanding of the program. A basic block B in a function can similarly be cloned into B', obfuscated into B'', and an

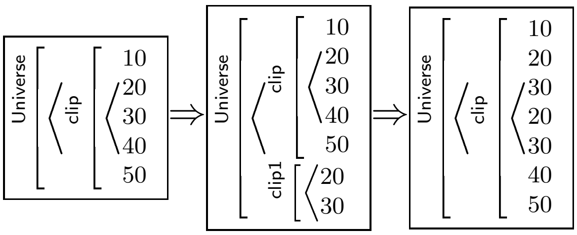


Fig. 5. Watermarking an image using echo hiding, by applying the **split** and **merge** primitives

opaque predicate can be inserted to alternate between executing the two blocks: `if` $P^?$ `then` B `else` B'' [5].

4.3 The Split and Merge Primitives

Split and **merge** are two common protection transformations, particularly in software situations where these are simple to implement. **Split** breaks up an object into smaller parts each of which are easier to hide or protect. **Merge** blends two unrelated objects together to make it appear as if they belong together. To create mass confusion, splitting and merging are often used in conjunction: take two unrelated objects A and B and split them into parts A_1, A_2, B_1, and B_2, then merge A_1 with B_1 and A_2 with B_2.

When attacked, some animal species will split themselves and let the attacker have one part for dinner. This is known as *autotomy* [6]. This technique is particularly useful when the species is also able to *regenerate* the lost part. In our classification this is a form of **tamperproofing** which we will discuss further in Section 4.9 below.

Squirrels use so-called *scatter hoarding* to **split** up the food they've gathered and cache it at different locations in their territory. The larger Gray Squirrel (Sciurus carolinensis) is known to steal the caches of the European Red Squirrel (Sciurus vulgaris), but the **split** makes it less likely that all the food will be lost.

Human organizations frequently split themselves up into groups to prevent an adversary from destroying the entire organization in one attack. This is true, for example, of terrorist networks which split themselves up in into smaller, autonomous cells. Each cell is less conspicuous than the whole network, and an attack (from an outside force or an inside informant) will affect only the cell itself, not the network as a whole.

Some software watermarking algorithms **split** the watermark into several smaller pieces such that each piece can be embedded less conspicuously [7,8]. **Merging** is also a natural operation, since the watermark object has to be attached to some language structure already in the program.

Echo hiding is an audio watermarking technique in which short echoes (short enough to be undetectable to the human ear) are used to embed marks. A really short echo encodes a 0 and a longer one a 1. In our model, embedding a single-bit watermark m is accomplished by a) settling on an embedding location

p, b) **copying** D_0 bits for a 0 and D_1 bits for a 1 from p, and c) **merging** the copy back into the clip. In this embedding example, shown in Figure 5, $D_0 = 2, D_1 = 3, p = 2, m = 0$.

Obfuscation algorithms make extensive use of **splitting** and **merging** to break up, scatter, and merge pieces of a program, thereby creating confusion. For example, to confuse someone trying to reverse engineer a program containing two functions f and g, each can be split in two parts (yielding f_a, f_b, g_a, g_b) and then the parts can be merged together forming two new functions $f_a||g_b$ and $f_b||g_a$. Any calls to f or g will have to be modified, of course, to call the new functions, and to call them in such a way that only the relevant parts are executed [4].

Two common protection strategies are **shrink** and **expand**. Shrinking oneself to become as small as possible makes it easier to hide in the environment. Expanding, on the other hand, is the operation of making oneself as large as possible, larger objects being harder to destroy than smaller ones. **Shrink** and **expand** are just variants of **split** and **merge**, however, so we don't include them as primitive operations. An object shrinks itself by splitting into two parts, one essential and one superfluous, disposing of the fluff. Expansion, similarly, is merging yourself with some new building material grabbed from the surrounding environment.

The Pufferfish uses a combination of several defense strategies. One is to **expand** its size (by inflating itself with water or air from its surroundings) in order to appear threatening or a less easy target to an attacker. The blue whale has grown so big that it only has one natural predator, the Orca.

That *size* is an important feature of hiding or protecting a secret is evident from the expression *"as hard as finding a needle in a haystack."* Implicit in this expression is that the needle is small and the haystack big. Hiding the *Seattle Space Needle* in a normal size haystack would be hard, as would hiding a sewing needle in a stack consisting of three pieces of hay. So, a common strategy for hiding something is to **shrink** it, such as a spy compressing a page of text to a microdot. This is often combined with **decoying**, increasing the size of the environment, for example using a really big haystack in which to hide the needle or hiding the microdot in the Bible instead of in the 264 words of the Gettysburg address. Expansion is, of course, also a common human strategy. A bigger vault is harder to break into than a smaller one, a bigger bunker requires a bigger bomb to destroy it, and in a crash between an SUV and a VW Beetle one driver is more likely to walk away unscathed than the other.

The principle behind LSB (Least Significant Bit) image watermark embedding is that the low order bits of an image are (more or less) imperceptible to humans and can be replaced with the watermarking bits. In our model, this is a **shrink** operation (shrinking the image gets rid of the imperceptible fluff) followed by **merging** in the watermarking bits.

It is important not to keep passwords or cryptographic keys in cleartext in computer memory. An attacker who gets access to your machine can search through memory for tell-tale signs of the secret: a 128-bit crypto key, for example,

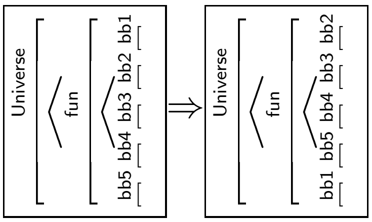


Fig. 6. Watermarking a program by applying the **reorder** primitive

has much higher entropy than ordinary data and will be easy to spot [9]. There are two simple ways of fixing this problem: either increase the entropy of the data surrounding the key (for example by encrypting it) or decrease the entropy of the key itself. The latter can be accomplished for example by **splitting** the key into several smaller pieces which are then spread out over the entire program. Thus, this is an application of the **split** and **reorder** primitives. We will examine **reordering** next..

4.4 The Reorder Primitive

The **reorder** transformation can be used to place objects in a random order, thereby *sowing* confusion. But a reordering can also *contain* information. Think of 007 entering a room and finding that the gorgeous blonde Russian operative who was just abducted left him a message: the martini glass has been moved to the *left* of the Baretta, letting James know that he needs to be on the lookout for Blofeld.

In various cons such as the *shell game* or *three-card-monty*, the secret (the *pea* or the *queen*, respectively) is hidden from the victim by sequences of deft **reorderings** by the con-man. This combines **reorder** with the **dynamic** primitive which we will see in Section 4.10.

Many watermarking algorithms, in media as well as software, make use of **reordering** to embed the mark. The idea is that a watermark number can be represented as a permutation of objects. For example, in the Davidson-Myhrvold software watermarking algorithm the watermark is embedded by permuting the basic blocks of a function's Control Flow Graph. In the example in Figure 6, the watermark number 5 is embedded by ordering the permutations of the numbers $[1 \ldots 5]$:

$$[1,2,3,4,5],[2,1,3,4,5],[2,3,1,4,5],[2,3,4,1,5],\underline{[2,3,4,5,1]},\ldots$$

picking the 5th permutation to reorder the basic blocks.

Many code obfuscation algorithms also make use of the **reorder** primitive. Even when there is an arbitrary choice on how to order declarations or statements in a program, programmers will chose the "most natural" order over a random one. Randomly reordering functions within a program will destroy the information inherent in the grouping of functions into modules, and the order of functions within modules.

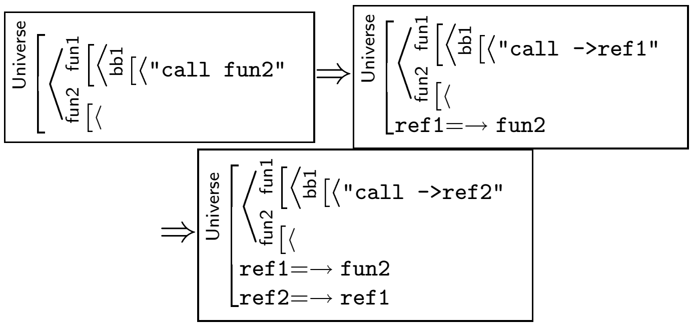


Fig. 7. Obfuscating a program by applying the **indirect** primitive

Of course, **reorder** can also be used as an attack: to destroy a watermark based on ordering all the attacker has to do is apply the algorithm again, destroying the previous ordering at the same time as he's inserting his own mark.

4.5 The Indirect Primitive

In baseball, the coach uses hand signs to indicate to the batter whether to hit away, bunt, etc. To prevent the opposing team from *stealing* the signs the coach will use **decoying** as well as adding a level of **indirection**. The actual sign is **merged** with a sequence of bogus decoy signs and a special *indicator* sign. The indicator gives the location of the actual sign, typically the one following directly after the indicator.

Indirection is also a common adventure movie plot device. In *National Treasure*, Nicolas Cage is lead by a sequence of increasingly far-fetched clues (a frozen ship in the arctic, a meerschaum pipe, the Declaration of Independence, a $100 bill, etc.), each one pointing to the next one, eventually leading to the final location of the hidden treasure (a Freemason temple).

Like our other primitives, **indirection** is often used in combination with other protection strategies. Our hero will find that the next clue is inside a box hidden under a rock (**covering**), that there are many boxes under many rocks (**decoying**), that the box will explode unless he unlocks it just the right way (**tamperproofing**), and he that needs to find *both* pieces of the clue in order to locate the treasure (**splitting**).

One of our particularly clever friends came up with the following scheme for protecting her foreign currency while backpacking around Europe. First, she sewed the different currencies (this was before the European Union adopted the Euro) into different parts of her dress, a form of **covering**. Next, in order to remember where the Deutchmark, French Francs, etc., were hidden, she wrote down their locations on a piece of paper (**indirection**). Of course, if someone were to find this note she'd be in trouble, so she wrote it in French (**mapping**) using the Cyrillic alphabet (another **mapping**), banking on few common thieves being fluent in both Russian and French. She never lost any money.

Since exact pointer analysis is a hard problem [10], **indirection** has also become a popular technique in protecting software [5,11]. The idea is to replace a language construct (such as a variable or a function) by a reference to it. This adds a confusing level of indirection. Just like Nicholas Cage running around chasing one clue after another, an attacker analyzing a protected program would have to chase pointer-to-pointer-to-pointer until finally reaching the real object. Consider the example in Figure 7. Here we start out with two functions `fun1` and `fun2`, where `fun1` contains one instruction, a call to `fun2`. We apply the **indirect** primitive to add an indirect reference `ref1`, and then the **map** primitive to replace all references to `fun2` by an indirect reference through `ref1`. We then repeat the process to force an attacker to unravel two levels of indirections.

4.6 The Map Primitive

The **map** primitive typically protects an object by adding confusion, translating every constituent component into something different. If the mapping function has an inverse, this obviously needs to be kept secret.

Translation is a form of **mapping** where we map every word m in a dictionary to another word r, usually in a different language, keeping the mapping secret. The Navajo Code talkers are famous for their rôle in World War II, conveying messages in a language unfamiliar to the Japanese. Humans use this trick to confuse outsiders, or often to build cohesion among members of a group. Consider, for example, the `l33t` language used by youths in online communities. It sets them apart from others who don't know the language, but it also protects their communication from outsiders (such as their parents).

Obfuscated language mappings occur in many professional fields as well, including computer science, medicine, and law. Steven Stark [12] writes in the Harvard Law Review that *"one need not be a Marxist to understand that jargon helps professionals to convince the world of their occupational importance, which leads to payment for service."* In other words, obfuscating your professional language protects you from competition from other humans of equal intelligence, but who have not been initiated into your field.

The most obvious way to protect a secret is to not tell anyone about it! This is sometimes known as *security-through-obscurity.* An example is the layout of many medieval cities in the Middle East. At first it may seem that there is no good reason for their confusing alleyways until you realize that the lack of city planning can actually be a feature in the defense of the city. Without a map only those who grew up in the city would know how to get around — an obvious problem for attackers. But this strategy only works in an environment with poor communications; as soon as there's a *Lonely Planet* guide to your city your attempts to use secrecy to protect yourself will have failed.

Figure 8 shows the simplest form of code obfuscation, *name obfuscation* [13], a form of translation that replaces meaningful identifiers with meaningless ones.

A common way of watermarking English text is to keep a dictionary of synonyms, replacing a word with an equivalent one to to embed a 0 or a 1 [14]. See Figure 9 where we've used a dictionary to embed a 1 at position 2 in the text.

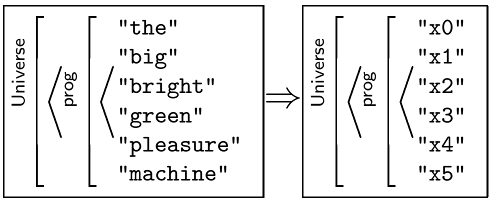


Fig. 8. Applying the **map** primitive to rename identifiers

Cryptographers use the terms *confusion* and *diffusion* to describe properties of a secure cipher [15]. Confusion is often implemented by *substitution* (replacing one symbol of the plaintext with another symbol) which, in our terminology is a **mapping**. Diffusion is implemented by *transposition* (rearranging the symbols), what we call **reordering**. A *product cipher* creates a secure cipher from **compositions** of simple substitutions and transpositions.

4.7 The Mimic Primitive

Mimicry is, of course, the greatest form of flattery. We can use it as many different forms of protection. *Camouflage*, for example, is trying to look like an object in your environment. Mimicry can also be a deterrent — you can try to look like someone or something you're not, to scare off attackers. In our model, the **mimic** primitive simply copies a property from one object to another.

The chameleon is probably the animal most well-known for use of camouflage, **mimicking** the colors of items in its background in order to avoid detection. Many other animals avoid predators in a similar way. As we noted earlier, scientists are not sure exactly why the Zebra has stripes. One theory is that lions (being color blind) cannot pick out a zebra from the background when it stands still in high grass, i.e. the zebra **mimics** it's background. Another theory contends that the stripes confuse Tse-tse flies (Glossina palpalis).

Peter Wayner's *mimic functions* [16] create new texts which steganographically encode a secret message. In order not to arouse suspicion, the texts are made such that they **mimic** texts found in our daily lives, such as transcripts of a baseball announcer, shopping lists, or even computer programs. If the mimicry is good, i.e. if the statistical properties of a generated text are close enough to those of real texts, then we will be able to send secret messages across the Internet without anyone noticing.

The same technique has been been used by spies. For example, during World War II, Japanese spy *Velvalee Dickinson* wrote letters about her doll collection to addresses in neutral Argentina. When she was writing about dolls she actually was talking about ship movements. She was eventually caught and jailed.

In 2004, during a vehicle search, Hungarian police found what appeared to be a fake Viagra tablet. Apart from being pink, it was rhombus shaped, had Pfizer imprinted on one side and VGR50 on the other — clearly a bad Viagra counterfeit. However, further analysis revealed that the tablet was just **mimicking**

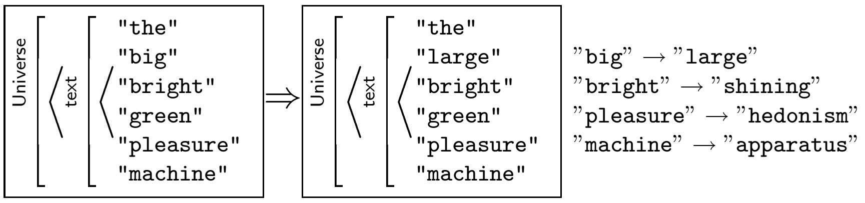


Fig. 9. Applying the **map** primitive to embed a watermark in English text

Viagra and instead contained amphetamine. See Figure 10. Here we've applied the **mimic** primitive three times, once for each attribute. Why the drug smugglers did not also **mimic** the highly recognizable blue color of Viagra tablets is hard to know, but it's possible that they thought that appearing to be a bad counterfeit would make it less likely for the tablet be tested for illegal substances.

A central concept in surreptitious software is *stealth.* Any code that we introduce as a result of the protection process must fit in with the code that surrounds it, or it will leave tell-tale signs for the attacker to search for. In other words, the new code must **mimic** aspects of the original code written by humans. There can be many such aspects, of course: the size of the code, the instructions used, the nesting of control structures, and so on, and a stealthy protection algorithm must make sure that the introduced code **mimics** every aspect. By embedding a watermark in a modified **cloned** copy of an existing function, Monden's [17] software watermarking algorithm **mimics** the style of code already in the program, thereby increasing stealth. As seen in Figure 11 this is actually accomplished using a combination of the **clone** and the **map** primitives.

4.8 The Advertise Primitive

By default, our model assumes that objects keep all information about themselves secret. In other words, Alice can see what other objects are around her, but she only knows their identities, she can't look *inside* them. This is how we normally protect ourselves: We call the primitive which breaks this secrecy **advertise**. In our model it is represented by a special property `advertise` which lists the names of the properties visible to the outside.

One way **advertising** assists the defenses of an object is by identifying itself to its defenders. Flamingoes (Phoenicopterus ruber ruber) bring up their young in large crèches, however, parents feed and take care of only their own chicks. The chicks **advertise** their appearance and vocalization. These characteristic allow parents to recognize and defend their own progeny. In software, similar birthmarks [18] — unique characteristics of a program — allow an author to detect piracy by recognizing characteristics of their own programs in other software.

Another common use of this primitive is for the defender to advertise the strength of his defenses in the hopes that an adversary will stay away. There is nothing, of course, that says the defender must advertise *correct* information! In fact, a common use for this primitive is to lead an attacker astray by feeding them falsehoods.

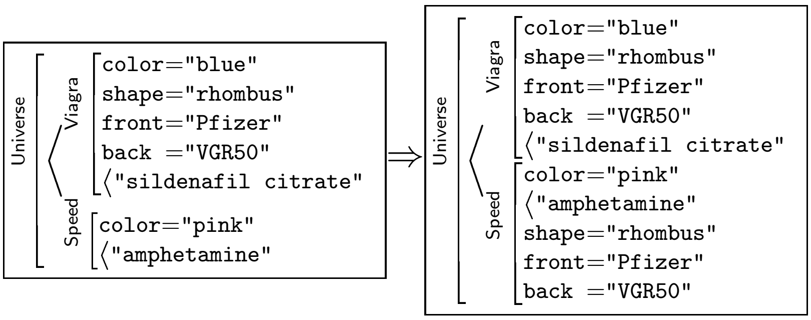


Fig. 10. Using the **mimic** primitive to protect an illegal drug from detection

Toxic species often use bright colors to advertise their harmfulness. This is known as *aposematic coloration* [6]. The easily recognizable red-yellow-black striped coloration pattern of the highly poisonous Coral snake is an example of this. The King snake protects itself by **mimicking** the Coral snake, while actually being non-venomous. Falsely **advertising** "Look, I'm venomous!" will confuse predators to stay away. The transformations are shown in Figure 12.

In many ways, **advertise** can be seen as the opposite of *security through obscurity* — rather than keeping our defenses secret we openly display them to our adversary, taunting them to "go ahead, take your best shot!" In an ideal scenario they will take one look at us, walk away disheartened, and launch an attack against a less well defended target. In a less ideal scenario, knowing details of our defenses will allow them to work out a chink in our armor - like a 2 meter exhaust vent leading straight into the core of our Death Star allowing anyone with a spare proton torpedo to reduce us to so much space debris.

We already mentioned the recent TV mini-series *Traffic*, where terrorists smuggle smallpox virus into the United States, sending the authorities on a wild goose chase across the country hunting for the right container among several **decoy** containers with the same number. In the end, it turns out that the terrorists had used **advertise** ("the virus is in one of these containers `*wink,wink*`"), to trick the DEA: *all* the containers were actually decoys and the actual carriers of the virus were the illegal immigrants on board the cargo ship.

False **advertising** also works in the software protection domain. A program can advertise that it is watermarked, when in fact it isn't, or **advertise** that it is **tamperproofed** using a particular algorithm, when in fact it is using another one.

4.9 The Tamperproof Primitive

Tamperproofing has two parts, detecting that an attack has occurred and reacting to this. The reaction can be a combination of

1. self-destructing (in whole or in part),
2. destroying objects in the environment (including the attacker), or
3. regenerating the tampered parts.

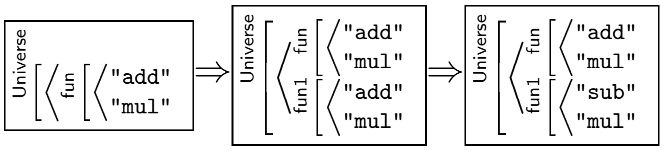


Fig. 11. Stealthy software watermarking using **mimicry**. We first **clone** an existing function and then **map** its instructions to new ones to embed the mark.

Some animals can **regenerate** destroyed parts of their bodies (usually limbs and tails but in some cases even parts of internal organs) after an attack. Starfish can regrow their entire organism given just one arm and the central disk.

A Pufferfish uses a combination of several defense strategies: he can **expand** his size to appear threatening to an attacker, and an attacker who never-the-less starts munching on him will have to deal with the neurotoxins of his internal organs, a form of **tamperproofing**. It may not save an individual Pufferfish's life, but it might save his brothers and sisters since the attacker is either dead or has developed a distaste for Pufferfish.

Turtles and many other animals use exoskeletons (a form of **covering**) to protect themselves. Some combine this with **tamperproofing**, using poisons to make themselves unpalatable once the protective layer has been removed by the predator. The Hawksbill turtle (Eretmochelys imbricata) has both a shell and poisonous flesh.

Humans using tamperproofing to protect themselves is a common movie plot: the terrorist straps explosives to his chest and holds a dead-man's-trigger in his hand so that if the hero takes him out everyone will die in the ensuing blast. Another common plot device is leaving an envelope containing compromising information about your nemesis with your lawyer, with instructions that "in the case of my untimely demise, mail this to the Washington Post."

The terrorist in the above scenario is *self-tamper-detecting* as well as *self-tamper-responding*. This seems to be the most common case: you monitor some part of yourself (the health of your internal organs, for example), and if tampering is evident you execute the tamper-response (releasing the trigger a split second before expiring). The second scenario above shows that both detection and response can be external, however: the lawyer examines the obituaries every day and, if your name shows up, executes the tamper-response.

Unlike lower order animals like newts and salamanders, humans have very little regenerative power — we can only regrow some skin and a part of our liver.

Tamperproofing is, of course, an extremely important part of surreptitious software. A common form of tamper-detection is to compare a hash computed over the program to the expected value [19]. Some tamperproofing algorithms make use of *regeneration* [20]. to fix parts of a program after an attack. The idea is to replace a destroyed part with a fresh copy, as shown in Figure 13.

Other types of response are also common: a program can retaliate by (subtly or spectacularly) self-destructing [21] or destroying files in its environment. For

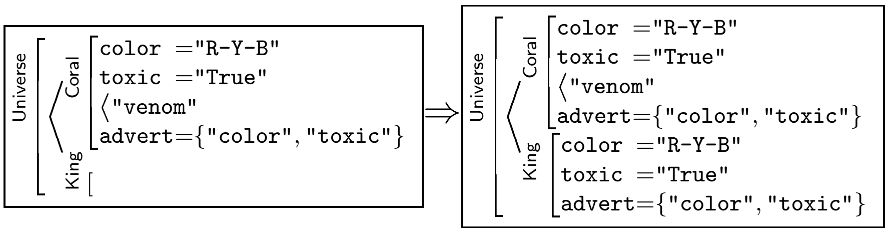


Fig. 12. The King snake applies the **advertise** primitive to fool predators that it contains venom, even though it does not. (`advertise` is here abbreviated `advert`.)

example, the *DisplayEater* screen motion catcher program will delete your home directory if you try to use a pirated serial number [22].

4.10 The Dynamic Primitive

Our final primitive, **dynamic**, is used to model protection techniques which use dynamic behavior to throw off an attacker. Here, *continuous change* itself is used to confuse an attacker. The change can take a variety of forms: fast movement, unpredictable movement, and continuous evolution of defenses.

Moving faster than your adversary is in many ways the ultimate protection technique in the animal world. If the cheetah can't catch you, he can't eat you! There is a trade-off between agility and speed on the one hand and physical layers of protection on the other. A turtle can afford to be slow because he's carrying around a thick protective exoskeleton. The Pronghorn antelope (Antilocapra americana) has made a different trade-off: it can run 98 km/h to get away from a predator such as the mountain lion which is only able to do 64 km/h. On the other hand, the antelope is soft and vulnerable on the outside and should he get caught, it's game over.

Anyone who has tried to kill a particularly annoying fly knows that while speed is important, so is agility and unpredictability. Consider, for example, the fruit fly (Drosophila Melanogaster) which flies in a sequence of straight line segments separated by 90 degree turns, each of which it can execute in less than 50 milliseconds. Since the movements appear completely random, a predator will have to be very lucky and persistent to get the best of him.

Evolution itself is, of course, an exercise is continuous change. As pray develop better defenses, predators develop better attacks. HIV is one example of a particularly successful organism: because of its fast replication cycle and high mutation rate, a patient will experience many virus variants during a single day. This makes it difficult for the immune system to keep up and for scientists to develop effective vaccines.

Mobility is key in modern warfare. During the Gulf War, Iraqi forces successfully moved Scud missiles around on *transporter-erector-launcher* trucks to avoid detection by coalition forces: "even in the face of intense efforts to find and destroy them, the mobile launchers proved remarkably elusive and survivable" [23]. Just like the Pronghorn antelope and the turtle have made different

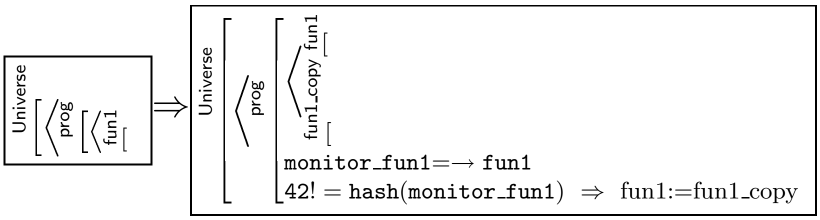


Fig. 13. Using **tamperproofing** to regenerate a part of a program that has been tampered with

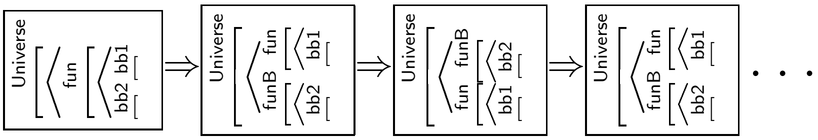


Fig. 14. Aucsmith's tamperproofing algorithm [24] uses the **dynamic** primitive

trade-offs on the scale from *slow-but-impenetrable* to *fast-but-vulnerable*, so have military forces.

Just like in the natural virus world, in the computer virus world change is used to foil detection. A computer virus that modifies itself for every new generation will stress the capabilities of the virus detector: it has to find some signature of the virus that does not change between mutations.

Many software protection algorithms are dynamic, in one form or another. Aucsmith's tamperproofing algorithm [24], for example, first breaks up a program in chunks which are then encrypted. One chunk is decrypted, executed, re-encrypted, swapped with another chunk, and the process repeats, as shown in Figure 14. (Here we're not modeling the encryption. This could be done with the **map** primitive.) This process results in an unpredictable address trace that an attacker will find difficult to follow, making the code hard to tamper with.

The **dynamic** primitive is a higher order function taking an object x and a function f as arguments, iteratively generating $\langle fx, f(fx), f(f(fx))), \ldots \rangle$.

5 Summary

It's an interesting exercise to come up with scenarios — biological, historical, or computational — that can't be expressed using the primitives we've suggested in this paper. Also interesting is to show that a particular primitive can be simulated by a combination of others, and thus eliminated from the model. We invite the community to join us in improving the model in this way.

It should be kept in mind that any Turing complete model (such as a simple Lisp-based model) would do, but the goal here is not to come up with a minimalist set of primitives. Rather, we want to find primitives which elegantly express how the natural world has evolved protection strategies, how we as humans think about protecting ourselves and, most importantly, of course, how we can protect computer programs.

We are currently establishing an *open experimental environment for surreptitious software*, where various malicious attacks, protection techniques, and test programs from any interested parties can be created, launched, and can interact with each other according to certain protocols, leading to a “living laboratory” for evaluating software security through computational experiments [25].

References

1. Cunningham, W., Beck, K.: Using pattern languages for object-oriented programs. In: OOPSLA’87 (1987)
2. Garfinkel, S.: Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA (2005)
3. unknown: Decoys: Tanks but no tanks. Time Magazine (Monday, Feb. 4) (1991), `www.time.com/time/magazine/article/0,9171,972244,00.html`
4. Collberg, C., Thomborson, C., Low, D.: Breaking abstractions and unstructuring data structures. In: IEEE International Conference on Computer Languages 1998, ICCL’98, Chicago, IL (1998)
5. Collberg, C., Thomborson, C., Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs. In: ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages 1998, POPL’98, San Diego, CA (1998)
6. Cronin, G.: Defense Mechanisms. Salem Press, pp. 314–319 (2001)
7. Collberg, C., Huntwork, A., Carter, E., Townsend, G.: Graph theoretic software watermarks: Implementation, analysis, and attacks. In: Workshop on Information Hiding, pp. 192–207 (2004)
8. Collberg, C., Carter, E., Debray, S., Kececioglu, J., Huntwork, A., Linn, C., Stepp, M.: Dynamic path-based software watermarking. In: ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI 04), ACM Press, New York (2004)
9. Shamir, A., van Someren, N.: Playing Hide and Seek with stored keys. In: Franklin, M.K. (ed.) FC 1999. LNCS, vol. 1648, pp. 118–124. Springer, Heidelberg (1999)
10. Ramalingam, G.: The undecidability of aliasing. ACM TOPLAS 16(5), 1467–1471 (1994)
11. Wang, C., Hill, J., Knight, J., Davidson, J.: Software tamper resistance: Obstructing static analysis of programs. Technical Report CS-2000-12, University of Virginia (2000)
12. Stark, S.: Why lawyers can’t write. Harvard Law Review 97(1389) (1984)
13. Tyma, P.: Method for renaming identifiers of a computer program. US patent 6,102,966 (2000)
14. Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for data hiding. IBM Syst. J. 35(3-4), 313–336 (1996)
15. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal, 656–715 (1949)
16. Wayner, P.: Mimic functions. CRYPTOLOGIA 14(3) (1992)
17. Monden, A., Iida, H., Matsumoto, K., Inoue, K., Torii, K.: A practical method for watermarking Java programs. In: 24th Computer Software and Applications Conference (2000)

18. Myles, G., Collberg, C.: k-gram based software birthmarks. In: Proceedings of SAC (2005)
19. Horne, B., Matheson, L., Sheehan, C., Tarjan, R.E.: Dynamic self-checking techniques for improved tamper resistance. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, Springer, Heidelberg (2002)
20. Chang, H., Atallah, M.: Protecting software code by guards. In: Sander, T. (ed.) DRM 2001. LNCS, vol. 2320, Springer, Heidelberg (2002)
21. Gang Tan, Y.C., Jakubowski, M.H.: Delayed and controlled failures in tamper-resistant systems. In: Information Hiding 2006 (2006)
22. Farrell, N.: Mac Display Eater kills home files. The Inquirer (February 27, 2007)
23. Keaney, T.A., Cohen, E.A.: Gulf War Air Power Survey Summary Report (1993)
24. Aucsmith, D.: Tamper resistant software: An implementation. In: Anderson, R. (ed.) Information Hiding. LNCS, vol. 1174, pp. 317–333. Springer, Heidelberg (1996)
25. Wang, F.Y.: Computational experiments for behavior analysis and decision evaluation of complex systems. Journal of Systems Simulations 16(5), 893–897 (2004)

Characterizing Software Self-healing Systems

Angelos D. Keromytis

Department of Computer Science, Columbia University
1214 Amsterdam Ave. New York, NY 10027, USA
angelos@cs.columbia.edu

Abstract. The introduction of self-healing capabilities to software systems could offer a way to alter the current, unfavorable imbalance in the software security arms race. Consequently, self-healing software systems have emerged as a research area of particular interest in recent years. Motivated by the inability of traditional techniques to guarantee software integrity and availability, especially against motivated human adversaries, self-healing approaches are meant to complement existing approaches to security.

In this paper, we provide a first attempt to characterize self-healing software systems by surveying some of the existing work in the field. We focus on systems that effect structural changes to the software under protection, as opposed to block-level system reconfiguration. Our goal is to begin mapping the space of software self-healing capabilities. We believe this to be a necessary first step in exploring the boundaries of the research space and understanding the possibilities that such systems enable, as well as determining the risks and limitations inherent in automatic-reaction schemes.

Keywords: Self-healing, reliability, availability, software security.

1 Introduction

Self-healing capabilities have begun to emerge as an exciting and potentially valuable weapon in the software security arms race. Although much of the work to date has remained confined to the realm of academic publications and prototype-building, we believe that it is only a matter of time before deployed systems begin to incorporate elements of automated reaction and self healing. Consequently, we believe it is important to understand what self-healing systems are, why they evolved in the first place, and how they operate. Given the infancy of the area as a research focus, we only have a relatively small and, given the potential application scope of self-healing techniques, highly diverse sample set to draw upon in defining this space. Despite this, we believe that some high-level common characteristics of self-healing software systems can be discerned from the existing work.

What, exactly, is a self-healing system? For the purposes of our discussion, a self-healing software system is a software architecture that enables the continuous and automatic monitoring, diagnosis, and remediation of software faults. Generally, such an architecture is composed of two high-level elements: the software service whose integrity and availability we are interested in improving, and the elements of the system that perform the monitoring, diagnosis and healing. The self-healing components can be viewed as a form of middleware — although, in some systems, such a separation is difficult to delineate.

Self-healing systems evolved primarily as a result of the failure of other techniques, whether in isolation or combination, to provide an adequate solution to the problem of software reliability. More specifically, self-healing techniques try to strike a balance between reliability, assurance, and performance (with performance generally in an inverse relationship to the first two). An important difference between self-healing and the traditional fault-tolerant architectures and techniques is that the former try to identify and eliminate (or at least mitigate) the root cause of the fault, while the latter generally only bring the system to a state from which it can resume execution. Thus, fault-tolerant systems can be viewed as primarily geared against rarely occurring failures.

The diversity of techniques for effecting self-healing reflects both the relative immaturity of the field and the large scope of failure and threat models such systems must cope with. Most work to date has focused on a narrow class of faults and/or the refinement of a specific detection or mitigation technique. Although comprehensive frameworks have been proposed [1, 2], none has been fully implemented and demonstrated to date. Although we expect this picture to remain the same for the foreseeable future, more comprehensive techniques and systems will emerge as we achieve a better understanding of the capabilities and limitations of existing proposed approaches.

Finally, we wish to distinguish between block-level system reconfiguration-based healing, and lower-level structural modification-based healing techniques. The former treat software as a black box with some configurable parameters, and focus on rearranging the way the system components interact with each other. The latter depend on specific, "low-level" techniques to detect, mitigate and mask/correct defects in the software. As a starting point, we focus on the latter approach, both because of our familiarity with this space and because of the use of such structural-modification techniques as the building elements for system-reconfiguration approaches. To the extent that software becomes componentized, we believe that these two high-level approaches are likely to converge in terms of tools and techniques used.

In the remainder of this paper, we will expand on the main three questions we posed at the beginning of this section: what are self-healing systems, why have they emerged, and how they operate, using specific examples from the literature.

2 Self-healing Systems: What

At a high level, self-healing software techniques are modeled after the concept of an Observe Orient Decide Act (OODA) feedback loop, as shown in Figure 1.

The high-level intuition is that, if proactive or runtime protection mechanisms are too expensive to use in a blanket manner, they should instead be used in a targeted manner. Identifying where and how to apply protection is done by observing the behavior of the system in a non-invasive manner. The goal of this monitoring is to detect the occurrence of a fault and determine its parameters, *e.g.,* the type of fault, the input or sequence of events that led to the it, the approximate region of code where the fault manifests itself, and any other information that may be useful in creating fixes.

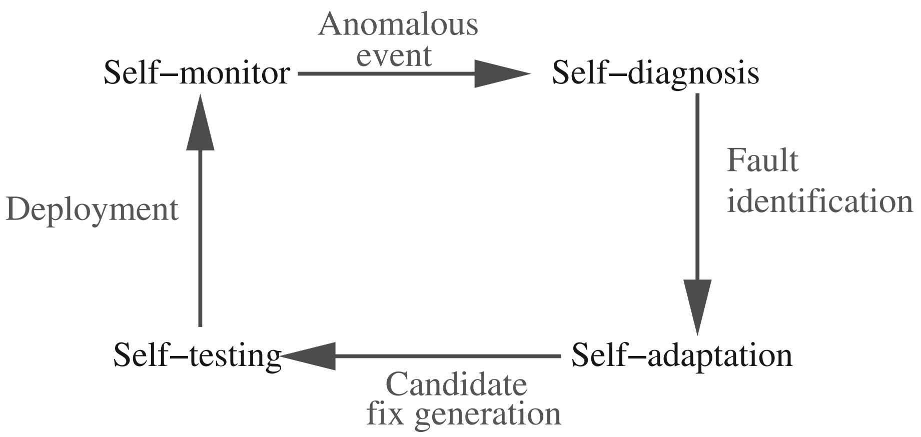


Fig. 1. General architecture of a self-healing system. The system monitors itself for indications of anomalous behavior. When such behavior is detected, the system enters a self-diagnosis mode that aims to identify the fault and extract as much information as possible with respect to its cause, symptoms, and impact on the system. Once these are identified, the system tries to adapt itself by generating candidate fixes, which are tested to find the best target state.

Following identification, the system will need to create one or more possible fixes tailored to the particular instance of the fault. The nature of these fixes depends on types of faults and the available protection mechanisms. Potential fixes to software faults include snapshot-rollback, input filtering, increased monitoring or isolation for the vulnerable process, selective application of any runtime protection mechanism, and others.

Each candidate fix produced by the system may then be tested to verify its efficacy and impact on the application (*e.g.,* in terms of side effects or performance degradation). This testing can take several forms, including running pre-defined test-suites, replaying previously seen traffic (including the input that triggered the fault), *etc.* If an acceptable fix is produced, the system is updated accordingly. This can be done through established patch-management and configuration-management mechanisms, or any other suitable mechanism.

Note that different fixes, or fixes of different accuracy/performance levels, may be successively applied as the system spends more time analyzing a fault. For example, the initial reaction to a failure may be a firewall reconfiguration. After further analysis, the system may produce a software patch or a content filter that blocks the specific input that caused the fault. Finally, the system may then replace the specific content filter with a generalized signature, obtained through signature generalization [3], dynamic analysis of the targeted software [4], or other means.

3 Self-healing Systems: Why

Despite considerable work in fault tolerance and reliability, software remains notoriously buggy and crash-prone. The current approach to ensuring the security and availability of software consists of a mix of different techniques:

- **Proactive techniques** seek to make the code as dependable as possible, through a combination of safe languages (*e.g.,* Java [5]), libraries [6] and compilers [7, 8], code analysis tools and formal methods [9,10,11], and development methodologies.
- **Debugging techniques** aim to make post-fault analysis and recovery as easy as possible for the programmer that is responsible for producing a fix.
- **Runtime protection techniques** try to detect the fault using some type of fault isolation such as StackGuard [12] and FormatGuard [13], which address specific types of faults or security vulnerabilities.
- **Containment techniques** seek to minimize the scope of a successful exploit by isolating the process from the rest of the system, *e.g.,* through use of virtual machine monitors such as VMWare or Xen, system call sandboxes such as Systrace [14], or operating system constructs such as Unix *chroot()*, FreeBSD's *jail* facility, and others [15, 16].
- **Byzantine fault-tolerance and quorum techniques** rely on redundancy and diversity to create reliable systems out of unreliable components [17, 1, 18].

These approaches offer a poor tradeoff between assurance, reliability in the face of faults, and performance impact of protection mechanisms. In particular, software availability has emerged as a concern of equal importance as integrity.

The need for techniques that address the issue of recovering execution in the presence of faults is reflected by recent emergence of a few novel research ideas [19, 20]. For example, *error virtualization* [19, 21] operates under the assumption that there exists a mapping between the set of errors that *could* occur during a program's execution (*e.g.,* a caught buffer overflow attack, or an illegal memory reference exception) and the limited set of errors that are explicitly handled by the program's code. Thus, a failure that would cause the program to crash is translated into a "return with an error code" from the function in which the fault occurred (or from one of its ancestors in the stack). These techniques, despite their novelty in dealing with this pressing issue, have met much controversy, primarily due to the lack of guarantees, in terms of altering program semantics, that can be provided. Masking the occurrence of faults will always carry this stigma since it forces programs down unexpected execution paths. However, we believe that the basic premise of masking failures to permit continued program execution is promising.

In general, we believe that a new class of **reactive** protection mechanisms need to be added to the above list. Some techniques that can be classified as reactive include Intrusion Prevention Systems (IPS) and automatically generated content-signature blockers, *e.g.,* [22]. Most such systems have focused on network-based prevention, augmenting the functionality of firewalls. However, a number of trends make the use of such packet inspection technologies unlikely to work well in the future:

- Due to the increasing line speeds and the more computation-intensive protocols that a firewall must support (such as IPsec), firewalls tend to become congestion points. This gap between processing and networking speeds is likely to increase, at least for the foreseeable future; while computers (and hence firewalls) are getting faster, the combination of more complex protocols and the tremendous increase in the amount of data that must be passed through the firewall has been and likely will continue to out-pace Moore's Law [23].

– The complexity of existing and future protocols makes packet inspection an expensive proposition, especially in the context of increasing line speeds. Furthermore, a number of protocols are inherently difficult to process in the network because of lack of knowledge that is readily available at the endpoints (*etc.* FTP and RealAudio port numbers).
– End-to-end encryption, especially of the on-demand, opportunistic type effectively prevents inspection-based systems from looking inside packets, or even at packet headers.
– Finally, worms and other malware have started using polymorphism or metamorphism [24] as cloaking techniques. The effect of these is to increase the analysis requirements, in terms of processing cycles, beyond the budget available to routers or firewalls.

All these factors argue for host-based reactive protection mechanisms.

4 Self-healing Systems: How

Most defense mechanisms usually respond to an attack by terminating the attacked process. Even though it is considered "safe", this approach is unappealing because it leaves systems susceptible to the original fault upon restart and risks losing accumulated state.

Self-healing mechanisms complement approaches that stop attacks from succeeding by preventing the injection of code, transfer of control to injected code, or misuse of existing code. Approaches to automatically defending software systems have typically focused on ways to proactively or at runtime protect an application from attack. Examples of these proactive approaches include writing the system in a "safe" language, linking the system with "safe" libraries [6], transforming the program with artificial diversity, or compiling the program with stack integrity checking [12]. Some defense systems also externalize their response by generating either vulnerability [4, 25, 26] or exploit [27, 28, 29, 30, 3] signatures to prevent malicious input from reaching the protected system.

Starting with the technique of *program shepherding* [31], the idea of enforcing the integrity of control flow has been increasingly researched. Program shepherding validates branch instructions to prevent transfer of control to injected code and to make sure that calls into native libraries originate from valid sources. Control flow is often corrupted because input is eventually incorporated into part of an instruction's opcode, set as a jump target, or forms part of an argument to a sensitive system call. Recent work focuses on ways to prevent these attacks using tainted dataflow analysis [32,22,4]. Abadi *et al.* [33] propose formalizing the concept of Control Flow Integrity (CFI), observing that high-level programming often assumes properties of control flow that are not enforced at the machine level. CFI provides a way to statically verify that execution proceeds within a given control-flow graph (the CFG effectively serves as a policy). The use of CFI enables the efficient implementation of a software shadow call stack with strong protection guarantees. However, such techniques generally focus on integrity protection at the expense of availability.

The acceptability envelope, a region of imperfect but acceptable software systems that surround a perfect system, as introduced by Rinard [34] promotes the idea that current software development efforts might be misdirected. Rinard explains that certain regions of a program can be neglected without adversely affecting the overall availability of the system. To support these claims, a number of case studies are presented where introducing faults such as an off-by-one error does not produce unacceptable behavior. This work supports the claim that most complex systems contain the necessary framework to propagate faults gracefully and the error toleration allowed (or exploited) by some self-healing systems expands the acceptability envelope of a given application.

4.1 Self-healing Techniques

Some first efforts at providing effective remediation strategies include failure-oblivious computing [20], error virtualization [19,21], rollback of memory updates [29,35], crash-only software [36], and data-structure repair [37]. The first two approaches may cause a semantically incorrect continuation of execution (although the Rx system [38] attempts to address this difficulty by exploring semantically safe alterations of the program's environment).

TLS. Oplinger and Lam [35] employ hardware Thread-Level Speculation to improve software reliability. They execute an application's monitoring code in parallel with the primary computation and roll back the computation "transaction" depending on the results of the monitoring code.

Failure-Oblivious Computing. Rinard *et al.* [39] developed a compiler that inserts code to deal with writes to unallocated memory by virtually expanding the target buffer. Such a capability aims to provide a more robust fault response rather than simply crashing. The technique presented by Rinard *et al.* [39] is subsequently introduced in a modified form as *failure-oblivious computing* [20]. Because the program code is extensively re-written to include the necessary checks for *every* memory access, the system incurs overheads ranging from 80% up to 500% for a variety of different applications. Failure-oblivious computing specifically targets memory errors.

Data-structure Repair. One of the most critical concerns with recovering from software faults and vulnerability exploits is ensuring the consistency and correctness of program data and state. An important contribution in this area is that of Demsky and Rinard [37], which discusses mechanisms for detecting corrupted data structures and fixing them to match some pre-specified constraints. While the precision of the fixes with respect to the semantics of the program is not guaranteed, their test cases continued to operate when faults were randomly injected. Similar results are shown by Wang *et al.* [40]: when program execution is forced to take the "wrong" path at a branch instruction, program behavior remains the same in over half the times.

Rx. In Rx [38], applications are periodically checkpointed and continuously monitored for faults. When an error occurs, the process state is rolled back and replayed in a new "environment". If the changes in the environment do not cause the bug to manifest, the program will have survived that specific software failure. However, previous work [41, 42] found that over 86% of application faults are independent of the operating environment and entirely deterministic and repeatable, and that recovery is likely to be successful only through application-specific (or application-aware) techniques.

Error Virtualization. Error virtualization [19, 21] assumes that portions of an application can be treated as a transaction. Functions serve as a convenient abstraction and fit the transaction role well in most situations [19]. Each transaction (vulnerable code slice) can be speculatively executed in a sandbox environment. In much the same way that a processor speculatively executes past a branch instruction and discards the mispredicted code path, error virtualization executes the transaction's instruction stream, optimistically "speculating" that the results of these computations are benign. If this *microspeculation* succeeds, then the computation simply carries on. If the transaction experiences a fault or exploited vulnerability, then the results are ignored or replaced according to the particular response strategy being employed. The key assumption underlying error virtualization is that a mapping can be created between the set of errors that *could* occur during a program's execution and the limited set of errors that the program code explicitly handles. By virtualizing errors, an application can continue execution through a fault or exploited vulnerability by nullifying its effects and using a manufactured return value for the function where the fault occurred.

Modeling executing software as a transaction that can be aborted has been examined in the context of language-based runtime systems (specifically, Java) [43,44]. That work focused on safely terminating misbehaving threads, introducing the concept of "soft termination". Soft termination allows threads to be terminated while preserving the stability of the language runtime, without imposing unreasonable performance overheads. In that approach, threads (or *codelets*) are each executed in their own transaction, applying standard ACID semantics. This allows changes to the runtime's (and other threads') state made by the terminated codelet to be rolled back. The performance overhead of that system can range from 200% up to 2,300%.

One approach that can be employed by error virtualization techniques is the one described by Locasto *et al.* [45], where function-level profiles are constructed during a training phase that can, in turn, be used to predict function return values. While this technique is useful for predicting appropriate return values, especially in the absence of return type information, it suffers from the same problems as error virtualization, *i.e.,* it is oblivious to errors.

ASSURE. ASSURE [46] is an attempt to minimize the likelihood of a semantically incorrect response to a fault or attack. ASSURE proposes the notion of *error virtualization rescue points*. A rescue point is a program location that is known to successfully propagate errors and recover execution. The insight is that a program will respond to malformed input differently than legal input; locations in the code that successfully handle these sorts of anticipated input "faults" are good candidates for recovering to a safe execution flow. ASSURE can be understood as a type of exception handling that dynamically identifies the best scope to handle an error.

DIRA. DIRA [29] is a technique for automatic detection, identification and repair of control-hijacking attacks. This solution is implemented as a GCC compiler extension that transforms a program's source code adding heavy instrumentation so that the resulting program can perform these tasks. The use of checkpoints throughout the program ensures that corruption of state can be detected if control sensitive data structures are overwritten. Unfortunately, the performance implications of the system make it unusable as a front-line defense mechanism.

Vigilante. Vigilante [4] is a system motivated by the need to contain rapid malcode. Vigilante supplies a mechanism to detect an exploited vulnerability (by analyzing the control flow path taken by executing injected code) and defines a data structure (Self-Certifying Alert) for exchanging information about this discovery. A major advantage of this vulnerability-specific approach is that Vigilante is exploit-agnostic and can be used to defend against polymorphic worms.

A problem that is inherent with all techniques that try to be oblivious to the fact that an error has occurred is the ability to guarantee session semantics. Altering the functionality of the memory manager often leads to the uncovering of latent bugs in the code [47].

5 Self-healing Systems: Future Directions

Given the embryonic state of the research in self-healing software systems, it should come as no surprise that there are significant gaps in our knowledge and understanding of such systems' capabilities and limitations. In other words, this is an extremely fertile area for further research. Rather than describe in detail specific research topics, we outline three general research thrusts: fault detection, fault recovery/mitigation, and assurance.

Fault Detection. One of the constraining factors on the effectiveness of self-healing systems is their ability to detect faults. Thus, ways to improve fault detection at lower memory and computation cost will always be of importance. One interesting direction of research in this topic concerns the use of hardware features to improve fault detection and mitigation [48]. Another interesting area of research is the use of collections of nodes that collaborate in the detection of attacks and faults by exchanging profiling or fault-occurrence information [49]. Although such an approach can leverage the size and inherent usage-diversity of popular software monocultures, it also raises significant practical issues, not the least of which is data privacy among the collaborating nodes.

We also believe that the next generation of self-healing defense mechanisms will require a much more detailed dynamic analysis of application behavior than is currently done, possibly combined with *a priori* behavior profiling and code analysis techniques. This will be necessary to detect application-specific semantic faults, as opposed to the "obvious" faults, such as application crashes or control-hijack attacks, with which existing systems have concerned themselves to date. Profiling an application can allow self-healing systems to "learn" common behavior [45]. The complementary approach is to use application-specific integrity policies, which specify acceptable values for (aspects of) the application's internal runtime state, to detect attacks and anomalies [50].

Fault Recovery/Mitigation. To date, most systems have depended on snapshot/recovery, often combined with input filtering. The three notable exceptions are error virtualization, failure-oblivious computing, and data-structure repair. The former two techniques can be characterized as "fault masking", while the last uses learning to correct possibly corrupt data values.

The technical success of self-healing systems will depend, to a large extent, on their ability to effectively mitigate or recover from detected faults, while ensuring system availability. Thus, research on additional fault-recovery/masking/mitigation techniques is of key importance, especially when dealing with faults at different (higher) semantic levels. Similar to fault detection, two high-level approaches seem promising: profiling applications to identify likely "correct" ways for fault recovery [46], and using application-specific recovery policies that identify steps that the system must undertake to recover from different types of faults [50].

Assurance. Finally, to gain acceptance in the real world, self-healing systems and techniques must provide reasonable assurances that they will not cause damage in the course of healing an application, and that they cannot be exploited by an adversary to attack an otherwise secure system. This is perhaps the biggest challenge faced by automated defense systems in general. Self-healing software systems of the type we have been discussing in this paper may need to meet an even higher standard of assurance, given the intrusiveness and elevated privileges they require to operate. Although to a large extent acceptance is dictated by market perceptions and factors largely outside the technical realm, there are certain measures that can make self-healing systems more palatable to system managers and operators. In particular, transparency of operation, the ability to operate under close human supervision (at the expense of reaction speed), "undo" functionality, and comprehensive fix-testing and reporting capabilities all seem necessary elements for a successful system. Although some of these elements primarily involve system engineering, there remain several interesting and unsolved research problems, especially in the area of testing.

6 Conclusions

We have outlined some of the recent and current work on self-healing software systems, describing the origins and design philosophy of such systems. We believe that self-healing systems will prove increasingly important in countering software-based attacks, assuming that the numerous research challenges (and opportunities), some of which we have outlined in this paper, can be overcome.

Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-06-2-0221 and by NSF Grant 06-27473. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

1. Reynolds, J.C., Just, J., Clough, L., Maglich, R.: On-Line Intrusion Detection and Attack Prevention Using Diversity, Genrate-and-Test, and Generalization. In: Proceedings of the 36^{th} Hawaii International Conference on System Sciences (HICSS) (2003)
2. Keromytis, A.D., Parekh, J., Gross, P.N., Kaiser, G., Misra, V., Nieh, J., Rubenstein, D., Stolfo, S.: A Holistic Approach to Service Survivability. In: Proceedings of the ACM Survivable and Self-Regenerative Systems Workshop, ACM Press, New York (2003)
3. Wang, X., Li, Z., Xu, J., Reiter, M.K., Kil, C., Choi, J.Y.: Packet Vaccine: Black-box Exploit Detection and Signature Generation. In: Proceedings of the 13^{th} ACM Conference on Computer and Communications Security (CCS), pp. 37–46. ACM Press, New York (2006)
4. Costa, M., Crowcroft, J., Castro, M., Rowstron, A.: Vigilante: End-to-End Containment of Internet Worms. In: Proceedings of the Symposium on Systems and Operating Systems Principles (SOSP) (2005)
5. Gosling, J., Joy, B., Steele, G.: The Java Language Specification. Addison Wesley, Reading (1996)
6. Baratloo, A., Singh, N., Tsai, T.: Transparent Run-Time Defense Against Stack Smashing Attacks. In: Proceedings of the USENIX Annual Technical Conference (2000)
7. Jim, T., Morrisett, G., Grossman, D., Hicks, M., Cheney, J., Wang, Y.: Cyclone: A safe dialect of C. In: Proceedings of the USENIX Annual Technical Conference, pp. 275–288 (2002)
8. Necula, G.C., McPeak, S., Weimer, W.: CCured: Type-Safe Retrofitting of Legacy Code. In: Proceedings of the Principles of Programming Languages (PoPL) (2002)
9. Chen, H., Wagner, D.: MOPS: an Infrastructure for Examining Security Properties of Software. In: Proceedings of the ACM Computer and Communications Security (CCS) Conference, pp. 235–244. ACM Press, New York (2002)
10. Ganapathy, V., Jha, S., Chandler, D., Melski, D., Vitek, D.: Buffer Overrun Detection using Linear Programming and Static Analysis. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), pp. 345–364. ACM Press, New York (2003)
11. Yang, J., Kremenek, T., Xie, Y., Engler, D.: MECA: an Extensible, Expressive System and Language for Statically Checking Security Properties. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), pp. 321–334. ACM Press, New York (2003)
12. Cowan, C., Pu, C., Maier, D., Hinton, H., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q.: Stackguard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks. In: Proceedings of the USENIX Security Symposium (1998)
13. Cowan, C., Barringer, M., Beattie, S., Kroah-Hartman, G.: Formatguard: Automatic protection from printf format string vulnerabilities. In: Proceedings of the 10th USENIX Security Symposium (2001)
14. Provos, N.: Improving Host Security with System Call Policies. In: Proceedings of the 12^{th} USENIX Security Symposium, pp. 257–272 (2003)
15. Watson, R.N.M.: TrustedBSD: Adding Trusted Operating System Features to FreeBSD. In: Proceedings of the USENIX Annual Technical Conference, Freenix Track, pp. 15–28 (2001)
16. Loscocco, P., Smalley, S.: Integrating Flexible Support for Security Policies into the Linux Operating System. In: Proceedings of the USENIX Annual Technical Conference, Freenix Track, pp. 29–40 (2001)
17. Yin, J., Martin, J.P., Venkataramani, A., Alvisi, L., Dahlin, M.: Separating Agreement from Execution for Byzantine Fault Tolerant Services. In: Proceedings of the ACM Symposium on Operating Systems Principles (SOSP), ACM Press, New York (2003)
18. Cox, B., Evans, D., Filipi, A., Rowanhill, J., Hu, W., Davidson, J., Knight, J., Nguyen-Tuong, A., Hiser, J.: N-Variant Systems: A Secretless Framework for Security through Diversity. In: Proceedings of the 15^{th} USENIX Security Symposium, pp. 105–120 (2005)

19. Sidiroglou, S., Locasto, M.E., Boyd, S.W., Keromytis, A.D.: Building a Reactive Immune System for Software Services. In: Proceedings of the USENIX Annual Technical Conference, pp. 149–161 (2005)
20. Rinard, M., Cadar, C., Dumitran, D., Roy, D., Leu, T., Beebee, W.J.: Enhancing Server Availability and Security Through Failure-Oblivious Computing. In: Proceedings of OSDI (2004)
21. Sidiroglou, S., Giovanidis, Y., Keromytis, A.: A Dynamic Mechanism for Recovery from Buffer Overflow attacks. In: Proceedings of the 8th Information Security Conference (ISC) (2005)
22. Newsome, J., Song, D.: Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In: The 12th Annual Network and Distributed System Security Symposium (2005)
23. Dahlin, M.: Serverless Network File Systems. PhD thesis, UC Berkeley (1995)
24. Ször, P., Ferrie, P.: Hunting for Metamorphic. Technical report, Symantec Corporation (2003)
25. Newsome, J., Brumley, D., Song, D.: Vulnerability–Specific Execution Filtering for Exploit Prevention on Commodity Software. In: Proceedings of the 13^{th} Symposium on Network and Distributed System Security (NDSS 2006) (2006)
26. Cui, W., Peinado, M., Wang, H.J., Locasto, M.E.: ShieldGen: Automated Data Patch Generation for Unknown Vulnerabilities with Informed Probing. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (2007)
27. Liang, Z., Sekar, R.: Fast and Automated Generation of Attack Signatures: A Basis for Building Self-Protecting Servers. In: Proceedings of the 12^{th} ACM Conference on Computer and Communications Security (CCS), ACM Press, New York (2005)
28. Locasto, M.E., Wang, K., Keromytis, A.D., Stolfo, S.J.: FLIPS: Hybrid Adaptive Intrusion Prevention. In: Proceedings of the 8^{th} International Symposium on Recent Advances in Intrusion Detection (RAID), pp. 82–101 (2005)
29. Smirnov, A., Chiueh, T.: DIRA: Automatic Detection, Identification, and Repair of Control-Hijacking Attacks. In: Proceedings of the 12^{th} ISOC Symposium on Network and Distributed System Security (SNDSS) (2005)
30. Xu, J., Ning, P., Kil, C., Zhai, Y., Bookholt, C.: Automatic Diagnosis and Response to Memory Corruption Vulnerabilities. In: Proceedings of the 12^{th} ACM Conference on Computer and Communications Security (CCS), ACM Press, New York (2005)
31. Kiriansky, V., Bruening, D., Amarasinghe, S.: Secure Execution Via Program Shepherding. In: Proceedings of the 11^{th} USENIX Security Symposium (2002)
32. Suh, G.E., Lee, J.W., Zhang, D., Devadas, S.: Secure Program Execution via Dynamic Information Flow Tracking. In: Proceedings of the 11^{th} International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-XI) (2004)
33. Abadi, M., Budiu, M., Erlingsson, U., Ligatti, J.: Control-Flow Integrity: Principles, Implementations, and Applications. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), ACM Press, New York (2005)
34. Rinard, M.: Acceptability-oriented Computing. In: Proceedings of ACM OOPSLA, ACM Press, New York (2003)
35. Oplinger, J., Lam, M.S.: Enhancing Software Reliability with Speculative Threads. In: Proceedings of the 10^{th} International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X) (2002)
36. Candea, G., Fox, A.: Crash-Only Software. In: Proceedings of the 9^{th} Workshop on Hot Topics in Operating Systems (HOTOS-IX) (2003)
37. Demsky, B., Rinard, M.C.: Automatic Detection and Repair of Errors in Data Structures. In: Proceedings of ACM OOPSLA, ACM Press, New York (2003)
38. Qin, F., Tucek, J., Sundaresan, J., Zhou, Y.: Rx: treating bugs as allergies - a safe method to survive software failures. In: Herbert, A., Birman, K.P. (eds.) Proceedings of ACM SOSP, pp. 235–248. ACM Press, New York (2005)

39. Rinard, M., Cadar, C., Dumitran, D., Roy, D., Leu, T.: A Dynamic Technique for Eliminating Buffer Overflow Vulnerabilities (and Other Memory Errors). In: Proceedings of ACSAC (2004)
40. Wang, N., Fertig, M., Patel, S.: Y-Branches: When You Come to a Fork in the Road, Take It. In: Proceedings of the 12^{th} International Conference on Parallel Architectures and Compilation Techniques (2003)
41. Chandra, S., Chen, P.M.: Wither Generic Recovery from Application Faults? A Fault Study using Open-Source Software. In: Proceedings of DSN/FTCS (2000)
42. Chandra, S.: An Evaluation of the Recovery-related Properties of Software Faults. PhD thesis, University of Michigan (2000)
43. Rudys, A., Wallach, D.S.: Termination in Language-based Systems. ACM Transactions on Information and System Security 5 (2002)
44. Rudys, A., Wallach, D.S.: Transactional Rollback for Language-Based Systems. In: ISOC Symposium on Network and Distributed Systems Security (SNDSS) (2001)
45. Locasto, M.E., Stavrou, A., Cretu, G.F., Keromytis, A.D., Stolfo, S.J.: Quantifying Application Behavior Space for Detection and Self-Healing. Technical Report CUCS-017-06, Columbia University Computer Science Department (2006)
46. Sidiroglou, S., Laadan, O., Keromytis, A.D., Nieh, J.: Using Rescue Points to Navigate Software Recovery (Short Paper). In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (2007)
47. Brooks, F.P.: Mythical Man-Month, 1st edn. Addison-Wesley, Reading (1975)
48. Locasto, M.E., Sidiroglou, S., Keromytis, A.D.: Speculative Virtual Verification: Policy-Constrained Speculative Execution. In: Proceedings of the New Security Paradigms Workshop (NSPW), pp. 170–175 (2005)
49. Locasto, M., Sidiroglou, S., Keromytis, A.: Software Self-Healing Using Collaborative Application Communities. In: Proceedings of the Internet Society (ISOC) Symposium on Network and Distributed Systems Security (SNDSS) (2006)
50. Locasto, M.E., Stavrou, A., Cretu, G.F., Keromytis, A.D.: From STEM to SEAD: Speculative Execution for Automated Defense. In: Proceedings of the USENIX Annual Technical Conference (2007)

Assumptions: The Trojan Horses of Secure Protocols

Paulo Verissimo

Univ. Lisboa
pjv@di.fc.ul.pt

Abstract. Secure protocols rely on a number of assumptions about the environment which, once made, free the designer from thinking about the complexity of what surrounds the execution context.

Henceforth, the designer forgets about the environment and moves on proving her algorithm correct, given the assumptions. When assumptions do not represent with sufficient accuracy the environment they are supposed to depict, they may become the door to successful attacks on an otherwise mathematically correct algorithm. Moreover, this can happen as unwitting to systems as a Trojan Horse's action.

We wish to discuss the theoretical underpinnings of those problems and evaluate some recent research results that demonstrate a few of those limitations in actual secure protocols.

1 Introduction

Secure protocols rely on a number of assumptions about the environment which, once made, free the designer from thinking about the complexity of what surrounds the execution context. Henceforth, the designer forgets about the environment and moves on proving her algorithm correct, given the assumptions. When assumptions do not represent with sufficient accuracy the environment they are supposed to depict, they may become the door to successful attacks on an otherwise mathematically correct algorithm. Moreover, this can happen as unwitting to systems as a Trojan Horse's action.

Intrusion Tolerance has become a reference paradigm for dealing with faults and intrusions, achieving security (and dependability) in an automatic way. However, there are issues specific to the severity of malicious faults (attacks and intrusions) that made the problems and limitations introduced above very visible and, what is more, very plausible. Intrusion-tolerant protocols that deal with intrusions much along the lines of classical fault tolerance, like for example Byzantine agreement, atomic broadcast, state machine replication, or threshold secret sharing, have become a reference in this field.

Using them as example, we wish to discuss the theoretical underpinnings of those problems and evaluate some recent research results that demonstrate a few of those limitations in actual secure protocols.

2 Classical Distributed Algorithms Design

The design of distributed algorithms follows a well-determined path and fault/intrusion-tolerant (FIT) algorithms are no exception. The basic proposition underlying the design of FIT algorithms is, informally:

FIT - Given n processes and f a function of n, and a set H of assumptions on the environment, then for at least $n - f$ correct processes, algorithm $\mathcal{A}$ satisfies a pre-defined set of properties $\mathcal{P}$, i.e. executes correctly.

Classical distributed algorithms design has focused its attention on *"algorithm A satisfies a pre-defined set of properties $\mathcal{P}$, i.e. executes correctly"*, consider it the mathematics viewpoint: assumptions are accepted as a given, and it is proved that the algorithm satisfies the properties.

There is nothing wrong with this in principle, but how about looking critically at other parts of the proposition? For example, *"a set H of assumptions on the environment"*. In fact, the usual path is to start with the weakest possible assumptions, normally, in distributed systems security, one talks about arbitrary failure modes, over asynchronous models. The unfortunate fact is that such weak assumptions restrain the amount of useful things that can be done. Namely:

– algorithms are safe, but normally inefficient in time and message complexity;
– deterministic solutions of hard problems such as consensus, Byzantine Agreement or State Machine Replication with Atomic Broadcast are impossible, a result known as FLP impossibility [1];
– furthermore, timed problems (e.g., e-commerce, online stock-exchange, web applications with SLAs, SCADA, etc.) are by definition impossible in a time-free world.

If efficient/performant FIT algorithms are sought, one has to assume controlled failure modes (omissive, fail-silent, etc.). Moreover, for solving the above problems of determinism or for building any timely services (even soft real-time), one must relax the asynchrony assumption and assume at least partially synchronous models. However, this brings a problem: these algorithms will only work to the coverage of those assumptions. Unfortunately, relaxing these assumptions amounts to creating attack space for the hacker, unless something is done to substantiate them:

– controlled failures are hard to enforce in the presence of malicious faults;
– partial synchrony is susceptible to attacks on the timing assumptions;
– even in benign but open settings (e.g., Internet), synchrony is difficult to implement.

As such, there is a significant body of research continuing to assume arbitrary failure modes over asynchronous models, instead turning itself to weakening the semantics of what can be achieved in those conditions.

Some results look very promising and useful if one is to solve non-timed problems with the highest possible coverage:

– toning down determinism, for example, through randomised versions of the above-mentioned algorithms, e.g. consensus;
– tone down liveness expectations, for example, through indulgence, which means that the algorithms may not be guaranteed to terminate, but they will always keep safety;
– use other (sometimes weaker) semantics which do not fall under the FLP impossibility result, for example, through reliable broadcast, secret sharing, quorums, etc.

Another alternative is toning down the allowed fault or intrusion severity, for example, through hybrid fault models [2], which imply that the quorum of f faults that is accepted from a set of processes is divided amongst different fault types, for example Byzantine and crash, $f = f_B + f_c$.

When even the simplest timing is needed, the system can no longer be time-free, for example if a periodical operation or the timely trigger of an execution are needed. In that case, one has to tone down asynchrony, for example, through time-free or timed partially synchronous algorithms [3,4]. Some solutions to circumvent FLP, even non-timed, rely on a form of partial synchrony as well, eventual synchrony, such as the failure detectors [5]. However, these only fare well under benign failure modes.

Unlike the former, these two alternatives pull the boundary of arbitrary failures and of asynchrony, respectively, a bit back, with the corresponding risks in coverage.

3 Assumptions as Vulnerabilities

It is usually said that one should make the weakest assumptions possible. Let us question why. For example, people assume that large-scale (i.e., Internet) systems are asynchronous not for the sake of it, but just because it is hard to substantiate that they behave synchronously. So, confidence (coverage) on the former assumption is higher than on the latter one. Likewise with assuming Byzantine vs. benign behaviour, if the system is open or not very well known. In other words, the asynchrony/Byzantine assumptions would lead to safer designs in this case, though probably not the most effective and efficient.

"Every assumption is a vulnerability"

is a frequently heard quote, which of course leads us right onto the above-mentioned path of *arbitrary failure modes over asynchronous models.* Why? Because it looks like we are not making any assumption: we do not assume anything about the behaviour of processes; we do not assume anything about time.

The caveat is that asynchrony/Byzantine yield so weak a model that it prevents us from doing some important things, as shown earlier. When problems offer significant hardness, algorithm designers often insert some synchrony in the underlying model, or relax the arbitrary behaviour just a bit, trying to circumvent these difficulties.

According to the quote above, these extra assumptions are vulnerabilities. Furthermore, whilst some are explicit and can deserve attention from the designer, such as the above-mentioned hybrid faults or timed partial synchrony, others are implicit, like assuming that the periodical triggering of a task, a time-out, or the reading of a clock, are meaningful w.r.t. real time in an asynchronous model.

Observation 1 - These subtle feathers of synchrony introduce vulnerabilities which often go undetected, exactly because they are not clearly assumed. In fact, they are bugs in the model, as there are bugs in software: the algorithm designer relies that the system will perform as the assumptions say, but the latter conceal a different behaviour, just as buggy software performs differently than assumed. The consequence is that one may no longer have safe designs with regard to time, despite using asynchronous system models.

4 On Resource Exhaustion

Back to the “pure” Byzantine/asynchronous model, under this no-assumptions model we tend to rely on the fact that we are not making assumptions on time or behaviour, and consider the system extremely resilient. We assume that the system lives long enough to do its job. Can we, really?

In fact, we are still making important assumptions: on the maximum number of allowed faults/intrusions f; on the harshness of the environment or the power of the attacker, respectively for accidental or malicious faults, giving the speed at which the latter are performed; about fault independence or on the expectation that faults/attacks do not occur simultaneously in several nodes or resources. That is, accepting these assumptions as vulnerabilities, those systems, despite following an asynchronous and Byzantine model, are in fact subject to several threats:

- unexpected resource exhaustion (e.g. the number of replicas getting below threshold);
- attacks on the physical time plane (faults and attacks produced at too fast a rate versus the internal pace of the system);
- common-mode faults and attacks (simultaneous attacks against more than one replica).

These problems have been recognized by researchers, who have devised some techniques to keep systems working long enough to fulfil their mission, such as ensuring a large-enough number of replicas at the start and/or using diversity (e.g., different hardware and software, n-version programming, obfuscation) to delay resource exhaustion. However, with static or stationary f-intrusion-tolerant algorithms, even in asynchronous Byzantine environments, it is a matter of time until more than f intrusions occur.

This prompts us for looking critically at other parts of the proposition FIT presented in section 2, like for example *"for at least $n - f$ correct processes"*. In this way, we accept that the proposition is conditional to there being $n - f$ or more correct processes. What if we end-up with less than $n - f$?

Two things may have happened here. That fact may come from an inadequate implementation or design decision, and there is really nothing the algorithm designer can do about it: an adequate algorithm over an inadequate implementation. However, the problem may have a more fundamental cause, something that might be determined at algorithm design time. Were it true, and we would have an inadequate algorithm to start with, and no design or implementation to save it.

How should theory incorporate this fact? For example, by enriching proposition FIT with a safety predicate that would assert or deny something like *"Algorithm $\mathcal{A}$ always terminates execution with at least $n-f$ correct processes."*.

This predicate was called Exhaustion Safety, which informally means that the system maintains the required resources, e.g., the amount of processes or nodes, to guarantee correct performance in the presence of intrusions [6]. As a corollary, an f-intrusion-tolerant distributed system is exhaustion-safe if it terminates before $f + 1$ intrusions being produced. In consequence, a result that would establish at design time that the above predicate would almost never be true or not be guaranteed to be true throughout execution of algorithm $\mathcal{A}$, for whatever real setting, would imply that the algorithm, under the assumed model, would be inadequate to achieve FIT, because it could not be exhaustion-safe.

There has been some research trying to solve the "$n - f$" issue, that is, trying to keep systems working in a perpetual manner or, in other words, achieving exhaustion-safety. The techniques used have been called reactive or proactive recovery (e.g., rejuvenating, refreshing) [7].

Some of these works have assumed an asynchronous model with arbitrary failures in order to make the least assumptions possible. However, given the hardness of the problems to solve, which include for example being able to trigger rejuvenations periodically or reboot a machine within a given delay, these systems end-up making a few assumptions (some of which implicit) that in normal, accidental-fault cases, would have acceptable coverage.

However, in the malicious fault scenario, which they all address, these assumptions will certainly be attacked, giving room for another set of "Trojan-horse"-like pitfalls. What leads to the pitfall is that in normal conditions, clocks and timeouts in asynchronous systems seem to have a real time meaning, but the truth is that a clock is a sequence counter, a timeout does not have a bounded value. System execution and relevant assumptions follow the internal timeline, but attacks can follow the physical timeline. In consequence, there is no formal relation between both, the two timebases are said to be *free-running*. In this case, attacks are produced by hackers in real time (external, physical time), which can in that way delay or stall recovery until more that f intrusions are produced. These problems would have been unveiled if the predicate resource-exhaustion had been used and evaluated at algorithm design time [8].

Observation 2 - The explanation of why systems that are otherwise correct according to the stated assumptions under the asynchronous model used, may fail, is simple. Under attack, the internal timeline can be made to slide w.r.t. the physical timeline at the will of the attacker. For example, for a given speed of an attack along the physical timeline, the internal timeline can be made to progress slowly enough to enable the intrusion. However, since the system is asynchronous, it is completely oblivious to and thus unprotected from, this kind of intrusions: the slow execution could be a legitimate execution. In consequence, harmful as they may be, these intrusions do not even entail a violation of the system's explicit assumptions and resource exhaustion comes unwittingly.

5 On the Substance of Assumptions

Why do the things discussed in the previous sections happen? Let us introduce a postulate:

Postulate 1: Assumptions and models should represent the execution environment of a computation in a substantive and accurate way.

In Computer Science (like in Physics), assumptions should be substantive and models accurate, that is, they should represent the world where the computations are supposed to take place, faithfully enough.

In this scenario, an assumption need not necessarily be a vulnerability: if the assumption depicts the world "as is", there is nothing special that the adversary can attack there. Interestingly, in the formal framework after this postulate, the initial motto might be re-written as:

"Every non-substantiated assumption is a vulnerability"

One should not avoid making assumptions, but instead: *make the least set of assumptions needed to solve the problem at hand, making sure that they have satisfactory coverage*, i.e. that they represent the environment faithfully enough. However, it is not always the case that such a postulate is followed.

Take the synchrony dimension and consider an observation made about it: *"synchronism is not an invariant property of systems"* [9]. In fact, systems are not necessarily either synchronous or asynchronous, the degree of synchronism varies in the time dimension: during the timeline of their execution, systems become faster or slower, actions have greater or smaller bounds. However, and very importantly, it also varies with the part of the system being considered, that is, in the space dimension: some components are more predictable and/or faster than others, actions performed in or amongst the former have better defined and/or smaller bounds.

This opened the door to some research on *hybrid distributed systems models* [10] which helped address and solve some of the contradictions expressed in the previous sections. Suppose that we endorsed such a hybrid model, where different subsystems can follow different assumptions with regard to time or failure modes.

Assuming, (1) a synchronous channel on a synchronous subsystem coexisting with its asynchronous counterparts under hybrid distributed systems model, is totally different from assuming, (2) that the former is achieved over an environment following the asynchronous model. Under (2), the hypothesis (synchrony) deviates from reality (asynchrony), that is, it has limited substance and as such is more prone to failure, for example, under attack by an adversary. This fragility does not exist at the start under (1), since the hypothesis (synchrony) matches reality (synchrony). Besides, coverage can be made as high as needed by construction, by using architectural hybridisation, a system design technique that matches hybrid distributed systems models [11].

In fact, the same comments apply to a set of constructs proposed recently by several researchers, whose common distinctive feature is the assumption of 'stronger' components in otherwise 'weak' systems: watchdog triggers, real time clocks and timeouts, periodic task dispatchers, crypto chips, trusted component units, synchronous or faster communication channels. Some of these systems contain an unspoken hybridisation, in the sense that they propose hybrid components, but work with a homogenous model and architecture. Those constructs would perhaps be better deployed under a computational model that understands hybridisation, and an architecture that substantiates the 'stronger' assumptions, to avoid the risk of failures, either accidental or deriving from successful attacks to the vulnerabilities caused by the model mismatches introduced [8].

Hybrid distributed systems models, versus homogeneous models, have the advantage of letting different parts of the system follow different assumptions and models. As such, they accurately represent the limit situations causing the problems discussed in Sections 3 and 4.

6 Conclusion

In this paper, we alerted to the fact that, in computer science, assumptions go well beyond simple mathematical abstractions. This certainly applies to the design of secure systems and algorithms. Assumptions should represent with sufficient accuracy the environment they are supposed to depict, else they amplify the attack space of the adversary and may become the door to security failures of an otherwise mathematically correct algorithm. We exemplified situations where this occurs in two sets of scenarios.

For example when arbitrary failure modes over asynchronous models are assumed but some constraints to behaviour and asynchrony are inserted, without necessarily being made explicitly, let alone enforced. These subtle feathers of synchrony introduce vulnerabilities which often go undetected, exactly because they are not clearly assumed. They become natural targets to the adversary.

The second scenario concerned the problem of resource exhaustion, in the same asynchronous models. The speed of attack is fundamental to determine the pace at which replicas can fail in a FIT algorithm, and in consequence help determine the speed of the defence mechanisms implemented by the same algorithm. However, we showed that no relation between both can be formally defined, because they develop along different timebases which are free-running. However, such a relation is sometimes forced, with the nasty consequence that the system becomes vulnerable in a way that subsequent intrusions may even not entail a violation of the system's explicit assumptions and resource exhaustion may come unwittingly.

Both scenarios define what we may metaphorically call "Trojan-horse"-like pitfalls: the algorithm designer relies that the system will perform as the assumptions say, but in both cases the latter conceal a different behaviour.

References

1. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. Journal of the ACM 32(2), 374–382 (1985)
2. Meyer, F., Pradhan, D.: Consensus with dual failure modes. In: Proceedings of the 17th IEEE International Symposium on Fault-Tolerant Computing, pp. 214–222. IEEE Computer Society Press, Los Alamitos (1987)
3. Dwork, C., Lynch, N., Stockmeyer, L.: Consensus in the presence of partial synchrony. Journal of the ACM 35(2), 288–323 (1988)
4. Christian, F., Fetzer, C.: The timed asynchronous system model. In: Proceedings of the 28th IEEE International Symposium on Fault-Tolerant Computing, pp. 140–149. IEEE Computer Society Press, Los Alamitos (1998)
5. Chandra, T., Toueg, S.: Unreliable failure detectors for reliable distributed systems. Journal of the ACM 43(2), 225–267 (1996)
6. Sousa, P., Neves, N.F., Verissimo, P.: How resilient are distributed f fault/intrusion-tolerant systems? In: Proceedings of the Int. Conference on Dependable Systems and Networks, pp. 98–107 (2005)
7. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing, pp. 51–59. ACM Press, New York (1991)
8. Sousa, P., Neves, N.F., Verissimo, P.: Hidden problems of asynchronous proactive recovery. In: Third Workshop on Hot Topics in System Dependability (HotDep'07) (2007)
9. Verissimo, P., Casimiro, A.: The Timely Computing Base model and architecture. Transactions on Computers - Special Issue on Asynchronous Real-Time Systems 51(8) (August 2002) A preliminary version of this document appeared as Technical Report DI/FCUL TR 99-2, Department of Computer Science, University of Lisboa (April 1999)
10. Verissimo, P.: Travelling through wormholes: a new look at distributed systems models. SIGACTN: SIGACT News (ACM Special Interest Group on Automata and Computability Theory) 37(1) (Whole Number 138) (2006)
11. Verissimo, P.: Uncertainty and predictability: Can they be reconciled? In: Schiper, A., Shvartsman, A.A., Weatherspoon, H., Zhao, B.Y. (eds.) Future Directions in Distributed Computing. LNCS, vol. 2584, pp. 108–113. Springer, Heidelberg (2003)

Smart Cards and Digital Security

Jean-Daniel Aussel

Gemalto, Technology and Innovation, 6 rue de la verrerie,
92197 Meudon cedex, France
jean-daniel.aussel@gemalto.com

Abstract. Smart cards are portable tamper-resistant cryptographic devices that play a key role in digital security. This paper reviews the latest use of smart cards in securing networks, online services, operating systems, and card-holder identity. Smart card network authentication is routinely used on GSM and 3G networks, and this paper shows how the same infrastructure can be extended to perform WiFi access point authentication. Securing online services with smart card is traditionally performed using public key cryptography and certificates, or using one-time-passwords. This paper presents new smart card authentication methods that either allow to reuse already issued cards or infrastructure, or provide stronger card-to-server mutual authentication. Finally, the paper will show how smart cards and trusted platform module have complementary roles for securing the operating systems, and the use of smart cards in identity frameworks such as liberty alliance or Microsoft cardspace.

Keywords: smart card, authentication, security, trusted computing, liberty alliance.

1 Introduction

Smart cards are portable tamper-resistant cryptographic devices that play a key role in digital security. Billions of smart cards have been deployed to secure the Global System for Mobile Communications (GSM) or 3rd Generation (3G) wireless networks, to perform secure payments, or to provide secure documents, such as electronic passports or identity cards. Smart cards have also been used to secure personal computers (PC) but with less overwhelming acceptance, mostly to authenticate users in a corporate environment.

Securing PCs with smart cards is becoming a growing concern in the consumer environment, to better secure existing services, such as home banking online payment services, or to secure emerging services such as triple-play services.

Home banking, online payment, or valued online services are traditionally protected by passwords, which caused them to be the attack of key loggers, phishing, pharming, or DNS poisoning attacks to retrieve the user password and impersonate the real user. Smart cards as two-factor authenticators are important components for securing the end-user identity and credentials[1]. Two-factor authentication is based on *something you know*, such as the personal identification number (PIN) for a smart card, and *something you have*, such as the smart card itself.

Smart cards are also key components for securing triple-play or quadruple-play services provided by telecommunication operators. Triple-play is the provisioning of telephony, high-speed internet access, and telephony over a single broadband connection. Quadruple-play is the triple-play with the addition of wireless telephony, in which WiFi compatible GSM/3G handsets are switching to Voice-over-IP (VoIP) over the internet broadband connection when the user is at home. Smart cards can provide strong authentication of the end-user to the network and to triple-play services, but also provide user credentials and triple-play client applications portability and integrity.

In the first part, we review how smart cards can be used to authenticate the user at the network layer to retrieve an IP address for the PC. The next section is addressing the use of smart cards to authenticate the user at the application level to get access to internet services. Finally, the last section describes how smart card can provide portability and integrity of the client applications.

2 Network Authentication

2.1 Virtual Private Network with PKI

Virtual Private Network (VPN) can be established on PCs with smart cards using the Extensible Authentication Protocol[2]. Windows networking components typically establish a VPN using the EAP-TLS[3] protocol based on public key infrastructure (PKI). A client X509 certificate and its associated private key is used to perform mutual authentication of the user to the remote VPN gateway. The involved cryptography is performed using the Crypto Application Programming Interface (CAPI)[4] (API). Windows operating systems include pure software base and extended cryptographic service providers (CSP), in which case the VPN is established using a certificate and a private key stored on the PC. However, CSPs supporting dedicated smart cards can be deployed on the PC, in which case the VPN is established by computation in the smart card, with the private key securely stored in the smart card, which results in two-factor strong authentication. After successful mutual authentication, a virtual network interface is created, that tunnels packets over the regular broadband PC connection.

2.2 WiFi Authentication

For WiFi authentication to a hot-spot, windows wireless networking components can use EAP-TLS thru a supplicant, which is a system library that allows to authenticate to a WiFi access point, also known as hot-spot. However, telecommunication operators are not keen to use EAP-TLS for authentication their customers toward WiFi hot-spots, one of the main burden being the deployment of the PKI infrastructure, including certificate generation, deployment and management, and the operation of certificate authorities and certificate revocation lists. For this reason, two EAP protocols have been implemented for Wireless LAN authentication: the EAP-SIM[5] and EAP-AKA[6] protocols, that have the advantage of using the mobile network operators cryptographic infrastructure. The EAP-SIM interface between the PC and the SIM is standardized by the ETSI[7] and the WLAN smart card consortium[8].

The EAP-SIM provides strong authentication and session key distribution using the GSM subscriber identity module (SIM). On the PC, the SIM is inserted into a card reader, or can be available with a USB device form factor. GSM authentication is based on a challenge/response mechanism. The SIM card and mobile operator server share a secret key K_i. The A3/A8 authentication algorithm that runs on the SIM card is given a 128-bit random number as a challenge, and computes a 32-bit response and a 64-bit key K_c from the challenge and K_i. The challenge, 32-bit response and K_c constitute a triplet. On the server side, the EAP messages are processed by a radius server connected to the subscriber Home Location Register (HLR) thru an IP/SS7 gateway. The radius server can retrieve a set of triplets from the HLR and perform authentication. Multiple authentication triplets can be combined to create authentication responses and encryption keys of greater strength than individual triplets. EAP-SIM also includes network authentication, user anonymity and fast re-authentication.

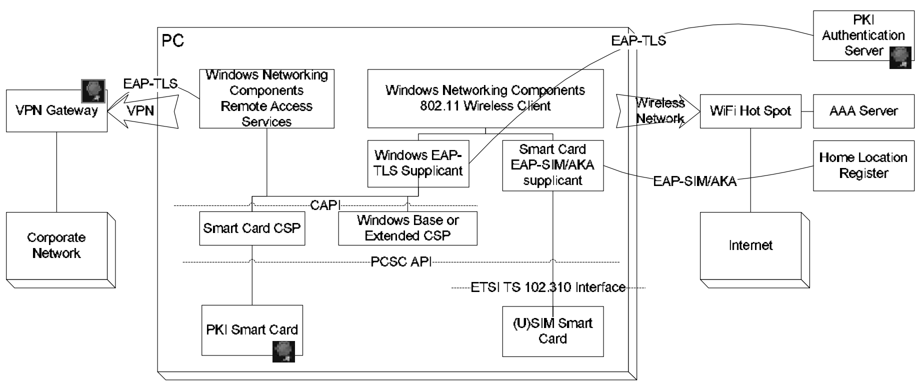

PC
EAP-TLS
PKI Authentication Server
EAP-TLS
VPN Gateway
VPN
Windows Networking Components Remote Access Services
Windows Networking Components 802.11 Wireless Client
Wireless Network
WiFi Hot Spot
AAA Server
Windows EAP-TLS Supplicant
Smart Card EAP-SIM/AKA supplicant
EAP-SIM/AKA
Home Location Register
CAPI
Corporate Network
Smart Card CSP
Windows Base or Extended CSP
Internet
PCSC API
ETSI TS 102.310 Interface
PKI Smart Card
(U)SIM Smart Card

Fig. 1. Network level strong authentication with smartcards. Corporate networks tend to use PKI with EAP-TLS, either thru the remote access services or 802.11 wireless network components. EAP-SIM/AKA is better fit for authenticating consumers towards public hot spots, since it allow reuse of the mobile network server infrastructure for authentication and billing.

EAP-SIM and EAP-AKA have been implemented as wireless network supplicants in both PCs and Windows mobile GSM/3G handsets. On handsets, EAP-SIM/AKA allow the subscriber to obtain an IP connection from a wireless hot-spot, either in a public spot or at home in case of quadruple-play subscription, and perform Voice-over-IP calls or any other IP client application. EAP-SIM/AKA provide means to mobile network operators for authenticating subscribers based on their existing back-office infrastructure.

3 Internet Services Authentication

Successful network authentication provides the PC with an IP connection, either thru a virtual network interface and tunneled over the broadband connection for a VPN, or thru a wireless network interface for 802.11. However, separate application level authentication is still required to have access to internet services for several reasons. First, several users could share the PC, and the user of the internet services, e.g. a home banking application, might not be the same user as the user authenticated at the network level. Moreover, the internet services might be provided by different business actors than the one providing the network connection. Finally, some services might require explicit user consent or proof of presence. For these reasons, different application level methods have been developed to authenticate a user to an internet service. These authentication are mostly used for web server access from a browser, but can be extended to any client protocol, such as the Session Initialization Protocol (SIP) for Voice-over-IP.

3.1 Authentication with PKI and Certificates

Browsers can mutually authenticate with remote servers using the SSL/TLS protocols. The server and end-users are provided with a X509 certificates containing a public key, and a separate private key. The client private key is either stored in a secure location on the PC and protected by password, or stored in a smart card and protected by PIN. In the later case, the private key cryptography is performed in the smart card, and the private key is never read from the card. In some cases, the private/public key pair is generated by the smart card, ensuring that the private key is never available outside the card. In the SSL/TLS protocol, the certificates and private keys are used to mutually authenticate the server and client and negotiate a session key for data encryption.

Browsers typically use external components to perform the required cryptography, and these components are interfaced with standard APIs, such as CAPI[4] for Internet Explorer and PKCS#11[9] for the Mozilla/firefox/Netscape family of browsers. The use of standard APIs allow the plug-in of different implementations of these cryptographic components, called cryptographic service providers (CSP) for CAPI and cryptoki for PKCS#11. In particular, CSPs and cryptokis supporting smart cards can perform two-factor strong authentication for establishing a SSL/TLS sessions.

PKI strong authentication has been used essentially within the corporate environment, because of the burden of deploying the hardware (smart card readers and smart cards), provisioning the client software (smart card reader drivers and CSP or cryptoki components), and managing the certificates. On the server side, no modification of the internet servers is required, since SSL/TLS is supported in virtually all the web servers, or can be implemented using hardware SSL accelerators.

3.2 One-Time Passwords

A popular alternate method for web access authentication using smart cards is the one-time password (OTP). An OTP is a generated password valid only once. Several software or devices can be used to generate the OTP, including personal digital assistants, mobile phones, dedicated hardware tokens, the most secure mean being smart cards which provide tamper-resistant two-factor authentication: a PIN to unlock the OTP generator (*something you know)*, and the OTP smart card itself (*something you have).*

Until recently, OTP solutions were based on proprietary time-based or event-based algorithms. Recently, OATH-HOTP[10] was defined as an open standard by major actors of the industry, to ease interoperability between the devices and servers from different vendors. The HOTP algorithm is based on a secret key and a counter shared by the device and the server, and uses standard algorithms such as SHA-1 and HMAC. On the other hand, smart cards issuers such as financial institutions are pushing OTP algorithms based on their already deployed cryptographic infrastructure [11,12] to decrease the cost of the OTP devices and associated server infrastructure. These solution only require deployment of simple electronic devices with a display and a push button, the OTP being calculated using the payment smart card.

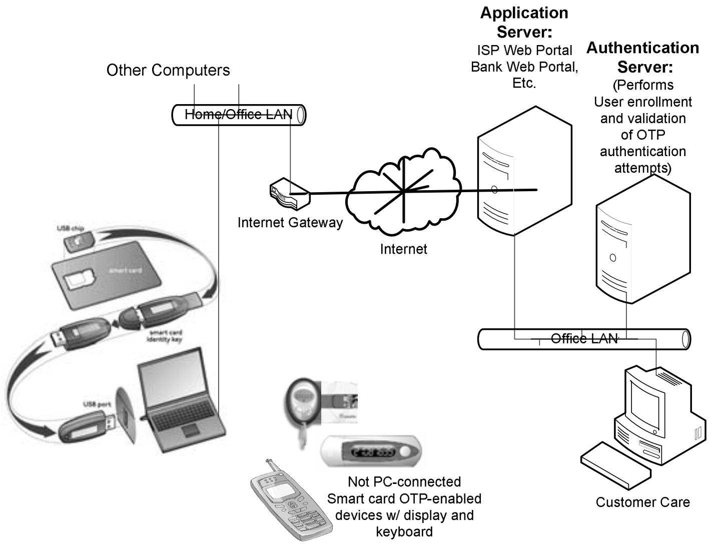


Fig. 2. Internet service authentication with smart card based OTP. The OTP is generated in non connected mode using a smart card inserted into a handset or a dedicated device with push button and display, and the end-user has to key in the password into the browser. In connected mode, the smart card is inserted into the PC and an associated browser plug-in can perform automated form-filling of the password into the browser.

OTP has several advantages over the PKI authentication: it does not require deployment of CSPs or card and reader drivers, nor the management of X509 certificates. OTP is also easy to integrate in solutions that already support simple login and password, in which case only the server has to be modified to validate the password with the OTP server. For authentication servers such as RADIUS servers, supporting the OTP requires the addition of a dedicated plug-in that performs OTP validation. These plug-in generally interface with a dedicated cryptographic hardware that holds a master key that allows computation of the secret key corresponding to an individual OTP smart card.

OTP have the disadvantage of requiring the user to key-in the password, but above all is sensible to security attacks such as the man-in-the-middle attack, since there is no mutual authentication of the PC and the internet server. An attacker can retrieve a valid login/password using a mock-up site, and impersonate the user to the real internet web site.

3.3 Extensible Web Authentication Framework

For network authentication, the EAP allows for arbitrary authentication methods such as EAP-TLS and EAP-AKA. The EAP messages are transported without interpretation over the network components, e.g. the WiFi access point or supplicant, and are only interpreted by the smart card and the radius server authentication policy. An web extensible authentication framework has been built on this principle, for browser authentication with EAP[13].

The extensible authentication framework components are shown in figure 3. When connecting to a web site thru a browser, the user is directed to an authentication web address that holds an EAP gateway java servlet. By accessing the EAP servlet, the browser loads a signed ActiveX or plug-in, the Card Access Module (CAM). The EAP servlet and the CAM are then acting as gateways that carry transparently EAP messages between the smart card and the Radius server. As a result, any protocol can be implemented on top of this framework, the content of the messages being known only by the smart card and the radius server authentication policy.

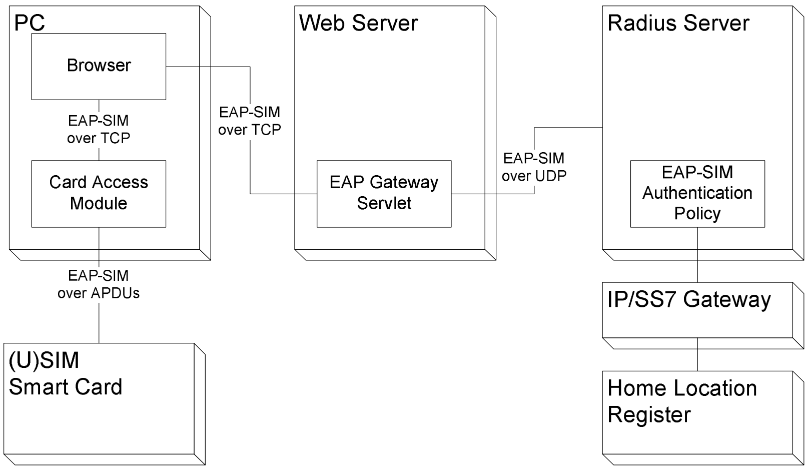


Fig. 3. The extensible strong authentication framework for internet web services authentication. The Card Access Module (CAM) and EAP servlet are acting as gateway to pass EAP messages between the smart card and the radius server. Since the messages are not interpreted by the CAM and the servlet, any authentication method can be implemented. In this figure, the EAP-SIM authentication is illustrated.

This extensible web authentication framework has been used to implement browser authentication with the EAP-AKA algorithm[13], as described in figure 3, and authentication with an Eurocard-Mastercard-Visa (EMV)[14] payment smart card, as described in figure 4. In this EMV strong authentication, a complete payment transaction with a zero amount is performed to authenticate the user.

This extensible authentication framework has the advantage over the PKI standard browser authentication to be open to new protocols, and hence allow the reuse of an existing infrastructure and not require a PKI deployment. Typically, financial institutions can reuse their issued cards and server infrastructure by implementing an authentication based on the EMV specifications, or mobile network operators can reuse their existing authentication and billing servers and deploy SIM cards for PC authentication to their subscribers.

Compared to the OTP authentication, this framework can implement protocols with mutual authentication of the card and server, such as EAP-AKA, and hence avoid man-in-the-middle attacks.

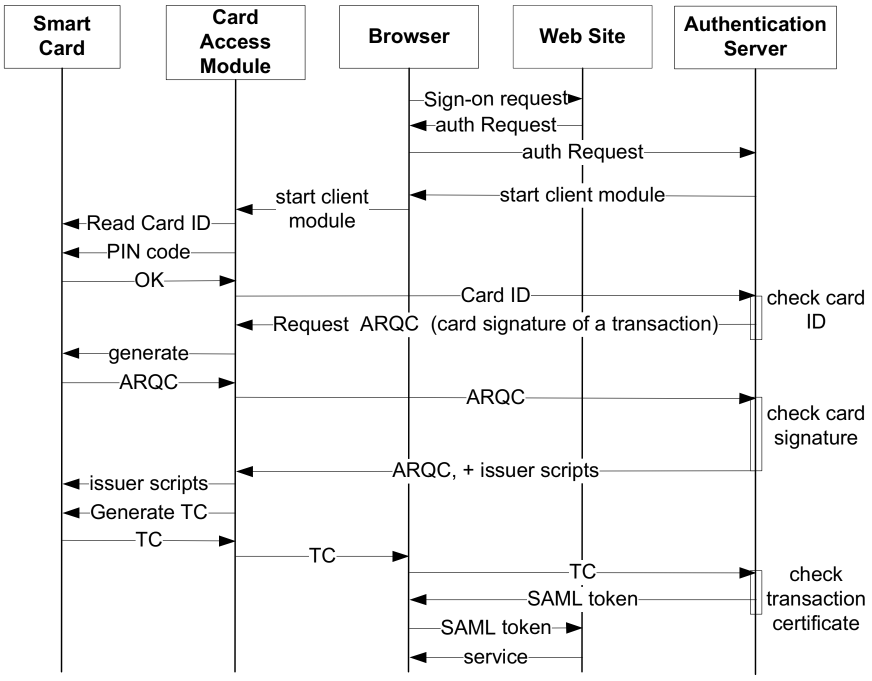


Fig. 4. End-to-end EMV browser authentication. The authentication is equivalent to a zero-amount EMV payment transaction.

3.4 Web Authentication for Identity Frameworks

Given the cost of the deployment and operation of a strong authentication solution for controlling the access to web services, some frameworks have been developed to ensure a clear separation between the web service provider, i.e. the actor that provides a service that requires authentication, and the identity provider, i.e. the actor that authenticate and identify to third-parties the end-user. Actors that already have an authentication infrastructure and deployed credentials, such as mobile network operators or banks, can position as identity providers and leverage on their infrastructure, whereas web service providers can focus on their core business and outsource authentication.

Liberty Alliance is a consortium of industries that defines a set of specifications for identity federation and single-sign-on[15]. Identity federation in liberty alliance is based on the Security Assertion Markup Language (SAML) defined by OASIS [16].

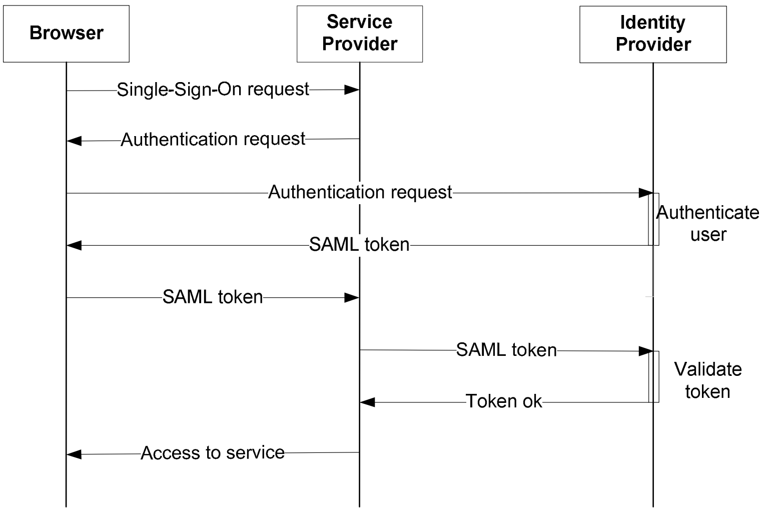


Fig. 5. Liberty Alliance Single-Sign-On. The actual authentication is not standardized in liberty alliance specifications.

As shown in figure 5, the single-sign-on (SSO) procedure is performed thru redirection of the user browser. The service provider redirects the authentication request to the identity provider (IDP). The IDP authenticates and identifies the user, and return thru the browser a SAML token to the service provider. The service provider can optionally validate further the token offline, and returns access to the service if the token is valid. The SSO requires a one-shot initialization phase called federation, in which the IDP and service provider exchange an opaque identifier to the user. This opacity ensures that the IDP and service provider do not share the respective identity of the user.

The authentication method is not specified yet by liberty alliance, which requires custom integration of the various methods in the different IDP implementations. IDP operators are free to deploy any strong authentication they see fit. In particular, Mobile network operators have a strong incentive to use their deployed SIM cards and servers for operating an IDP. One option is to use the EAP-AKA/SIM web authentication described earlier. This solution has the drawback to require the issuing of new SIM cards dedicated to PC authentication. Another option is to use the OTA channel and existing SIM cards and the user handset [13], as described in figure 6.

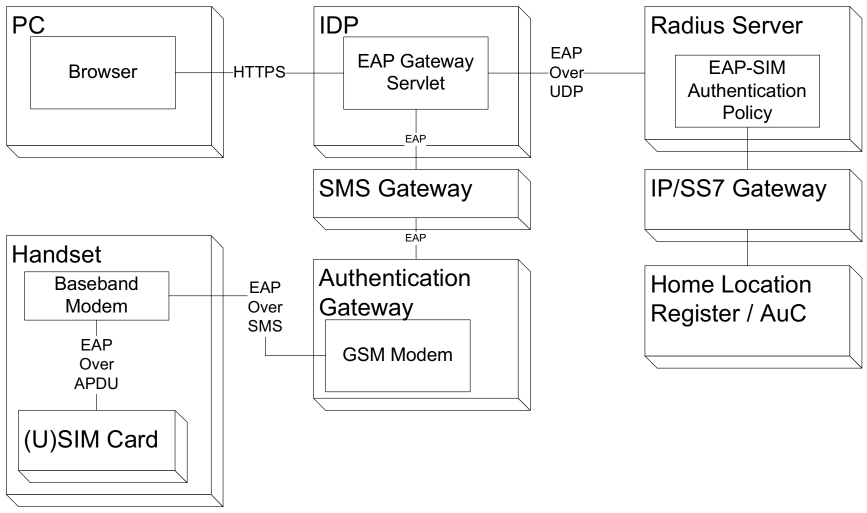


Fig. 6. Liberty alliance strong authentication using over-the-air short-messages

In this solution, called SIMStrong-over-SMS, when the end-user is redirected to the IDP for authentication an end-to-end EAP-SIM protocol is performed between the SIM card in the handset and the Radius server. All EAP messages are exchanged over SMS between the card and the IDP, and as standard radius messages over UDP between the IDP and the radius server. A SIM toolkit applet [17] in the SIM card is prompting for user-consent on the handset, and on consent and successful authentication, the SAML token is returned by the IDP to the browser, who is then authenticated towards the service provider.

The same strong authentication over the SMS channel has been implemented for the recent Microsoft Cardspace framework. Cardspace is a claim based identity management system, in wich a web service provider, called Relying Party in the Cardpace framework, is requesting identity claims to the user. The user can select a virtual card thru a card selector that provides the required claims. Some cards are user-managed, i.e. the claims are not certified, but other claims are certified and managed by an identity provider. To retrieve the claims of a managed card, the user must authenticate to a Secure Token Server (STS) which returns an encrypted and signed token.

The authentication means for cardspace are limited to login/password, Kerberos and X509 certificates. Strong authentication with smart cards is performed either using OTP, or X509 certificates. For OTP strong authentication, the managed card is a login/password card in which the user has to key-in the OTP generated by the smart card device, and the STS is connected to an authentication server that validates the OTP, with all the pros and cons of the OTP discussion above. The X509 strong authentication is based on PKI, in which the STS authenticates the user using a challenge-response mechanism based on the X509 certificate of the user in the managed card and a private key stored in the smart card. Cardspace client components are accessing the smart card thru a new API, the Crypto API Next Generation (CNG).

Smart cards providers typically write a smart card mini-driver[19], also known as a card module, to interface their smart card to the CNG.

The cardspace client framework, i.e. the selector and authentication protocol, is a closed-source component provided by Microsoft and part of the Vista operating system. It is therefore impossible to add an authentication method to the client framework, besides Kerberos, login/password and X509 authentication. Using a second channel, such as the over-the-air channel for mobile network operators allow to perform any type of strong authentication between the STS and the card. This has been implemented for SMS strong authentication[18] as illustrated in figure 7.

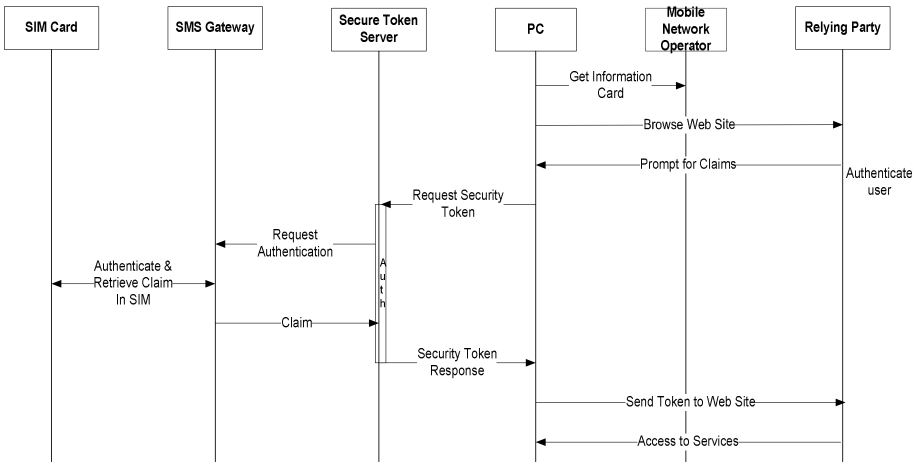


Fig. 7. Cardspace strong authentication over SMS. The secure token server is authenticating the user over the air, and retrieving user's claims inside the SIM card. A SIM toolkit applet is prompting the user for consent.

This second channel can also be a separate TCP/IP connection between the STS and a proprietary component executing on the end-user PC.

4 PC Software Integrity

In the most secure authentication protocols, the security is performed end-to-end between the smart card and the authentication server. The PC and network component in between are just acting like transparent gateway, and cannot decrypt the content of the exchanged messages. However there are several cases where software on the PC has to interact with the card, for example a VoIP soft-phone needs to access the smart card to authenticate the PC to the SIP network. Attackers could impersonate the VoIP softphone to the card, and perform service fraud by calling premium rate numbers, content theft of the phonebook, or wire tap all calls. Similarly, an attacker could hack the browsers and perform man-in-the-middle attacks on connected devices that generate OTP and form-fill browser pages.

To counter these attacks, several initiatives are attempting to validate platform or software integrity towards the smart cards. These initiatives include the Trusted Platform Module.

4.1 Trusted Platform Module and Smart Cards

To address trust concerns while ensuring interoperability and privacy protection, the computer industry has endorsed the specification of hardware-based security computing platforms through the Trusted Computing Platform Alliance (TCPA) then the Trusted Computing Group (TCG). From a technical standpoint, a trusted platform requires (i) a dedicated hardware, the trusted platform module (TPM) implementing the trusted functions, (ii) extra firmware and software implementing measurement and cryptographic functions, and (iii) finally an enhanced operating system.

Initially, the role of the smart cards in the Trusted Computing arena was often blurred by the fact that beyond the strict usage of platform authentication and attestation, the TPM was often presented as a 'general-purpose' hardware security device able to provide the same services than a smart card. Many papers described the TPM simply as a smartcard soldered to the motherboard and the industry has considered that the TPM should be used to provide all the security services to both platform and users of the platform. This position has put TPM and smart cards in a situation of direct competition. However, it has been demonstrated that smart cards still brings value in three key areas[20-21]: (i) the user convenience. The smart card is mobile and can interact with many devices compared to the TPM that is physically attached to the platform (ii) the privacy. The end-user has a direct control over his

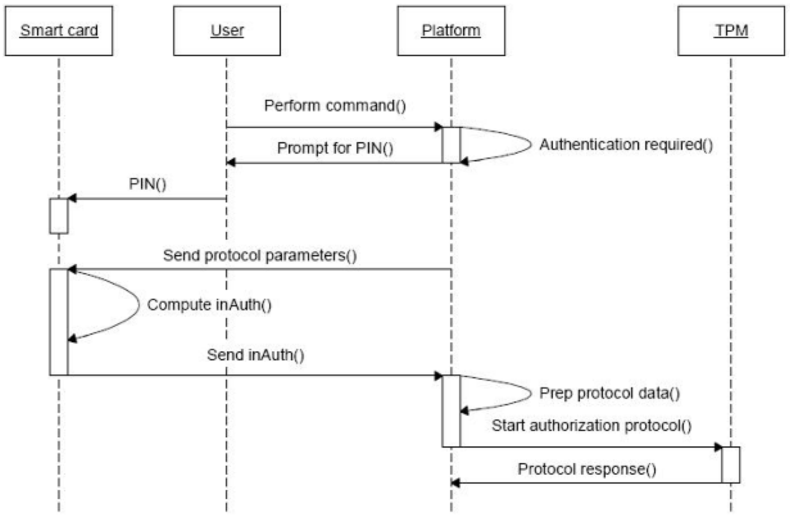


Fig. 8. TPM Authentication workflow where the platform delegates to the smart card part of the computation of the authentication protocol, in this case the inAuth input value [20]

credentials and is allowed to express an explicit consent when presenting them or not (iii) the user authentication. The user presence is guaranteed by the smartcard introduction and PIN presentation.

Concerning the user authentication, access to the TPM protected object is a major issue. This access is protected by a 20-bit shared secret key, the Authorization Data (AD). Knowledge of this 20-bit key, generally in the form of a hash of a password, is a complete proof of ownership of the TPM. New authorization work-flow were proposed [21] to provide two-factor authentication to the TPM with a smart card, and have the additional advantage of not exposing the AD outside the card, as described in figure 8. A new working group of the TCG, the authentication group, has been recently created to specify a standard mechanism to allow authentication sources such as smart cards, to authorize TPM actions [22].

However, the TPM approach for software integrity is not currently fit for large deployment in the consumer arena, such as for home banking or internet payment applications: not all PC are equipped with a TPM, its associated software stack, and it is not supported by all operating systems. An alternate solution is to deploy and store the critical software on the smart card.

4.2 USB Smart Cards

The smart cards are conventionally accessed thru the ISO7816-3 and ISO7816-4 interfaces, i.e. on a half-duplex serial link between the card and a smart card reader. Interfacing the card to the PC thus required the deployment of a smart card reader and its associated device driver. To ease software deployment, a standard reader interface has been defined by the industry, the USB Chip/Smart Card Interface Devices (CCID). Most modern readers now support the CCID standard, and therefore use the same device driver, available on latest operating systems. However, the latest smart cards are USB devices that can implement several USB device interfaces, and do not require a smart card reader since they are connected directly to a USB port of the PC. The USB Device class Integrated Circuit Card Device (ICCD)[23] has been issued to define a standard USB interface to a USB card, as it was the case for the CCID standard for readers. With the ICCD interface, legacy applications access the card thru the PCSC interface and view the USB card as a card inside a smart card reader.

In the same time, the processing power of the card and the available storage has increased, and smart cards can now implement two additional interfaces: the classical mass-storage USB interface, where the card is seen as a removable disk or CDROM drive, and the CDC-EEM interface[25], where the card is seen as a remote TCP/IP node connected thru a network interface card.

The mass-storage interface of the card is used for portability and integrity of the applications, as well as privacy of the user files. Typically, the smart card aware applications can be stored on the mass-storage partitions, avoiding any software deployment on the PC and ensuring the integrity of these portable applications. The access to the card can be protected by a PIN, and a launch pad can give instant access to the applications stored on the card. The on-card mass-storage can be partitioned in separate read-only or read-write partitions, eventually protected by PIN, which provide privacy of the on-card files.

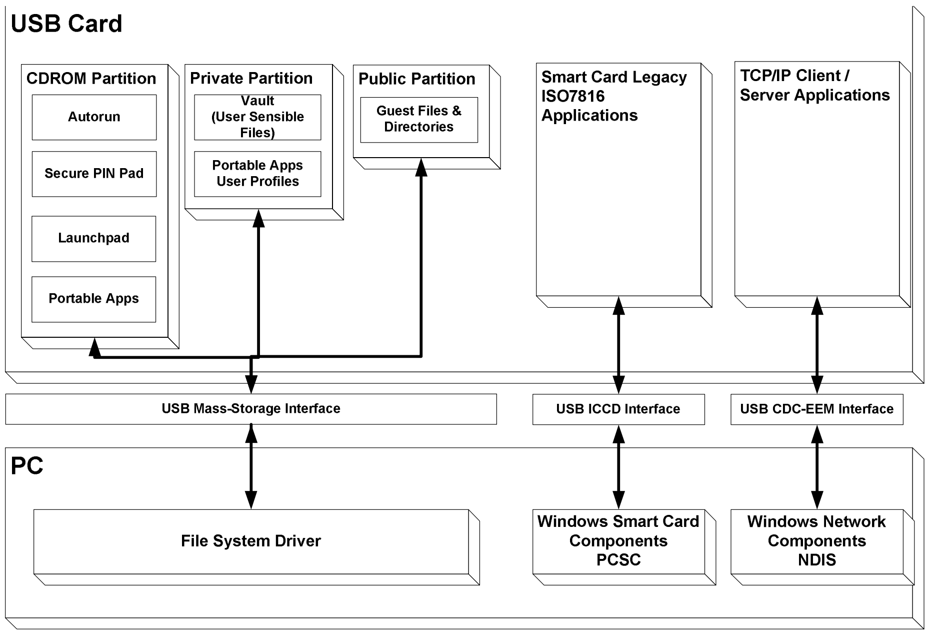

USB Card
CDROM Partition
Autorun
Secure PIN Pad
Launchpad
Portable Apps
Private Partition
Vault (User Sensible Files)
Portable Apps User Profiles
Public Partition
Guest Files & Directories
Smart Card Legacy ISO7816 Applications
TCP/IP Client / Server Applications
USB Mass-Storage Interface
USB ICCD Interface
USB CDC-EEM Interface
PC
File System Driver
Windows Smart Card Components PCSC
Windows Network Components NDIS

Fig. 9. USB smart card. From the PC, the smart card is seen as a file system with read-only and read-write partitions, a smart card reader, and a network interface card. All three interfaces are optional.

In addition, since the portable applications are stored on the card and accessible by the card operating system, security mechanisms can be implemented to ensure that the card is accessed by un-tampered applications. In the xID card[26], the portable application is a host agent that can perform end-to-end TLS security for internet web access authentication. This host-agent is stored on the mass-storage partition on the card, and contains a symmetric key randomly generated at each card insertion. This symmetric key is used to calculate session keys used to establish a secure channel between the host agent and the card. Additionally, these session keys are renewed on a regular basis, and the renewal procedure involves a calculation of the thumbprint of the host-agent on random memory locations, ensuring the integrity of the host-agent.

The network interface provides access to the card using the standard TCP/IP application protocols. This allow for the development of secure applications using standard internet technology. For example, the smart card can implement a web server, hosting static content such as html pages and javascript, and dynamic content such as servlets.

5 Conclusion

Smart cards are tamper resistant device that can dramatically enhance the security of the PC. The smart cards can be used for strong-authentication to gain access to the network or to web service providers, to ensure integrity and portability of a wide range of applications, and to provide privacy for the user identity attributes and personal data. The deployment of USB smart cards and applications is now seamless, since they do not require smart card readers and can be accessed thru standard USB interfaces such as the mass-storage interface. Moreover, the required application and environment can be stored on the card, solving the issue of software provisioning.

As a result, smart cards will play a key role in securing triple-play and quadruple-play services for telecommunication operators, and services for internet service providers.

Acknowledgements. This review paper is based on the work of a large group of individuals at gemalto, that I cannot list completely. The reference section will point to some of them that deserve full credit for the corresponding work. Special thanks however to Michel Beziat, Philippe Smadja, Jerome d'Annoville, Fadela Letourneur, Emmanuel Huet, Christian Cudonnec and Philippe Jutel.

References

1. Binational Working Group on Cross-Border Mass Marketing Fraud, Report on Phishing, October 2006 (2006), http://www.usdoj.gov/opa/report_on_phishing.pdf
2. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP), RFC 3748, IETF (June 2004)
3. Aboba, B.: PPP EAP TLS Authentication Protocol, RFC 2716, IETF (October 1999)
4. Microsoft, Cryptography API, http://msdn2.microsoft.com/en-us/library/aa380255.aspx
5. Haverinen, H., Salowey, J.: Extensible Authentication Protocol Method for GSM Subscriber Identity Modules (EAP-SIM), RFC 4186, IETF (January 2006)
6. Arkko, J., Haverinen, H.: Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA), RFC 4187, IETF (January 2006)
7. ETSI, Smart Cards: Extensible Authentication Protocol support in the UICC, V6.2.0 (September 2005)
8. WLAN Consortium, EAP-SIM Handler Specification Version 1.1 (August 1, 2004) Microsoft Cryptography API: Next Generation (2004), http://msdn2.microsoft.com/en-us/library/aa376210.aspx
9. RSA Laboratories, PKCS#11 v2.20: Cryptographic Token Interface Standard (June 28, 2004)
10. M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., Ranen, O.: HOTP: An HMAC-Based One-Time Password Algorithm, RFC 4226, IETF (December 2005)
11. Mastercard, OneSmart Authentication, https://mol.mastercard.net/mol/molbe/public/login/ebusiness/smart_cards/one_smart_card/biz_opportunity/cap/index.jsp
12. Visa, Dynamic Pass Code Authentication (2006), http://www.visaeurope.com/aboutvisa/products/dynamicpasscode.jsp
13. Van Thanh, D., Nachef, A., Aussel, J.-D., Jørstad, I., Perlman, R., Vigilante, J., Van Thuan, D., Jønvik, T., Ar Foll, F.: Offering SIM Strong Authentication to Internet Services, SIMstrong White Paper, 3GSM World Congress, Barcelona, February 13-16, 2006 (2006) available on http://www.simstrong.org
14. EMVCo, EMV 4.1 Specifications (June 2004), http://www.emvco.com/specifications.asp
15. Liberty Alliance Specifications, http://www.projectliberty.org/specifications__1

16. OASIS, SAML v2.0 specifications (March 2005), http://www.oasis-open.org/specs/index.php#samlv2.0
17. ETSI, Specification of the SIM Application Toolkit for the SIM – Mobile Equipment Interface, GSM 11.14 v. 5.9.0 (1996)
18. Van Thanh, D., Aussel, J.-D., Jørstad, I., Van Thuan, D., Jønvik, T., Andresen, L.: Unified SIM Strong Authentication for Cardspace and Liberty Alliance, 3GSM World Congress, Barcelona, February 12-15, 2007 (2007) available on http://www.simstrong.org
19. Microsoft, Smart Card Minidriver Specification for Windows Base Cryptographic Service Provider (Base CSP) and Smart Card Key Storage Provider (KSP), Version 5.06a (January 18, 2007)
20. George, P.: User Authentication with Smart Cards in Trusted Computing Architecture. In: SAM 2004, CSREA Press (2004)
21. George, P., Maunier, G.: Combining User and Platform Trust Properties to Enhance VPN Client Authentication. In: International Conference on Security and Management (SAM'05), Las Vegas, Nevada, USA, (June 20-23, 2005)
22. TCG, Authentication Working Group Charter (2007), https://www.trustedcomputinggroup.org/about/WG_Charters/Work_Group_Charters_Summary_rev1_2.pdf
23. Donnat, F., Drabczuk, N., Drews, S., Fruhauf, S., Leydier, R., Schneckenburger, C., Weiss, D.: USB Implementers Forum, ICCD Specification for USB Integrated Circuit(s) Card Devices, Revision 1.0 (April 22, 2005)
24. USB Implementers Forum: USB Serial Bus Communication Class, Subclass Specification for Ethernet Emulation Model Devices, Rev. 1.0 (February 2, 2005)
25. Ali, A.M., Lu, H.K.: Securing the Internet through Plug-n-Play Smart Cards. In: Proceedings of the 2007 International Conference on Internet Computing, ICOMP'07, Las Vegas (June 25–28, 2007)

Virus Throttle as Basis for ProActive Defense

Mauricio Sanchez

ProCurve Networking by HP
8000 Foothills Blvd., MS 5557, CA 95747, USA
mauricio.sanchez@hp.com

Abstract. The spread of viruses and worms has severe implications on the performance of virtually any network. Current methods to stop the propagation of malicious code rely on anti-virus signature recognition to prevent hosts from being infected. Unfortunately, the latency between the introduction of a new virus into a network and the implementation/distribution of a patch can be significant. Within this period, a network can be crippled by the abnormally high rate of traffic generated by infected hosts. Previous research has provided a mechanism for controlling the rate at which a host can make new network connections when exhibiting virus-like behavior. Extending this technology to network routers provides the benefit of network protection without the need for individual client support, and serves as an initial step in developing a virus-resilient network. This paper/presentation reflects on the unique challenge of adapting the Virus Throttle mechanism to HP ProCurve network switch routers. Also discussed is the method of proving that it works in realistic network conditions to protect against worms without interfering with normal network traffic.

Keywords: Switch, router, virus, worm, behavior, throttle.

1 Introduction

Security issues are not going away.

More networks are being attacked and threatened, in more devious and creative ways, than ever before. Incidents range from viruses and worms to Trojan horses and internal sabotage.

According to the 2006 CSI/FBI Computer Crime and Security Survey [1] of U.S. corporations, government agencies, financial institutions, medical institutions and universities, the majority of organizations experienced computer security incidents during the previous year. Of those that experienced incidents, nearly one-quarter reported six or more attacks during the year.

At the same time, the information technology (IT) industry itself is evolving in ways that make it both more important and more difficult to secure networks. Some important factors include:

- Openness driven by the Internet, and the need to make resources available – securely – to more people;

- An increasingly mobile workforce, and the challenge of making the network available whenever and wherever people want to connect; and
- The convergence of voice, video and data over a single network, which can deliver greater efficiency if they can be run over a single network, thus overcoming the hassle and expense of running multiple networks.

The costs of security are rising, as are the costs of failing to provide effective network security. We at ProCurve assert that organizations must consider the network infrastructure as an integral component of any comprehensive security solution and use methodologies and technologies that HP ProCurve has championed, such as with its Adaptive Networks[2] and ProActive Defense[3] technology visions.

2 Adaptive Networks and ProActive Defense

The right network infrastructure can help in meeting business goals, enabling an organization to compete effectively by fortifying security, reducing complexity across the organization and increasing productivity.

In an era of change, organizations must be concerned about managing complexity, countering security threats that accompany open and wireless networks, making information and resources accessible to those that need them, handling the burdens associated with regulatory compliance, and supporting current and future applications. Many organizations underestimate the importance of the right network infrastructure in IT's role of making the organization nimble and effective. Too often, even IT-savvy executives make the mistake of treating a network as simply "plumbing" for moving data around. In an era of fast-paced global competition, this mistaken approach to networking can be disastrous.

The ideal network infrastructure ensures that information assets remain secure from internal and external attacks, helps an organization become both internally and externally productive, and is easy to configure, operate and maintain. The network infrastructure must be built to flex and change with the organization and application needs. In a word, an organization must have a network that can adapt.

What is an adaptive network? It is a cohesive, flexible network infrastructure that enables an organization to:

1. fortify security,
2. reduce complexity and
3. increase productivity.

Importantly, an adaptive network is:

- Adaptive to users, which means personalization. Each user enjoys a personalized network experience, for the benefit of both the user and the organization. At the same time, personalization enables organizations to protect information and assets by controlling users' access, thus strengthening security. And because personalization in an adaptive network happens automatically, it is simple for network managers to deliver this powerful capability.
- Adaptive to applications, which means application enablement and optimization. Adaptive networks let you get the most from all your applications. They are

able to easily integrate whatever comes along, whether it's IP telephony, video conferencing, IP video surveillance, Web-based applications, on-demand computing, collaborative applications, video on demand, next-generation applications or future applications that have not yet even been conceived.

– Adaptive to organizations, which means evolving and responding to changing needs. Adaptive networks let organizations focus on their business and goals, rather than devoting exorbitant time, money and resources to managing the network and keeping it running. Organizations retain complete control over their network's operation, but implementation of their policies is handled automatically and centrally by an adaptive network.

With an adaptive network, an organization can focus on its core business so it becomes and remains more competitive within the rapidly evolving global ecosystem. The network infrastructure becomes a strategic asset helping an organization thrive and compete. Users get access to the information and resources they need to be most effective – while the critical information, resources and assets remains available over the network, secure from unauthorized users and safe from attacks.

From an IT perspective, key IT goals fulfilled by an adaptive network include reducing complexity, increasing return on overall IT investments, establishing a service-oriented architecture (SOA), consolidating technologies wherever possible and optimizing Web services, distributed applications, collaboration, virtualization and personalization.

2.1 Basis for the Adaptive Network: Intelligence

In ProCurve's adaptive networks vision, intelligence – of the right kind and in the right location – is the key ingredient. This outlook derives directly from the ProCurve Adaptive Edge Architecture (AEA)[4], the network industry's first strategic blueprint to meet the challenges of adding performance, security and intelligence at the network edge while providing a single strategy for a secure, mobile and converged network.

The AEA is significant because it contrasts with the competitive approach, which comprises a collection of strategies resulting in a complex and technically unwieldy "network of networks." ProCurve's AEA strategy – providing "control to the edge" with "command from the center" – was a breakthrough when first introduced and now has become widely accepted.

The AEA presented a stark contrast to the prevailing view of networks as core-centric, with the edge populated by unintelligent switches and connectivity devices. In contrast, ProCurve understood that changing business and technology needs demanded a network that could scale to accommodate more robust applications and capabilities – and that this could be achieved only with intelligent devices at the client edge of the network.

Built on an AEA foundation, an adaptive network must have embedded intelligence that allows it to understand the user, the policies established by the organization, the full range of applications and the needs of the organization itself. In an adaptive network, intelligence is distributed throughout the network, with emphasis placed at the network edge, where users and devices connect – and where traffic ultimately enters and exits the network.

2.2 ProCurve ProActive Defense

ProCurve's comprehensive security vision and strategy – ProCurve ProActive Defense – delivers a trusted network infrastructure that is immune to threats, controllable for appropriate use and able to protect data and integrity for all users. The three main pillars of the ProActive Defense strategy are as follows:

- *Access Control*: Proactively prevents security breaches by controlling which users have access to systems and how they connect in a wired and wireless network.
- *Network Immunity*: Detects and responds to internal network threats such as virus and worm attacks; monitors behavior and applies security information intelligence to assist network administrators maintain a high level of network availability.
- *Secure Infrastructure*: Secures the network for policy automation from unauthorized extension or attacks to the control plane; includes protection of network components and prevention of unauthorized managers from overriding mandated security provisions; also includes privacy measures to ensure the integrity and confidentiality of sensitive data: protection from data manipulation, prevention of data eavesdropping, end-to-end VPN support for remote access or site-to-site privacy, and wireless data privacy.

Figure 1 depicts the relationship between the three security "pillars" together, regulatory compliance and the adaptive edge architecture.

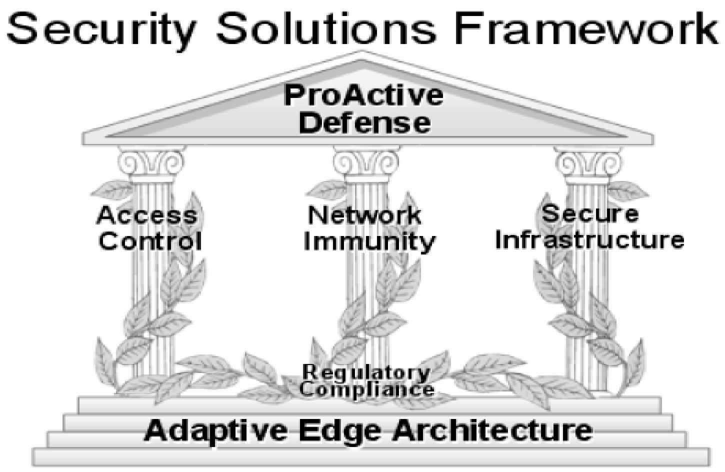


Fig. 1. ProCurve security solutions framework

2.2.1 Simultaneous Offense and Defense

A unique aspect of the ProCurve ProActive Defense vision and strategy is that it combines both the security offense and security defense at the same time and, most importantly, at the network edge. This combined offense and defense is possible only because ProActive Defense is based on Adaptive EDGE Architecture principles, which drive intelligence to the network edge while retaining centralized control and management.

2.2.2 Offense

The ProActive (offense) piece, which is primarily about access control, is a comprehensive way of managing access to the network, dealing with all types of users: everything from an uncontrolled user to an authenticated user to a fully trusted user.

Today, a multitude of devices connect to the network, including laptops, IP phones, peripherals, PDAs and various wireless devices as well as traditional desktop computers. It is essentially impossible for IT departments to mandate a specific operating environment for all devices that access the network. As a result, it is vital to employ a proactive access control solution that is comprehensive and capable of identifying and controlling access for all users and device types. The access control solution must be capable of proactively validating the integrity and operating state of all users and devices.

2.2.3 Defense

The defense piece of the ProCurve ProActive Defense starts with a trusted network infrastructure that is reliable, self-identifying and fully authenticated.

At the same time, the infrastructure must remain plug-and-play and easy to manage. Security is not effective if it is too complex to implement or if it degrades the performance of the overall system. For that reason, the ProCurve trusted network infrastructure includes built-in threat management and anomaly detection. These capabilities are embedded features that promote the defensive security posture of the trusted network infrastructure.

2.2.4 How ProCurve Implements ProActive Defense

Recognizing that network security is a process rather than a discrete solution, and that it must arise holistically from the network infrastructure itself, ProCurve weaves security capabilities throughout its network infrastructure and offerings.

Here are some highlights of how ProCurve implements its ProActive Defense strategy:

- ProCurve builds defensive security features into its switches, access points and other hardware, enabling the creation of a trusted network environment.
- ProCurve-designed network processor chips – notably, the fourth-generation ProVision™ ASIC[5] – embed policy enforcement capabilities into the Adaptive EDGE Architecture. The ProVision™ ASIC is built into the recently introduced ProCurve Switch 5400/3500 Series products and will be included in future products, as well.
- Integrated security and performance management, via ProCurve Manager (PCM) network management software and ProCurve Network Immunity Manager (NIM) [6], allows network security to be automated as well as pervasive, and it takes the complexity out of security management.

- Distribution of intelligence to the edge of the network enables effective proactive access control, which is enacted by ProCurve Identity Driven Manager (IDM) 2.0 [7], a software module for ProCurve Manager Plus (PCM+). IDM allows organizations to define network access policies that enforce secure access to the network and provide dynamic security and performance configuration to network ports as users connect. IDM lets network administrators proactively control access to the network based upon user or device identity, location, time of day and an end point's integrity.

A clear example of a security technology that captures the essence of adaptive networks and embodies the ProActive Defense mantra is the *Virus Throttle* feature present on several ProCurve infrastructure products. The virus throttle feature broke new ground for advanced security functionality for traditional network infrastructure devices when first introduced.

3 Virus Throttle: From HP Labs to ProCurve

Hewlett-Packard laboratories became interested in virus/worm phenomena in 2001 and began researching alternative methods to reduce their negative effects. Their work resulted in what has now become known as the HP *Virus Throttle*, which was made public in 2003 [5]. At the heart of the HP virus throttle is the assumption that hosts, i.e. an end-user machine such as a laptops or desktops, behave unequivocally different when infected with a worm versus when uninfected. The behavioral difference that can be noted between infected versus uninfected hosts is in the number of connections a given hosts attempts to generate. A connection was defined as a TCP SYN packet sent from a given host (e.g. IP 10.0.0.1) to another host (e.g. IP 10.0.0.2)

The HP lab researchers noted that the typical host running common network applications, such as email or web-browsing, does not need create many new connections to other machines within multi-second time windows. This behavior clearly contrasted with the behavior of a worm infected host. Hosts infected with worms, such as SQL Slammer[9], create hundreds to several thousand connections per second. The obvious and easily detected behavioral delta between infected versus uninfected led the HP lab researchers developing an elegant, yet simple, algorithm for throttling the number of connections a host can generate.

3.1 Need for Adaption

The original design and implementation of the HP Virus Throttle was entirely within the confines of an individual host. The HP labs researchers developed an implementation based on Linux and the added the virus throttle algorithm to the network transmit path. It maintains a working set of recently connected to addresses, and a queue of outbound connection requests. The throttle monitors all outbound connection attempts, allowing through those whose destination address is in the working set, and placing the others on the queue – significantly slowing the propagation of any worm. At regular intervals the throttle removes the top element of the queue, forwards all queued connection requests to its destination address and adds the address to the working set in place of one of the existing entries.

The memory and CPU requirements of the original throttle algorithm were designed to provide an acceptable overhead on a desktop PC or server, dealing with its own network traffic. Implementing it on the 5300xl[10][1] network switch router was a much bigger challenge, because of the following technical and product requirements:

- It had to handle multiple sources of traffic (for example up to 192 interfaces on a ProCurve 5300xl switch) versus just as single source.
- Most traffic is routed in our ProVision™ switch ASICs for performance reasons and our switch ASICs had neither the processing richness nor system resources to support the original virus throttle algorithm.
- Only a very small proportion of traffic is sent management processors (based on a general CPU) to be dealt with in (much slower) software.
- Performance was critical: the throttle could not add significant performance overheads. There is no way every packet can be examined in software against.
- Memory space is a scarce resource: the original host-based throttle algorithm needs to able to store a couple of hundred queued packets – there is not that much spare memory on a network switch.
- Switch hardware could not be redesigned in order to implement throttling. We had to work with the hardware as designed and leverage existing datapath processing blocks in novel ways.
- The ProCurve implementation had to work independent of transport protocol (TCP, UDP, ICMP).
- The ProCurve implementation had to have the ability to enable/disable throttling based on several criteria, including port and/or source address.

3.2 Resulting Throttle Algorithm

In order to understand the mechanics of the resulting virus throttle from ProCurve Labs R&D, it is first necessary to understand the basic architecture of the ProCurve 5300xl switch, which was the first platform to offer virus throttle capability.

The ProCurve 5300xl is a chassis-based routing switch with the following capability characteristics:

- Offers layer 2, 3 and 4 Ethernet switching in 4-slot or 8-slot modular form factors.
- Supports a maximum of 192 10/100 ports or 128 10/100/1000 ports.
- 76.8Gbps crossbar switching fabric for wire-speed intra- and inter-module switching with up to 48 million pps (packet per second) throughput on ProCurve ProVision™ switch ASICs.
- Layer 3 IP routing: provides routing of IP at media speed and a 10,000 route table size.

[1] ProCurve 5300xl Switch was first platform to implement virus throttle technology.

Fig. 2. ProCurve 5308xl-48G Routing Switch

The 5300 series routing switch consists of a chassis switch with a fabric chip on the backplane bus (the master) and individual classification/forwarding engines on each of the blades (the slave). In the master/slave system, new packets arrive on the ports and the switch ASIC checks an internal table known as the Hardware IP Address Table. If the source/destination addresses match host entries present in the hardware table, the packet is simply hardware routed and no additional value add processing is performed. If there is no corresponding entry for either source or destination, the packet is sent through the exception path, namely to the master CPU (on the backplane), and is considered *learn*. This exception path allows new route calculations to be programmed into the hardware ASIC, but also served as a natural point to insert our ProCurve virus throttle algorithm.

All traffic redirected to the master CPU for route calculations is first passed through our virus throttle implementation. In a nutshell, the ProCurve throttle blocks sources that send a burst of packets to more than a certain number of new IP addresses at a sustained rate higher than some threshold. The algorithm has two parameters: the rate threshold and a sensitivity that determines the length of the burst.

The throttle is armed when a source sends packets to "new" addresses at a rate consistently greater than that determined by the rate threshold. While armed, the switch hashes destination addresses of IP packets into the range [0 .. N-1], where N is the sensitivity. It does this as long as the source sustains the rate; if the instantaneous rate falls below the rate threshold, the switch stands down, and discards the tracking data for the source.

When the switch has seen exactly N distinct hash values (i.e., the complete set {0, 1, 2, ..., N-1}), the switch blocks the offending source for a predefined interval of 30 seconds to 5 minutes. After the penalty period has elapsed, the switch unblocks the offending source and clears its tracing data. Alternatively, the throttle may be configured to block the host indefinitely until explicitly re-enabled by a network administrator.

```
N == sensitivity (number of bits)
R == rate threshold (expressed as a time interval)

do forever:
   hash_table[*] = 0
```

```
while rate threshold > time since last unrecognised
dest addr for this src addr:

   bit number = fnv32a(dest addr) modulo N
   hash table[bit number] = 1
   if hash table[0 .. N-1] all == 1:
      set block on src address
      drop all pending packets for src addr
   else
      register dest addr for this src addr
      forward packet
```

Thus we were able to reduce the memory requirements of the ProCurve virus throttle down to N bits per source address as opposed to approximately 2 Kbytes on the HP Labs host-based algorithm.

4 Proving ProCurve Virus Throttle Works

When making claims about security properties of network equipment it is especially important to test these properties in realistic scenarios. Testing of the throttle had to answer three critical questions:

1. Does the hash algorithm work? (How certain can we be that the hash-based technique will be effective?)
2. Does the resulting throttle provide the desired mitigation against worms?
3. Can we show that it doesn't interfere with normal traffic?

4.1 Analysis of Hash-Based Approach

At the heart of the ProCurve throttle algorithm is the connection tracking table based on hashing and accumulating a complete N set within a given time frame. For the hashing operation we chose the fnv32a() hash function given its good reputation for being fast, well-behaved, and in the public domain[11]. In order to verify that our fnv32a hash-and-accumulate approach worked, we developed a simulator that would allow us to determine the typical run-length (i.e. the number of consecutive packets from a worm the switch would let pass before blocking the source) for different hash set sizes.

We code the ProCurve throttle algorithm using the fnv32a function and *n*-bit (where n is the switch sensitivity) bitmap. For each value of *n,* we ran 10,000 trials, each trial determining the number of uniform random 32-bit numbers (representing destination IP addresses in packets from a single source) it took to trip the throttle. We recorded the minimum and maximum numbers and calculated the mean and standard deviation. These results appear in Table 1.

The cumulative distribution curves reveal how likely a switch at a given sensitivity level is to meet some limit on the number of rogue packets forwarded. For example, if the switch is to stop a worm within 20 packets 90% of the time, the sensitivity must be set to 7 or less.

Table 1. Behavior of the fnv32a hash-bitmap based throttle algorithm

n	min	max	mean	std dev
2	2	14	3.02	1.43
3	3	30	5.50	2.59
4	4	43	8.36	3.82
5	5	50	11.48	5.03
6	6	59	14.82	6.36
7	7	76	18.12	7.45
8	8	90	21.87	8.77
9	9	94	25.26	9.71
10	10	127	29.40	11.23
11	11	151	33.27	12.61
12	13	151	37.29	13.85
13	14	140	41.47	15.15
14	16	147	45.67	16.34
15	17	194	49.92	17.99
16	18	181	54.28	18.98
17	21	242	58.57	19.97
18	23	194	63.01	21.51
19	25	220	67.59	22.47
20	26	290	72.13	24.39

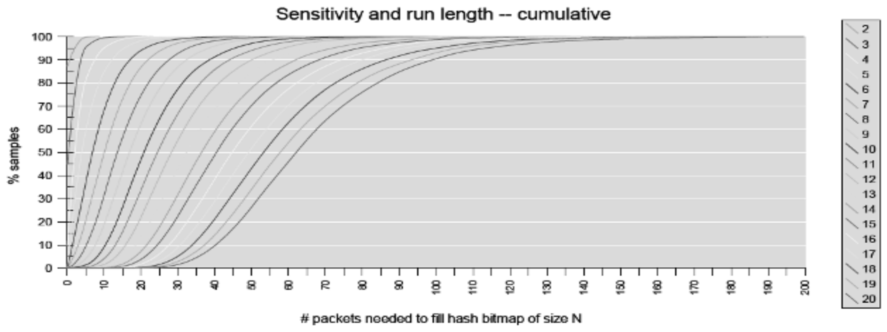


Fig. 3. Cumulative distributions of run lengths

The frequency distribution graph shows how the "spread" of run-lengths increases with the sensitivity (n). The rising side of the curve gives an indication of how likely the throttle is to react to legitimate bursts of packets, as seen with NetBIOS – the further to the left, the more likely false positives are to occur. The falling edge of the curve gives an idea of how many potentially infectious packets may pass through the switch in the worst case before the throttle blocks the source.

In establishing the type of distribution, we asserted that the behavior of the hash-based throttle approach algorithm is effectively the same problem as follows: how many times does an n-die have to be thrown before all n faces appear at least once. Assuming that we were only interested in the number of trials until first success, whereby success was defined as seeing an event (face of a die, index to a bit map) that we had not seen before, we conjectured that we were possibly dealing with a geometric distribution.

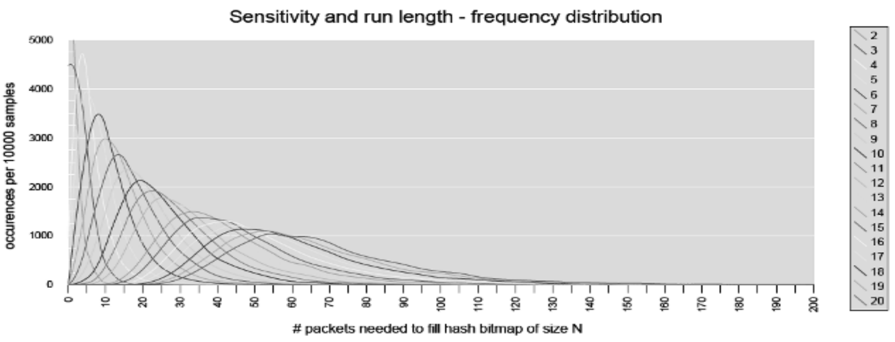


Fig. 4. Frequency distributions of run lengths

Again using the n-side die analogy, before a n-sided die is thrown for the first time (or in the case of the hash-algorithm the arrival of a packet with a specific destination address which we hash), the probability of success (i.e. throwing a value we have not seen before) is *n/n*, or 1. The expected number of throws, $E_0(X)$ is therefore 1. As the second throw is made, one face has already been seen and therefore the probability of throwing it again (i.e. a failure) is *1/n*, so the probability of throwing one we have not seen is *(n-1)/n.*

In a geometric distribution, the expected value for the number of trials is given by:

$E(X) = 1/P(success)$

Thus, having seen one face, we would expect to have to throw the die an average of $E_1(X)=1/((n-1/n)$ or $n/(n-1)$ times taken over many trials. Once we have seen two faces, the probability of throwing a value we have not seen is now *(n-2)/n* and the expected number of throws, $E_2(X)$ to achieve this is the reciprocal, or *n/(n-2).* If we carry on in this way until there is just one face left unseen, the probability of success is *1/n*, and the expected number of trials $E_{n-1}(X)$ is thus *n.*

Therefore, the expected number of throws we have to make in total is the sum of all these *n* expected values:

$$E(X) = E_0(X) + E_1(X) + E_2(X) + \dots + E_{n-1}(X)$$

which is

$$n/n + n/(n-1) + n/(n-2) + \dots + n/2 + n/1$$

which when rearranged is

$$n(1 + 1/2 + 1/3 + \dots + 1/(n-1) + 1/n)$$

While the series in brackets is simple, there is no straightforward formula for calculating an exact value. Fortunately, we were only interested in a few values of n and the table below shows the expected versus observed results for difference values of n.

Table 2. Expected number of distinct address (using geometric distribution approach) needed to hash to n different value and observed average run of 10,000 trials for difference values of *n*

n	expected	observed
2	3	3.02
3	5.5	5.5
4	8.33	8.36
5	11.42	11.48
6	14.7	14.82
7	18.15	18.12
8	21.74	21.87
9	25.46	25.26
10	29.29	29.4
11	33.22	33.27
12	37.24	37.29
13	41.34	41.47
14	45.52	45.67
15	49.77	49.92
16	54.09	54.28
17	58.47	58.57
18	62.91	63.01
19	67.41	67.59
20	71.95	72.13

In switch terms, *n* is the sensitivity and *E(X)* is the expected number of new destination addresses a source has to generate in a burst for the switch to block it.

Our analysis of the ProCurve throttle's hash-based approach demonstrated that it followed a geometric distribution that exhibited what on paper appeared good worm squelching capability. That is, let only a few number of worm traffic through before clamping down the offender.

4.2 Analysis with Real Network Traffic

Encouraged by positive results from the hash-based analysis, we undertook an analysis of the ProCurve throttle with real network traffic. Instead of attempting to create a synthetic traffic model, we reverted to capturing traffic for two weeks at an LAN edge switch within HP's network, upstream from roughly twenty, mostly Windows, a few Linux and handful of HP-UX end-user systems. We reduced the raw data to files containing just the IP address, TCP/UDP port numbers, and packet timestamps that taken altogether comprised approximately 35 million packets.

We then implemented a module in our throttle simulation to "replay" these packets in simulation time, preserving the intervals between packets and the full range of addresses. There are, of course, some limitations to this approach. It is a replay, not a simulation of real systems and their behaviors. For example, blocking a source address in real life would likely change the pattern of packets sent to that address by other systems (i.e. retries, missed connections, and so forth). For our purposes, however, this decided this was not a problem.

In these tests, we set up a series of simulated switches with different parameters. Sensitivity varied from 2 to 16 in increments of 2, while the rate threshold varied exponentially from 0.01 to 3 seconds, approximating 10^(n/2 – 2). We replayed 7 days' worth of captured traffic through each simulated switch, and counted the number of times the rate-threshold was triggered, the number of times the switch blocked an address, and the numbers of packets forwarded and dropped.

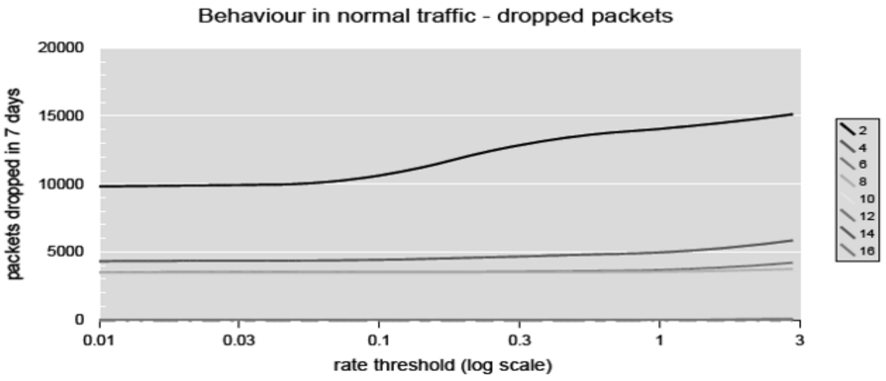


Fig. 5. Packets dropped due to false positives

Since the traffic was from healthy systems exhibiting a range of normal behaviors, the object was to find which combinations of parameters admitted all or most traffic without incident.

In figure 6, there are three distinct groups of curves corresponding to different sensitivity levels:

- Sensitivity of 2: causes a relatively large number of packets to be dropped, with a marked increase in the rate for rate thresholds greater than 0.1 seconds. Looking at the simulation log files suggests that many of these packets are completely “innocent”. This is probably unacceptable level of packet loss in a production network.
- Sensitivities of 4–8: drops around half the number of packets as n=2, but is stable for rate thresholds up to a second The logs suggest most of these packets are due to NetBIOS behavior; thus, on some types of networks, these values might not cause noticeable problems.
- Sensitivities of 10 and up: the switch drops relatively few packets and is stable for rate thresholds up 3 seconds. The logs reveal some NetBIOS packets being lost at a sensitivity of 10, and no packets dropped for 12-16.

We tested the ProCurve virus throttle by recording two weeks of real traffic from a busy network within HP. We built a simulation of the network switch and the throttle algorithm and replayed the real traffic patterns into the simulated switch and observed the results. We also implemented simulations of real and artificial worms and fed the traffic from them into the switch, both on their own and combined with real traffic patterns. A switch implementation was tested on a network with a mix of legitimate traffic and generated traffic exhibiting the behavior of a worm.

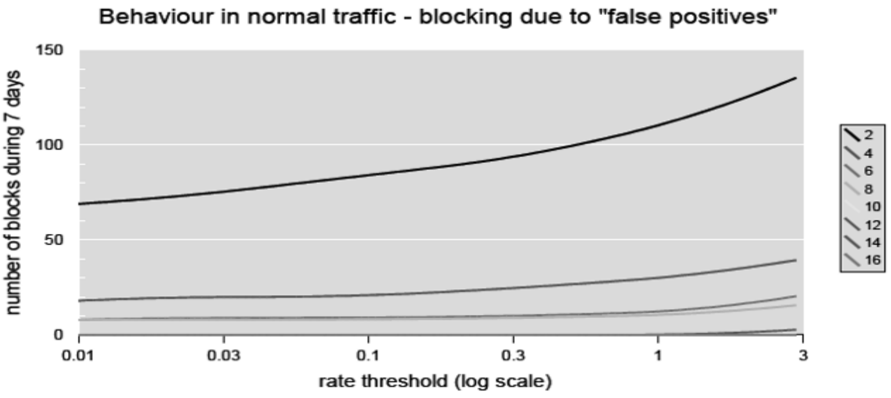


Fig. 6. Source blocking due to false positives

Our simulations showed that the throttle triggers reliably in the presence of worm traffic and interferes minimally with normal traffic. In particular:

1. Using the hash function as described above, the number of packets needed before the throttle blocks an infected source follows a compound geometric distribution. The sensitivity N controls the average number of packets that escape on to the network before the throttle blocks the infected source.
2. The throttle reliably detects and blocks real and artificial worms. The rate threshold must be higher than the gap between worm packets. (The recommended value exceeds the slowest observed worm by a factor of 100.)
3. The throttle interferes minimally with normal observed traffic for sensitivities of 10 and greater.

On the basis of the simulations, it was determined that a reasonable setting for the switch might be a sensitivity of 12 and a rate threshold of 1 second (or 1 Hz). The most variation in settings will come from adjusting the sensitivity: a setting of 4 will provide aggressive throttling while 16 will be rather permissive.

4.2.1 Attack Traffic Results

For our exploration of the ProCurve throttle with attack traffic, we developed a small set of simulated worms. Two, CodeRed II and Sapphire/Slammer are based on worms that are well known and which have caused considerable trouble in their time. A third, nicknamed Sweeper, is designed to probe the response of the switch more systematically.

Code Red (II) was the worm that sparked HP Lab's interest in throttling and other countermeasures. It is not a particularly fast worm: it generates around 100 packets a second, each to a different address. It generates target addresses using an interesting scheme that attempts to exploit locality of reference in networks. 1/2 of the addresses it generates are in the same CIDR Class B address space (e.g, 15.144.*.*), and 3/8 of the addresses are in the same Class A space (e.g, 15.*.*.*). The remainder are 32-bit random numbers.

Sapphire, or Slammer, provides a contrast with Code Red II. Unlike many worms, Sapphire propagates itself with a single, relatively short packet. It is extremely vigorous, generating hundreds or even thousands of packets a second. If Sapphire has a saving grace, it's that its address generation algorithm is flawed, so that it never sends packets to large chunks of the IP 32-bit address space. This may prevent spreading through a given network, but the high volume of traffic is problem enough.

Sweeper is a simulated worm designed to help characterise the throttle response. It starts sending infection packets slowly - eg, 0.25 Hz - and gradually increases the frequency until it reaches its maximum. The start and end rates, and the rate of change are all configurable. In other respects, Sweeper borrows from Code Red, notably the address generation scheme (1/2 same class B, 3/8 same class A, 1/8 random).

The tests with simulated worms give two significant results. First, the rate threshold must be greater than the time between worm-generated packets by a comfortable margin. Here is what happens with Code Red II. Figures 5 and 6 are taken from a simulation of 120 seconds following a single Code Red infection in a Class B network.

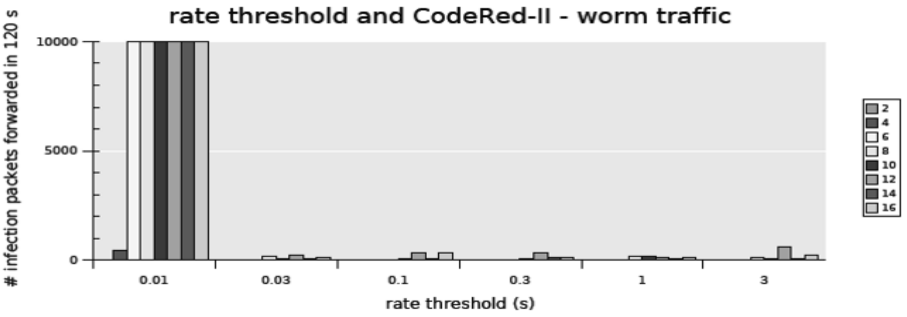


Fig. 7. Code Red slips past a small rate threshold

As observed in real life (and simulated here), Code Red generates packets at around 100 Hz, which is quite slow compared with other worms. At a rate threshold of 0.01 seconds, the switch lets most worm traffic through. Some inter-packet gaps are small enough to alert the switch, but others allow the switch to stand down before it blocks the source. The exception is for a sensitivity of 2.
With the switch not dealing with the problem, the traffic goes off the scale, and the number of infections is also significant.

For higher values of the rate threshold, the switch is effective at reducing the amount of worm traffic and resulting infections, although it's important to note that while the problem of excessive traffic is eliminated, the switch does not completely halt the spread of the worm at any sensitivity level.

For faster worms such as Sapphire/Slammer that generate packets at a very high rate, this problem doesn't appear. Clearly, the rate threshold needs to be chosen to cover the range of perceived threats.

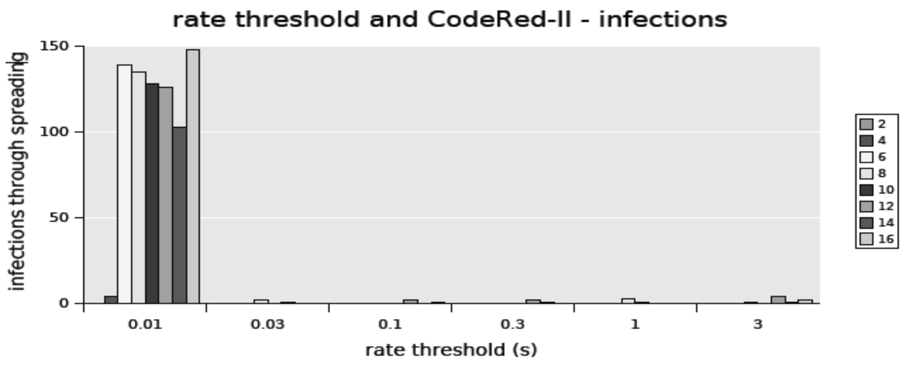


Fig. 8. Code Red infections in 120 seconds

4.2.2 Mixed "Clean" and Attack Traffic Results

The simulation allows an arbitrary set of simulated sources, both normal and worm traffic, to be combined.

With a combination of replayed traffic and simulated worms applied to it, the simulated switch behaved appropriately. Traffic was forwarded as in the traffic-only trial, while the worms were blocked in due course, also as seen before. The switch, as simulated, appears to perform as desired for suitable values of its parameters.

5 Current Status and Future Work

After introducing the ProCurve virus throttle in 2003 on the 5300xl switch, we have continued to refine our thoughts by:

- Designing and patenting a version of the virus throttle with auto-adjusting sensitivities and thresholds.
- Designing and patenting a version of the virus throttle that takes into account layer 4 (e.g. TCP or UDP) port information as part of its analysis
- Extending our next generation ProVision™ switch ASIC with a pure hardware (in ASIC) version of virus throttle that no longer needs to rely on software calculating hashes, maintaining a connection table, etc.
- Developing a version of virus throttle algorithm for our network manager software, ProCurve Manager (PCM), that uses sFlow samples as traffic information input.

Beyond these virus throttle enhancements, to see where ProCurve is headed, one can look at where we are now with other security technologies – and how we got here. Our adaptive networks vision, which is a clear roadmap for the future, arises naturally from our Adaptive EDGE Architecture and our basic value proposition, which has remained constant for many years.

ProCurve has long been dedicated to providing the future-proofed networks that customers need so they can easily adapt to change. This commitment is clearly seen in the ProCurve 5400zl chassis family and ProCurve 3500yl stackable family [13].

These products are Layer 3/4 LAN switches with wirespeed performance and integrated Gigabit PoE.

Both families are based on the ProVision™ ASIC, the fourth generation of custom-designed ASICs from ProCurve, which deliver intelligence and control to the edge of a network. With their capabilities and range, both switch series further sharpen ProCurve's Adaptive EDGE Architecture by delivering advanced functionality to the network edge to meet the evolving needs of security, mobility and convergence applications.

Joining the 5400/3500 intelligent switches are other important ProCurve products. For instance, the Wireless Edge Services Module (WESM) [14] integrates WLAN management and IDM role-based policy enforcement into ProCurve intelligent edge switches. The ProCurve Access Point 530 [15], the most recent ProCurve wireless access point, is a dual-radio 802.11b/g and 802.11 a/b/g access point offering flexible radio and antenna configuration, security, user authentication and access control services.

In addition to providing intelligence to the network edge, ProCurve builds extra capabilities into our intelligent edge switches that allow us to deliver new functionality by moving applications from appliances through blades and into the silicon on every port.

While predictions are necessarily uncertain, it's likely that the future of networks will be one of evolution rather than revolution: There will be further integration of security offense and defense, with ever easier-to-deploy solutions that will allow security protection to always be enabled. More network capabilities – such as personalization – will be automated, which will boost user productivity while greatly reducing the complexity of network management.

Acknowledgements. The author wishes to extend sincere gratitude to the following individuals at HP Labs and ProCurve who had a direct or indirect hand in the work presented in this paper (in alphabetical order): Paul Congdon, Chris R. Dalton, Jonathan Griffin, Ted Mulrane and Matt Williamson.

References

1. CSI/FBI 2006 Computer Crime and Security Survey (2007), http://www.gocsi.com/
2. HP ProCurve (2007). Making Adaptive Networks a Reality (2007), http://www.hp.com/ rnd/ pdfs/VisionTechnicalWhitePaper.pdf
3. HP ProCurve 2007. ProCurve ProActive Defense: A Comprehensive Network Security Strategy (2007), http://www.hp.com/rnd/pdfs/ProCurve_Security_paper_022107.pdf
4. HP ProCurve (2007). Protecting the Extended Enterprise Network: Security Strategies and Solutions from ProCurve Networking (2007), http://www.hp.com/rnd/pdfs/Adaptive_ EDGE_ Arch_wp.pdf
5. HP ProCurve (2007). ProVision ASIC: Built for the future (2007), http://www.hp.com/ rnd/ itmgrnews/ built_ for_ future.htm?jumpid=reg_R1002_USEN
6. HP ProCurve (2007). ProCurve Network Immunity Solution (2007), http://www.procurve.com/ security/Network_ Immunity_ Technical_Brief.pdf
7. HP ProCurve (2006). ProCurve Identity Driven Manager (2007) http://www.hp.com/rnd/ products/ management/idm/overview.htm

8. Twycoss, J., Williamson, M.: Implementing and Testing a Virus Throttle. In: Proceedings 12th USENIX Security Symposium, Washington (2003)
9. Wikipedia. SQL Slammer (2007), http://en.wikipedia.org/wiki/SQL_Slammer
10. HP ProCurve 5300xl switches – datasheet (2007), http://www.hp.com/rnd/products/switches/switch5300xlseries/overview.htm
11. Noll, L.C., Fowler-Noll-Vo (FNV) hash webpage (2007), http://www.isthe.com/chongo/tech/comp/fnv/index.html
12. InMon (2007) sFlow technology website. http://www.inmon.com/technology/index.php
13. HP ProCurve 5400zl/3400yl – datasheets (2007), http://www.hp.com/rnd/products/switches/ProCurve_Switch_3500yl-5400zl_Series/overview.htm
14. HP ProCurve Wireless Edge Services xl Module – datasheet (2007), http://www.hp.com/rnd/pdfs/datasheets/ProCurve_Wireless_Edge_Services_xl_Module.pdf
15. HP ProCurve Wireless Access Point 530 – datasheet (2007), http://www.hp.com/rnd/pdfs/datasheets/ProCurve_Access_Point_530.pdf

Technologies for Protection Against Insider Attacks on Computer Systems

Victor Serdiouk

Joint Stock Company "DialogueScience", 117105, Nagatinskaya 1, office 810, Moscow, Russia
vas@dialognauka.ru

Abstract. During last decade the number of successful intruder attacks has increased in many times. The damage caused by these attacks is estimated in hundreds millions of dollars. Insiders have a significant advantage over others who might want to harm an organization. Insiders can bypass physical and technical security measures designed to prevent unauthorized access. Mechanisms such as firewalls, intrusion detection systems, and electronic building access systems are implemented primarily to defend against external cyber threats. In spite of the complexity the problem, insiders can be stopped by means of a layered defense strategy consisting of policies, procedures, and technical controls. The paper describes a threat model of insider attacks and modern technologies that allow to protect computer systems against insiders. The paper covers advantages and disadvantages of different approaches that are used nowadays for detection and prevention of insider attacks.

Keywords: insider attacks, information security, intrusion detection systems.

1 Introduction

During last decade the number of successful intruder attacks has increased in many times. The damage caused by these attacks is estimated in hundreds millions of dollars. According to the latest results of research conducted by leading institutes and laboratories in the field of information security more than 80% of computer attacks are coming from inside of the company. The 2006 was the year with the largest volume of information leaks in history. The total number of people who suffered from these five leaks was a little under 50 million. For example, on May 3, 2006 criminals stole a hard drive from the house of an employee of the US Department of Veteran Affairs. As a result, personal details of 26.5 million veterans and 2.2 million active-duty servicemen fell into the hands of fraudsters.

Insiders are likely to have specific goals and objectives, and have legitimate access to the system. Security technologies have traditionally focused on perimeter defenses. Most of existing security systems like firewalls, IDS, IPS, antivirus systems, etc. are designed for detection of external threats and almost incapable of protection against insider attacks. These factors led to the necessity of development of new technologies and approaches for internal threat management that will be described in this paper.

The rest of the paper is structured as follows. Section 2 describes the model of insider, including the possible sources of attacks and its possible consequences. Section 3 presents technologies and systems that can be used for detection and prevention of insider attacks. Section 4 summarizes the main results of the paper.

2 Common Model of Insider Attack

The most widespread internal threats are unauthorized access to the computer system (server, PC, or database), searching or viewing confidential data without authorization, and attempts to circumvent or crack the security or audit system. Yet another threat is authorized manipulation of information — changing or deleting data, or saving or processing confidential information in a system not designed for it. According to this classification there are three main types of insider threats:

– fraud – obtaining property or services from the organization unjustly through deception or trickery;
– theft of information – stealing confidential information from the company;
– IT sabotage – acting with intention to harm a specific individual, the organization, or the organization’s data, systems, and/or daily business operations..

The insider threat model includes the following possible information leakage channels (Fig. 1):

– unauthorized copying of confidential data to external devices, such as USB-disks, Firewire-devices, BlueTooth adapters, DVD disks, etc. For example intruder can copy secret information to 100 GB USB stick and then sell it to competitor;
– unauthorized printing out of confidential documents on local and network printers and getting them out of the company;
– unauthorized transmission of confidential data to external Internet servers. For example the attacker can upload a document with confidential data to a Web-server by means of HTTP or FTP protocol and than download it at home. Insider can even encrypt the document before transmitting it over the network;
– theft of devices that store confidential data, i.e. the insider can steal the hard disk from the server, which process the financial data.

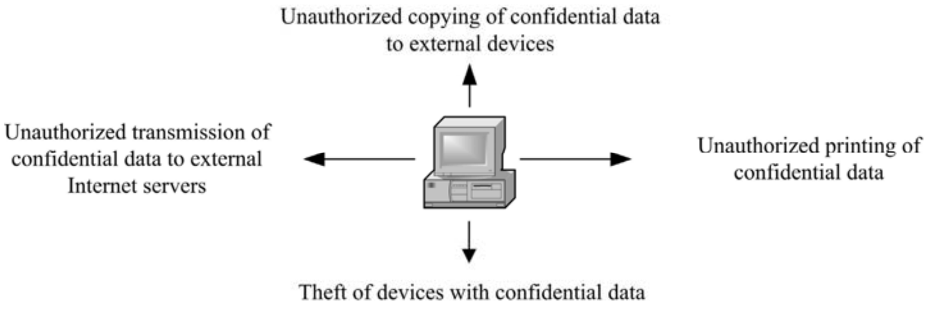

Unauthorized copying of confidential data to external devices
Unauthorized transmission of confidential data to external Internet servers
Unauthorized printing of confidential data
Theft of devices with confidential data

Fig. 1. The possible confidential information leakage channels

The typical source of insider attack is a disgruntled employee, which tries to cause financial damage to the company. These sorts of attacks can cause enormous damage, and not just financially. A confidential information leak is a serious blow to a company's reputation.

For best understanding of common model of insider attack we can assume that an attacker is goal-oriented. Also, he is aware of the location of his potential targets and how to reach them. Commonly, on the base of the resources available, the degree of access to the information, and the motive, recognizes five insider threat scenarios.

– Threat scenario 1. Insiders who make "innocent" mistakes and cause accidental disclosures. Probably the most common source of breached privacy. For example overhead hall conversations, misclassified data, an email, etc.
– Threat scenario 2. Insiders who abuse their record access privileges. Individuals that have access to data and violate the trust associated with that access.
– Threat scenario 3. Insiders who knowingly access information for spite or for profit. When an attacker has authorization to some part of the system, but not to the desired data and gains access to that data by other means.
– Threat scenario 4. The unauthorized physical intruder. The attacker has physical entry to points of data access but has no authorization to access the desired data.
– Threat scenario 5. Vengeful employees and outsiders, such as vindictive patients or intruders, who mount attacks to access unauthorized information, damage systems, and disrupt operations. (In the literature this threat also includes denial of-service attacks, even though the focus is on privacy issues).

Internal threats that are coming from the staff cannot be completely eliminated, but they can be managed and kept to a minimum. When creating an integrated security system, all the potential means of carrying out internal attacks must be taken into account. The protection strategy must be based on two principles: disabling features that are not necessary for users and constantly monitoring any activity of employees, related to confidential information access. Maintaining a balance between these principles is a constant compromise, but it is the only way to create a transparent and flexible security system that provides an effective protection against internal attacks of insiders. The description of different technologies that are used for the detection and prevention of internal threat is cited in Section 3.

3 Technologies for Protection Against Insider Attacks

At present only complex approach can guarantee an effective protection against internal threats. Insider attacks can be prevented only through a layered defense strategy consisting of policies, procedures, and technical controls. Complex approach begins with the establishment of appropriate and effective security policies. Effective policies help ensure that threats to critical assets are understood, managers and users are adequately trained, and actions to be taken when an insider attack is identified are defined. Whenever possible, the policy should reflect the mission of the organization that promulgates it. Therefore, it should codify the rules governing enterprise operations as they are reflected in its information infrastructure and should explicitly exclude activities or operations not needed to support the enterprise's mission. A mission-oriented security policy can aid in selection and introduction of organizational and technical security controls.

3.1 Organizational Security Controls

Organizational security controls usually include the following:

– implementation of risk assessment procedures, that allow define the critical assets of the company, create a threat model and develop a strategy that will minimize the security risks, related to insider attacks;
– development of information security policy, which allows to formalize the process of detection and prevention of insider attacks;
– organization of security awareness training for all employees. The first line of defense from insider threats is the employees themselves. All employees in an organization must understand that security policies and procedures exist, that there is a good reason why they exist, that they must be enforced, and that there can be serious consequences for infractions. Each employee needs to be aware of the organization's security policies and the process for reporting policy violations;
– separation of duties and least privilege. If all employees are adequately trained in security awareness, and responsibility for critical functions is divided among employees, the possibility that one individual could commit fraud or sabotage without the cooperation of another individual within the organization is limited. Effective separation of duties requires the implementation of least privilege, that is, authorizing people only for the resources they need to do their jobs.

Organizational security means should be complemented by technical security controls, that are described in Section 3.2.

3.2 Technical Security Controls

Over the years, organizations have deployed different types of products (e.g., identity management, firewalls, antivirus software) that could provide them with partial protection against insider threats. But as the existing products cannot provide high level of security, new technologies and solutions became necessary to achieve regulatory compliance and improved protection against information leakage. That realization has led to the emergence of new dedicated solutions that can be categorized into the following groups: network-based and desktop-based solutions.

Network-based solutions are used to monitor outbound network traffic and identify the unauthorized delivery of sensitive information via email, Web, IM, P2P, and other channels, according to corporate policies. Most of the network-based security systems are using an architecture in which network sniffers (in most cases Linux-based hardware appliances) are installed next to firewalls to monitor outbound network traffic. The sniffers then reassemble TCP sessions and analyze them to detect sensitive information leakage, as defined by policies and rules. Other products are using proxies for inspecting specific channels (in most cases SMTP email) and block communication when unauthorized contents are detected. In some cases, sniffers and proxies are used in combination.

Desktop-based solutions are designed to enforce data distribution policies on each corporate workstation (both PCs and laptops, in some cases). Based on agents installed into the operating system on each desktop, those solutions allow organizations to gain control over user activities, covering either network-related activities such as attaching a file to an email or IM message; or desktop activities such as printing, copying a file to a USB device, burning to a CD, and so on. Many desktop-based solutions cover both sides.

4 Overview of Existing Security Solutions for Protection Against Insiders

At present there are a lot of vendors on security market which offer security solutions, that allow to protect against insider attacks. This section of the paper presents information about advantages and disadvantages of different security solutions, based on the practical experience of JSC “DialogueScience” in installation and support of these products.

4.1 Microsoft Rights Management Service

Microsoft Corporation released Rights Management Service (RMS) in November 2003. The product became the third Microsoft solution implementing features necessary for digital rights management. Earlier the company released Windows Media Rights Management (for audio and video data), and Digital Asset Server (for e-books).

While forming a set of privileges and rules for a document, the user can restrict the following document uses: viewing, printing, saving, export (Save as), manipulations with clipboard (copy/paste), editing (modification), and use of macros. If a document is an e-mail message, it is possible additionally to restrict its forwarding and the opportunity to respond to it (reply and reply all). All these restrictions can be associated with individual users and for user groups. Every set of privileges and rules can be assigned absolute and relative expiry periods (e.g., a specific expiry date or a number of days after a specified time).

The common algorithm of RMS operation is described below (Fig. 2):

- The author creates a document and forms a set of privileges and rights to manipulate it (Publishing License). The application encrypts the document using a symmetric key.
- The application sends a Publishing License to an RMS server, which should sign it.
- RMS signs the Publishing License and returns it to the application.
- The author sends the file to the document recipients.
- The recipient opens the file. Opening the application sends a Use License request to the RMS server. The request includes the RM Account Certificate (RAC) of the recipient and the Publishing License of the document.

– RMS validates the request and RAC, and identifies the recipient. In the case of a successful validation, RMS grants the recipient a license to work with the document.
– The application receives the license from RMS and follows the rules it contains. The recipient then works with the document.

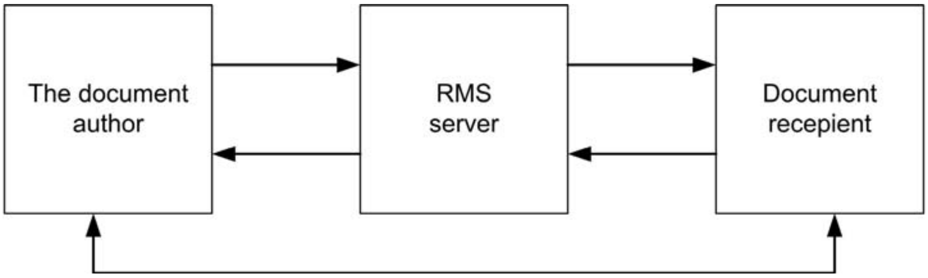


Fig. 2. The logical strucure of RMS

Microsoft RMS usage is based on the following concept. The user, the author of the document, personally defines the set of privileges and rules that restricts the use of a protected document by other users.

4.2 InfoWatch Enterprise Solution

The InfoWatch company was founded in 2003 as a subsidiary of Kaspersky Lab. In 2004 InfoWatch released InfoWatch Enterprise Solution (herein after referred as IES), an integrated solution preventing data leaks. The product is designed for use in large corporate LANs. The tasks of IES include, first of all, real-time blocking of leaks and notification of security officers about such incidents. The deployment of the IES integrated solution helps not only to prevent the theft of confidential information but also provides compatibility with the standards of IT security and compliance with existing laws.

IES is based on a distributed architecture consisting of the following software components:

– Web Monitor, which filters web traffic automatically blocking the leaks of confidential data via web channels (web mail, chats, forums, etc.).
– Mail Monitor, which filters e-mail traffic automatically blocking leaks of confidential data, including in attachments.
– Net Monitor, which monitors on workstation level user manipulations with confidential documents in Microsoft Office and Adobe Acrobat environments, controls the printing of documents, work with clipboard and also tracks file operations (file creation, deletion, modification, reading and renaming).
– Mail Storage receives in real time copies of e-mail messages, preserves them in storage and supports analytical selection in that storage. Thus, the component allows centralized storage of corporate correspondence and its retrospective analysis if necessary.

– Device Monitor allows real-time control over user access to communication ports on a workstation (CD-ROM, floppy, HDD, hard drives, removable dries, COM, LPT, USB, IrDA ports, Bluetooth, FireWire, Wi-Fi).

From the very beginning IES development took into account a wide range of supported formats. The current version of the product supports the following file formats: Microsoft Office (Word, Excel, PowerPoint), Adobe PDF, TXT, HTML, RTF, ZIP, ARJ and RAR. It also filters the following types of traffic: HTTP and SMTP. Support for image file formats (JPEG, GIF, BMP, etc.) and filtering of the traffic generated by instant messaging utilities (ICQ, MSN, AOL, Yahoo, etc.) is planned in upcoming versions.

4.3 The Security Policy Management System "Enterprise Guard"

The system "Enterprise Guard" is designed for active monitoring of users activities of computer system on the basis of collection and analysis of data on events registered on workstations of users.

The system «Enterprise Guard» has a distributed architecture comprising the following components (Fig. 3):

– Agents which are installed at the users' workstations that collect the necessary information. Thereat the agents use minimal hardware resources and do not affect users' workstations performance;
– Server which performs the function of collection, storage and analysis of information coming from the agents;
– Administrator console which is designed for the centralized management of server and agents. The administrator console also includes a report generator tool.

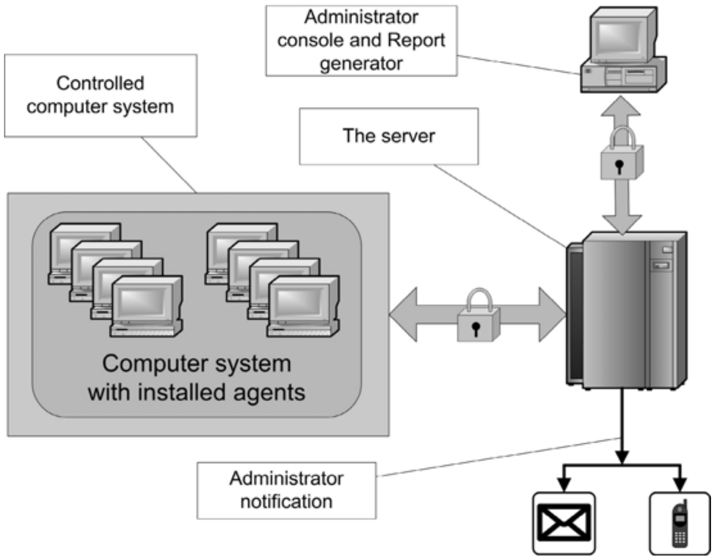

Administrator console and Report generator
Controlled computer system
The server
Computer system with installed agents
Administrator notification

Fig. 3. Structure of the system "Enterprise Guard"

The system "Enterprise Guard" makes possible collection of the following information from the users' workstations:

- beginning/end of the users working session;
- application execution/termination;
- file access operations of the launched applications;
- applications control total alteration;
- dialog boxes opening within the applications;
- keyboard input of CS users;
- application Access to the Internet sources;
- documents printing out of the applications;
- sending/receiving of e-mails.

Based on the analysis of information collected by the agents, "Enterprise Guard" detects the violation of security policy defined by the administrator. After the detection of such violations "Enterprise Guard" can use the following response methods:

- Sending an e-mail message about the detected violation Security Administrator
- Displaying the information about the detected violation on the administrator console;
- Blocking of the workstation of the intruder;
- Generating a screenshot from the intruder workstation;
- Blocking of application that was executed by the intruder;
- Displaying the warning message box on the screen of the potential intruder.

"Enterprise Guard" is equipped with self-security mechanisms providing the system integrity and cryptographic security of proprietary information distributed between the components on the network. Additionally the system controls the availability of its agents by means of sending special echo-request to each agent of "Enterprise Guard".

4.4 Security Control System "DeviceLock"

DeviceLock access management software is a solution to enforcing device related security policy. DeviceLock administrators can set permissions per peripheral port, device class, device model, and unique device. Simultaneously, they can grant or deny access per user group and user, even specifying day of the week and time. In addition, DeviceLock will audit all uploading and downloading activity through local drives and ports. DeviceLock consists of three parts (Fig. 4):

- DeviceLock Service which is installed on each client system, runs automatically, and provides device protection on the client machine while remaining invisible to that computer's local users.
- DeviceLock Enterprise Server is the optional component for centralized collection and storage of the shadow data and audit logs in MS SQL Server.
- The management console is the control interface that systems administrators use to remotely manage each system that has DeviceLock Service. DeviceLock ships with three different management consoles: DeviceLock Management Console

(the MMC snap-in), DeviceLock Enterprise Manager and DeviceLock Group Policy Manager (integrates into the Windows Group Policy Editor).

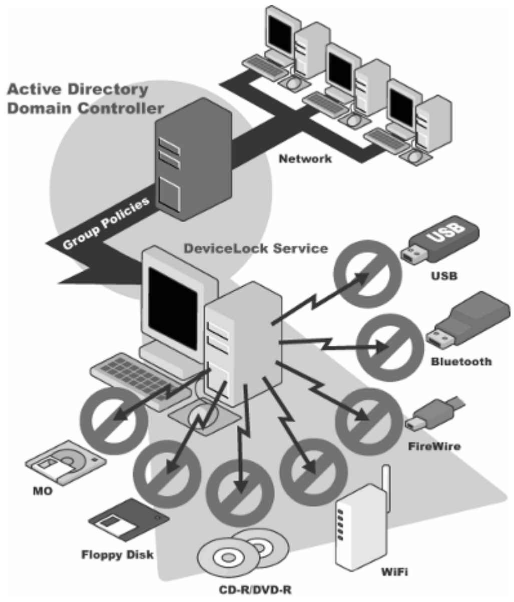


Fig. 4. The structure of system "DeviceLock"

DeviceLock allows to control which users or groups can access USB, FireWire, Infrared, COM and LPT ports, WiFi and Bluetooth adapters, DVD/CD-ROMs, floppy drives, other removable and Plug and Play devices. It is possible to set devices in read-only mode and control access to them depending on the time of day and day of the week. DeviceLock also allows you to authorize a specific model of device to access the USB port, while locking out all others. You can even "White List" a single, unique device, while locking out all other devices of the same brand and model, as long as the device manufacturer has supplied a suitable unique identifier, such as a serial number.

The DeviceLock optional data shadowing capability significantly enhances the corporate IT auditor's ability to ensure that sensitive information has not left the premises on removable media. It captures full copies of files that are copied to authorized removable devices, burned to CD/DVD or even printed by authorized end users. Shadow copies are stored on a centralized component of an existing server and any existing ODBC-compliant SQL infrastructure of the customer's choosing.

5 Conclusion

Recent events have shown that organizations across industries cannot afford to continue to ignore the potential for insider attacks. As organizations grow, they employ workforces that are increasingly spread across geographies; they implement systems that are more heterogeneous, more complex and more connected; they retain more confidential data; and they are subject to changing regulatory requirements. Traditional boundaries between organizations, partners, users and customers have become blurred, making security policies more difficult to define and enforce. Organizations must be prepared to fend off attacks wherever they originate — by addressing vulnerabilities in precisely this gap between traditional business and the open, distributed organizations of today and the future.

The threat of attack from insiders is real and substantial and the impact from insider attacks can be devastating. According to one complex case of financial fraud committed by an insider in a financial institution resulted in losses of almost $700 million. Insiders have a significant advantage over others who might want to harm an organization. Insiders can bypass physical and technical security measures designed to prevent unauthorized access. Mechanisms such as firewalls, intrusion detection systems, and electronic building access systems are implemented primarily to defend against external cyber threats. In spite of the complexity the problem, insiders can be stopped by means of a layered defense strategy consisting of policies, procedures, and technical controls.

References

1. Alberts, C., Audrey, D., Zajicek, M.: Defining Incident Management Processes for CSIRTs (CMU/SEI-2004-TR-015). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University (2004)
2. Avdoshin, S., Serdiouk, V.: Some approaches to information security of communication networks. Slovenia, Informatica 26, 1–10 (2002)
3. Cappelli, D., Moore, A., Shimeall, T., Trzeciak, R.: Common Sense Guide to Prevention and Detection of Insider Threats. Carnegie Mellon University (2006)
4. CERT. Survivability and Information Assurance Curriculum (SIA), 2006 (2006), http://www.cert.org/sia
5. Serdiouk, V.: Behavior-based model of detection and prevention of intrusions in computer networks. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, Springer, Heidelberg (2005)
6. Serdiouk, V.: Technologies for the protection against information leakage. Vek kachestva 3, 62–67 (2005)
7. Yachin, D.: InfoWatch: A Multilayered Approach for Information Leakage Detection and Prevention. IDC Whitepaper (2005)
8. Gordon, L., Loeb, M., Lucyshyn, W., Richardson, R.: CSI/FBI Computer Crime and Security Survey, Computer Security Institute (2006)
9. Ramkumar, C., Anusha, I., Hung, N., Shambhu, U.: A Target-Centric Formal Model For Insider Threat, Department of Computer Science and Engineering State University of New York at Buffalo Buffalo, NY 14260 (2003)
10. Anderson, R., Bozek, T., Longstaff, T., Meitzler, W., Skroch, M., Wyk, R.: Research on Mitigating the Insider Threat to Information Systems. In: Proceedings of a Workshop Held (2000)

Access Control and Declassification[⋆]

Gérard Boudol and Marija Kolundžija

INRIA, 06902 Sophia Antipolis, France
{Gerard.Boudol,Marija.Kolundzija}@sophia.inria.fr

Abstract. We integrate programming constructs for managing confidentiality in an ML-like imperative and higher-order programming language, dealing with both access control and information flow control. Our language includes in particular a construct for declassifying information, and constructs for granting, restricting or testing the read access level of a program. We introduce a type and effect system to statically check access rights and information flow. We show that typable programs are secure, that is, they do not attempt at making illegal read accesses, nor illegal information leakage. This provides us with a natural restriction on declassification, namely that a program may only declassify information that it has the right to read.

Keywords: Access control, declassification, language-based security, secure information flow, stack inspection, type and effect systems.

1 Introduction

In a world where more and more information is digitalized, and where more and more people have access to it, most often by means of dedicated software, protecting confidential data is a concern of growing importance. Controlling access rights is obviously necessary, and access control techniques have indeed been developed and implemented long ago. However, as it has been argued in [7,10,18] for instance, access control is not enough to ensure end-to-end confidentiality. One issue is to prevent authorized users to publicly disclose confidential data, especially when the "users" are publicly accessible softwares receiving and storing these data. Therefore, one should have means to control that such programs do not contain security bugs, and in particular that programs do not implement illegal *flows of information.* This is the aim of the language-based approach to information-flow security.

Since the pioneering work of Denning [7], the classical way of abstractly specifying secure information flow is to use a *lattice of security levels.* The "objects" – information containers – of a system are then labelled by security levels, and information is allowed to flow from one object to another if the source object has a lower confidentiality level than the target one. That is, the ordering relation on security levels determines the legal flows, and a program is secure if, roughly

[⋆] Work partially supported by the ANR-06-SETI-010 grant. The second author is supported by a PhD scholarship from Torino University.

speaking, it does not set up illegal flows from inputs to outputs. This was first formally stated via a notion of *strong dependency* by Cohen in [6], which is most often referred to as *non-interference*, according to the terminology used by Goguen and Meseguer in [9].

A lot of work has been devoted to the design of methods for analyzing information flow in programs. Since the work of Volpano, Smith and Irvine [24], the classical approach by now is to use a type system for this purpose, with well-known advantages, like preventing some programming errors at an early stage, while avoiding the run-time overhead caused by dynamic checks. Type systems for secure information flow have been designed for various languages, culminating with Jif (or JFlow, see [13]) and Flow CAML [22] as regards the size of the language (see the survey [18] for further references). In this paper we shall base our study on Core ML, a call-by-value λ-calculus extended with imperative constructs, where the "information containers," to which security levels are assigned, are memory locations – references, in ML's jargon (in an extended setting, that could also be files, entries in a database, library functions, or class names as in [2]).

There is still a number of issues to investigate in order to make the techniques developed following the language-based approach to information-flow security useful in practice – see [26] for a review of some of the challenges. One of the challenges is *declassification*. Indeed, there are many useful programs that need to declassify information, from a confidential status to a less secret one. A typical example is a password checking procedure, which delivers to any user the result of comparing a submitted password with secret information contained in a database, thus leaking a bit of confidential information. Such a program is, by definition, ruled out by the non-interference requirement, which is therefore too strong to be used in practice. The problem of taking into account declassifying programs in checking secure information flow has recently motivated a lot of work, like for instance [1,4,5,12,15,19]. We refer to [21] for a thorough discussion of this problem. In [1], the authors introduced a programming construct for managing declassification, that consists in declaring *flow policies* with a local scope. Then for instance, the critical parts of a password checking procedure should be enclosed into a statement that it is, for a while, legal to make information flow from a secret to a public level (exactly *which* information should flow is left to the programmer's responsibility). This is supported in [1] by the definition of a new confidentiality policy, called the *non-disclosure policy*, that generalizes non-interference while allowing one to deal with declassification.

There is no constraint on using the declassification construct in [1], and this is in contrast with most other studies, which aim at restricting the use of such an operation (sometimes without justifying such constraints, for lack of a corresponding extensional notion of security). In this paper, we shall show that putting access control into the picture provides us with a natural way to restrict declassification, namely: *a program may only declassify information that it has the right to read.* To do this, we integrate into the language a formal approach to access control that has recently been introduced to deal with the "stack inspection" mechanism of JAVA security architecture [8,17,23]. That is, we add to the language of [1][1] the constructs for managing access control that are considered in these papers, namely: a construct ($\ell \ltimes M$) for restricting the access right of program M to be lower than ℓ, a dual construct ($\mathsf{enable}\ \ell\ \mathsf{in}\ M$) for granting read access level at least ℓ to M, and ($\mathsf{test}\ \ell\ \mathsf{then}\ M\ \mathsf{else}\ N$) for testing whether the access level ℓ is granted by the context or not, and behave accordingly. In these constructs, ℓ is a security level, that is a member of the lattice that is used for directing information flow. In particular, the security level at which a reference is classified also represents the right to access the information it contains, that is, to read the reference. For instance, in order to transfer information classified at level ℓ_1 (or lower) into level ℓ_2 (where $\ell_1 \not\preceq \ell_2$ in the security lattice), one may write, using notations that are explained in the next section:

$$(\mathsf{test}\ \ell_1\ \mathsf{then}\ (\mathsf{flow}\ \ell_1 \prec \ell_2\ \mathsf{in}\ x_{\ell_2} := \,!\, y_{\ell_1})\ \mathsf{else}\ ())$$

Our main contribution is as follows. We extend the type system of [1] to deal with the access control constructs, thus integrating both access control and information flow control into a single static analysis technique. A similar integration was previously done in [2], but for a different language, which in particular does not involve declassification. The combination of access control and information flow is also discussed in [16] (with some further references), in a purely functional setting, following an approach which does not seem compatible with a bisimulation-based notion of a secure program, like the non-disclosure policy. Our main results are, first, a type safety property showing that access control is indeed ensured by the type system, that is, a typable program never attempts to read a reference for which it would not be granted the appropriate reading clearance. (A similar type safety result was established in [17,23], but for a purely functional language, with no imperative feature, thus not dealing with access control in our sense, and without any consideration for information flow, and, a fortiori, declassification). Second, we extend the soundness result of [1], showing that secure information flow is ensured by our type system. In this way, we achieve the enforcement of "end-to-end confidentiality" in our language, while restricting declassification in a natural way.

Note. For lack of space, the proofs are omitted, or only briefly sketched.

2 The language

2.1 Security (pre-)Lattices

The security levels are hierarchically organized in a *pre-lattice*, a structure defined as a pair $(\mathcal{L}, \preceq)$, where $\preceq$ is a preorder relation over the set $\mathcal{L}$, that is a reflexive and transitive, but not necessarily symmetric relation, such that for any $x, y \in \mathcal{L}$ there exist a meet $x \curlywedge y$ and a join $x \curlyvee y$ satisfying:

$$\begin{array}{cc}
x \curlywedge y \preceq x & x \preceq x \curlyvee y \\
x \curlywedge y \preceq y & y \preceq x \curlyvee y \\
z \preceq x \;\&\; z \preceq y \;\Rightarrow\; z \preceq x \curlywedge y & x \preceq z \;\&\; y \preceq z \;\Rightarrow\; x \curlyvee y \preceq z
\end{array}$$

[1] We do not consider threads in this paper. They would not cause any technical difficulty, but complicate the definitions and proofs.

The pre-lattices we use are defined as follows. We assume given a set $\mathcal{P}$ of *principals*, ranged over by p, $q\ldots$ (From an access control perspective, these are also called *permissions* [2,8], or *privileges* [17,23], while a "principal" is a set of permissions.) A *confidentiality level* is any set of principals, that is any subset ℓ of $\mathcal{P}$. The intuition is that whenever ℓ is the confidentiality label of an object, i.e. a reference, it represents a set of programs that are allowed to get the value of the object, i.e. to read the reference. Then a confidentiality level is similar to an access-control list (i.e. a set of permissions). From this point of view, a reference labelled $\mathcal{P}$ (also denoted $\bot$) is the most public one – every program is allowed to read it – whereas the label $\emptyset$ (also denoted $\top$) indicates a secret reference, and reverse inclusion of security levels may be interpreted as indicating allowed flows of information: if a reference u is labelled ℓ, and $\ell \supseteq \ell'$ then the value of u may be transferred to a reference v labelled ℓ', since the programs allowed to read this value from v were already allowed to read it from u.

We follow the approach of [1], where declassification is achieved by dynamically updating the lattice structure of confidentiality levels, by the means of local flow policies. A *flow policy* is a binary relation over $\mathcal{P}$. We let F, $G\ldots$ range over such relations. A pair $(p,q) \in F$ is to be understood as "information may flow from principal p to principal q", that is, more precisely, "*everything that principal* p *is allowed to read may also be read by principal* q". As a member of a flow policy, a pair (p,q) will be written $p \prec q$. We denote, as usual, by F^* the preorder generated by F (that is, the reflexive and transitive closure of F). Any flow policy F determines a preorder on confidentiality levels that extends reverse inclusion, as follows:

$$\ell \preceq_F \ell' \quad \Leftrightarrow_{\text{def}} \quad \forall q \in \ell'.\ \exists p \in \ell.\ p\, F^*\, q$$

It is not difficult to see that the preorder $\preceq_F$ induces a pre-lattice structure on the set of confidentiality levels, where a meet is simply the union, and a join of ℓ and ℓ' is

$$\{\, q \mid \exists p \in \ell.\ \exists p' \in \ell'.\ p\, F^*\, q \ \&\ p'\, F^*\, q \,\}$$

This observation justifies the following definition.

Definition (Security Pre-Lattices) 2.1. *A* confidentiality level *is any subset ℓ of the set $\mathcal{P}$ of* principals. *Given a* flow policy $F \subseteq \mathcal{P}\times\mathcal{P}$, *the confidentiality levels are pre-ordered by the relation*

$$\ell \preceq_F \ell' \quad \Leftrightarrow_{\text{def}} \quad \forall q \in \ell'.\ \exists p \in \ell.\ p\, F^*\, q$$

The meet *and* join, *w.r.t. F, of two security levels ℓ and ℓ' are respectively given by $\ell \cup \ell'$ and*

$$\ell \curlyvee_F \ell' = \{\, q \mid \exists p \in \ell.\ \exists p' \in \ell'.\ p\, F^*\, q \ \&\ p'\, F^*\, q \,\}.$$

2.2 Syntax and Operational Semantics

The language we consider is a higher-order imperative language *à la* ML, extended with constructs for dynamically granting and testing access rights, as in [2,8,17,23], and a construct for introducing local flow policies, as in [1]. The

$$
\begin{array}{rll}
M,\ N\ldots \in \mathcal{E}xpr ::= & V \mid (\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ N') \mid (MN) & \textit{expressions}\\
\mid & M\,;N \mid (\mathsf{ref}_{\ell,\theta}\,N) \mid (!\,N) \mid (M := N) & \\
\mid & (\ell \ltimes M) \mid (\mathsf{enable}\ \ell\ \mathsf{in}\ M) \mid (\mathsf{test}\ \ell\ \mathsf{then}\ M\ \mathsf{else}\ N) & \\
\mid & (\mathsf{flow}\ F\ \mathsf{in}\ M) & \\
V \in \mathcal{V}al ::= & x \mid u_{\ell,\theta} \mid \mathsf{rec}f(x).M \mid tt \mid f\!f \mid () & \textit{values}
\end{array}
$$

Fig. 1. Syntax

construct $(\ell \ltimes M)$ is used to restrict the access right of M by ℓ (this is similar to the "framed" expressions of [8], and to the "signed" expressions of [17]). This is a scoping construct: the current reading clearance is restored after termination of M. Our $(\mathsf{enable}\ \ell\ \mathsf{in}\ M)$ construct is slightly different from the one of [17,23], where ℓ is restricted to be a singleton (our semantics is accordingly slightly more liberal). It is used to locally extend the read access right of M by ℓ. The test expression checks whether a given level is enabled by the current evaluation context. The local flow declaration $(\mathsf{flow}\ F\ \mathsf{in}\ M)$ enables the policy F to be used while reducing M, usually for declassification purposes. (For more comments on the syntax, we refer to [1,2,8,17].)

The syntax is given in Figure 1, where x and f are any variables, ℓ is any confidentiality level, and F is any flow policy. A *reference* is a memory location u to which a confidentiality level ℓ is assigned. For technical reasons, we also record the type θ (see Section 3 below) of the contents of the reference. We denote by $\mathsf{loc}(M)$ the set of decorated locations $u_{\ell,\theta}$ occurring in M. These references are regarded as providing the *inputs* of the expression M. We let $\mathsf{fv}(M)$ be the set of variables occurring free in M, and we denote by $\{x \mapsto V\}M$ the capture-avoiding susbtitution of V for the free occurrences of x in M, where $V \in \mathcal{V}al$. We may write $\mathsf{rec}f(x).M$ as λxM whenever $f \notin \mathsf{fv}(M)$. As usual, $(\mathsf{let}\ x = N\ \mathsf{in}\ M)$ denotes (λxMN).

The reduction relation is a transition relation between configurations of the form (M, μ) where μ, the *memory* (or *heap*), is a mapping from a finite set $\mathsf{dom}(\mu)$ of references to values. The operation of updating the value of a reference in the memory is denoted, as usual, $\mu[u_{\ell,\theta} := V]$. We say that the name u is *fresh for* μ if $v_{\ell,\theta} \in \mathsf{dom}(\mu) \;\Rightarrow\; v \neq u$. In what follows we shall only consider *well-formed* configurations, that is pairs (M, μ) such that $\mathsf{loc}(M) \subseteq \mathsf{dom}(\mu)$ and for any $u_{\ell,\theta} \in \mathsf{dom}(\mu)$ we have $\mathsf{loc}(\mu(u_{\ell,\theta})) \subseteq \mathsf{dom}(\mu)$ (this property will be preserved by the operational semantics). As usual (see [25]), the operational semantics consists in reducing a *redex* (reducible expression) inside an *evaluation context.* Reducible expressions are given by the following grammar:

$$
\begin{array}{rl}
R := & (\mathsf{if}\ tt\ \mathsf{then}\ M\ \mathsf{else}\ N) \mid (\mathsf{if}\ f\!f\ \mathsf{then}\ M\ \mathsf{else}\ N) \mid (\mathsf{rec}f(x).MV)\\
\mid & V\,;N \mid (\mathsf{ref}_{\ell,\theta}\,V) \mid (!\,u_{\ell,\theta}) \mid (u_{\ell,\theta} := V)\\
\mid & (\ell \ltimes V) \mid (\mathsf{enable}\ \ell\ \mathsf{in}\ V) \mid (\mathsf{test}\ \ell\ \mathsf{then}\ M\ \mathsf{else}\ N)\\
\mid & (\mathsf{flow}\ \ell\ \mathsf{in}\ V)
\end{array}
$$

$$
\begin{array}{rcll}
\ell \vdash_G (\mathbf{E}[(\mathsf{if}\ tt\ \mathsf{then}\ M\ \mathsf{else}\ N)], \mu) & \rightarrow & (\mathbf{E}[M], \mu) & \\
\ell \vdash_G (\mathbf{E}[(\mathsf{if}\ \mathit{ff}\ \mathsf{then}\ M\ \mathsf{else}\ N)], \mu) & \rightarrow & (\mathbf{E}[N], \mu) & \\
\ell \vdash_G (\mathbf{E}[(\mathsf{rec} f(x).MV)], \mu) & \rightarrow & (\mathbf{E}[\{x \mapsto V\}\{f \mapsto \mathsf{rec} f(x).M\}M], \mu) & \\
\ell \vdash_G (\mathbf{E}[V\,; N], \mu) & \rightarrow & (\mathbf{E}[N], \mu) & \\
\ell \vdash_G (\mathbf{E}[(\mathsf{ref}_{\ell',\theta}\, V)], \mu) & \rightarrow & (\mathbf{E}[u_{\ell',\theta}], \mu \cup \{u_{\ell',\theta} \mapsto V\}) & u \text{ fresh for } \mu \\
\ell \vdash_G (\mathbf{E}[(!\, u_{\ell',\theta})], \mu) & \rightarrow & (\mathbf{E}[V], \mu) & \mu(u_{\ell',\theta}) = V\ \& \\
& & & \ell' \preceq_G \lfloor \mathbf{E} \rfloor_\ell \\
\ell \vdash_G (\mathbf{E}[(u_{\ell',\theta} := V)], \mu) & \rightarrow & (\mathbf{E}[()], \mu[u_{\ell',\theta} := V]) & \\
\ell \vdash_G (\mathbf{E}[(\ell' \ltimes V)], \mu) & \rightarrow & (\mathbf{E}[V], \mu) & \\
\ell \vdash_G (\mathbf{E}[(\mathsf{enable}\ \ell'\ \mathsf{in}\ V)], \mu) & \rightarrow & (\mathbf{E}[V], \mu) & \\
\ell \vdash_G (\mathbf{E}[(\mathsf{test}\ \ell'\ \mathsf{then}\ M\ \mathsf{else}\ N)], \mu) & \rightarrow & (\mathbf{E}[M], \mu) & \ell' \preceq_G \lfloor \mathbf{E} \rfloor_\ell \\
\ell \vdash_G (\mathbf{E}[(\mathsf{test}\ \ell'\ \mathsf{then}\ M\ \mathsf{else}\ N)], \mu) & \rightarrow & (\mathbf{E}[N], \mu) & \ell' \not\preceq_G \lfloor \mathbf{E} \rfloor_\ell \\
(\mathbf{E}[(\mathsf{flow}\ F\ \mathsf{in}\ V)], \mu) & \rightarrow & (\mathbf{E}[V], \mu) &
\end{array}
$$

Fig. 2. Reduction

Evaluation contexts are given by:

$$
\begin{array}{lll}
\mathbf{E} ::= [\,] \mid \mathbf{E}[\mathbf{F}] & & \textit{evaluation contexts} \\
\mathbf{F} := (\mathsf{if}\ [\,]\ \mathsf{then}\ M\ \mathsf{else}\ N) \mid ([\,]N) \mid (V[\,]) & & \textit{frames} \\
\quad \mid [\,]\,; N \mid (\mathsf{ref}_{\ell,\theta}[\,]) \mid (!\,[\,]) \mid ([\,] := N) \mid (V := [\,]) & & \\
\quad \mid (\ell \ltimes [\,]) \mid (\mathsf{enable}\ \ell\ \mathsf{in}\ [\,]) \mid (\mathsf{flow}\ F\ \mathsf{in}\ [\,]) & &
\end{array}
$$

The operational semantics of an expression depends upon a given global flow policy G and a default confidentiality level ℓ, which represents the access right assigned to the evaluated expression – as if we had to evaluate $(\bot \ltimes (\mathsf{enable}\ \ell\ \mathsf{in}\ M))$. Then the statements defining the operational semantics have the form

$$\ell \vdash_G (M, \mu) \rightarrow (M', \mu')$$

Given a global flow policy G, for any confidentiality level ℓ representing the current access level, we define the level granted by the evaluation context $\mathbf{E}$, denoted $\lfloor \mathbf{E} \rfloor_\ell$, as follows:

$$\lfloor [\,] \rfloor_\ell = \ell$$

$$\lfloor \mathbf{E}[\mathbf{F}] \rfloor_\ell = \begin{cases} (\lfloor \mathbf{E} \rfloor_\ell) \curlywedge_G \ell' & \text{if } \mathbf{F} = (\ell' \ltimes [\,]) \\ (\lfloor \mathbf{E} \rfloor_\ell) \curlyvee_G \ell' & \text{if } \mathbf{F} = (\mathsf{enable}\ \ell'\ \mathsf{in}\ [\,]) \\ \lfloor \mathbf{E} \rfloor_\ell & \text{otherwise} \end{cases}$$

Computing $\lfloor \mathbf{E} \rfloor_\ell$ is a form of "stack inspection," see [8,17,23]. The reduction rules are given in Figure 2. They are fairly standard, as regards the functional and imperative fragment of the language. One may observe that $(\mathsf{flow}\ F\ \mathsf{in}\ M)$ behaves exactly as M, and that the semantics of the constructs for managing access rights are as one might expect, given the definition of $\lfloor \mathbf{E} \rfloor_\ell$ above. We denote by $\xrightarrow{*}$ the reflexive and transitive closure of the reduction relation. More precisely, we define:

$$\frac{}{\ell \vdash_G (M,\mu) \xrightarrow{*} (M,\mu)} \qquad \frac{\ell \vdash_G (M,\mu) \xrightarrow{*} (M'',\mu'') \quad \ell \vdash_G (M'',\mu'') \rightarrow (M',\mu')}{\ell \vdash_G (M,\mu) \xrightarrow{*} (M',\mu')}$$

An expression is said to *converge* if, regardless of the memory, its evaluation terminates on a value, that is:

$$M\Downarrow \quad \Leftrightarrow_{\text{def}} \quad \forall\mu\,\exists V \in \mathcal{V}al\,\exists\mu'.\ (M,\mu) \xrightarrow{*} (V,\mu')$$

One can see that a constraint that could block the reduction is the dynamic checking of read access rights, that is the condition $\ell' \preceq_G \lfloor \mathbf{E} \rfloor_\ell$ when reading a reference $u_{\ell',\theta}$ in the memory, with ℓ as the given reading clearance. (Since we are only considering well-formed configurations, the value $\mu(u_{\ell',\theta})$ is always defined.) Indeed for instance the expression (omitting the types attached to references)

$$(\mathsf{flow}\ H \prec L\ \mathsf{in}\ v_L := !\,u_H)$$

is blocked if the current access right is $\{L\}$, and can only proceed if this right is at least $\{H\}$. This indicates that, using ($\mathsf{flow}\ F\ \mathsf{in}\ M$), one can only declassify information that one has the right to read. This is because the local flow policy declarations do not interfere with access control, i.e. they do not play any role in the definition of $\lfloor \mathbf{E} \rfloor_\ell$. More generally, to let information flow, such as in $v_{\ell'} := !\,u_\ell$, a program must have the right to read it.

One can easily check the usual property (see [25]) that, given a pair (ℓ, G), a closed expression is either a value, or a reducible expression, or is a faulty expression, either for typing reasons, or because it does not have the appropriate access rights, that is:

Lemma and Definition 2.2. *Let M be a closed expression. Then, for any ℓ and G, either $M \in \mathcal{V}al$, or for any μ there exist F, M' and μ' such that $\ell \vdash_G (M,\mu) \underset{F}{\rightarrow} (M',\mu')$, or M is (ℓ, G)-*faulty*, that is:*

(i) *$M = \mathbf{E}[(\mathsf{if}\ V\ \mathsf{then}\ N\ \mathsf{else}\ N')]$ and $V \notin \{tt, f\!f\}$, or*

(ii) *$M = \mathbf{E}[(VV')]$ and V is not a functional value $\mathsf{rec}f(x).M'$, or*

(iii) *$M = \mathbf{E}[(!\,V)]$ or $M = \mathbf{E}[(V := V')]$ where V is not a reference $u_{\ell',\theta}$, or*

(iv) *$M = \mathbf{E}[(!\,u_{\ell',\theta})]$ with $\ell' \not\preceq_G \lfloor \mathbf{E} \rfloor_\ell$.*

Our main aim in this paper is to show that we can design a type system that guarantees both secure information flow, as in [1], and, as in [2,17], the fact that a well-typed expression is never stuck, and therefore that the run-time checking of the access rights is useless for such expressions.

3 The Type and Effect System

Our type system elaborates on the one of [1], and, as such, is actually a *type and effect system* [11]. This is consistent with our "*state-oriented*" approach – as opposed to the "value-oriented" approach of [8,16,17,23] for instance – where only the access to the "information containers", that is, to the references in the memory, is protected by access rights. In particular, a value is by itself neither "secret" nor "public," and the types do not need to be multiplied by the set of confidentiality levels. Then the types are

$$\tau,\ \sigma,\ \theta \ldots\ ::=\ t \mid \mathsf{bool} \mid \mathsf{unit} \mid \theta\,\mathsf{ref}_\ell \mid (\tau \xrightarrow[\ell,F]{s} \sigma)$$

where t is any type variable and s is any "security effect" – see below. Notice that a reference type $\theta\,\mathsf{ref}_\ell$ records the type θ of values the reference contains, as well as the "region" ℓ where it is created, which is the confidentiality level at which the reference is classified. Since a functional value wraps a possibly effectful computation, its type records this *latent effect* [11], which is the effect the function may have when applied to an argument. It also records the "latent reading clearance" ℓ and the "latent flow policy" F, which are assumed to hold when the function is called. The judgements of the type and effect system have the form

$$\ell; F; \Gamma \vdash_G M : s, \tau$$

where Γ is a typing context, assigning types to variables, and s is a security effect, that is a triple (ℓ_0, ℓ_1, ℓ_2) of confidentiality levels. The intuition is:

- ℓ is the current read access right that is in force when reducing M;
- F is the current flow policy, while G is the given global flow policy;
- ℓ_0, also denoted by $s.\mathsf{c}$, is the *confidentiality level* of M. This is an upper bound (up to the current flow relation) of the confidentiality levels of the references the expression M reads that may influence its resulting *value*;
- ℓ_1, also denoted $s.\mathsf{w}$, is the *writing effect*, that is a lower bound (w.r.t. the relation $\preceq$) of the level of references that the expression M may update;
- ℓ_2, also denoted $s.\mathsf{t}$, is an upper bound (w.r.t. the current flow relation) of the levels of the references the expression M reads that may influence its *termination*. We call this the *termination effect* of the expression.

In the following we shall denote $s.\mathsf{c} \curlyvee_{F \cup G} s.\mathsf{t}$ by $s.\mathsf{r}$, assuming that F and G are understood from the context. There is actually an implicit parameter in the type system, which is a set $\mathcal{T}$ of expressions that is used in the typing of conditional branching. The single property that we will assume about this set in our proof of type soundness is that it only contains strongly converging expressions, that is:

$$M \in \mathcal{T} \ \Rightarrow\ M{\Downarrow}$$

According to the intuition above, the security effects $s = (c, w, t)$ are ordered componentwise, in a covariant manner as regards the confidentiality level c and the termination effect t, and in a contravariant way as regard the writing effect w. Then we abusively denote by $\bot$ and $\top$ the triples $(\bot, \top, \bot)$ and $(\top, \bot, \top)$ respectively. In the typing rules for compound expressions, we will use the join operation on security effects:

$$s \curlyvee_F s' \ =_{\text{def}}\ (s.\mathsf{c} \curlyvee_F s'.\mathsf{c}, s.\mathsf{w} \cup s'.\mathsf{w}, s.\mathsf{t} \curlyvee_F s'.\mathsf{t})$$

as well as the following convention:

$$\frac{}{\ell;F;\Gamma\vdash_G u_{\ell',\theta}:\bot,\theta\,\mathsf{ref}_{\ell'}}\;(\textsc{Loc})\qquad\frac{}{\ell;F;\Gamma,x:\tau\vdash_G x:\bot,\tau}\;(\textsc{Var})$$

$$\frac{\ell;F;\Gamma,x:\tau,f:(\tau\xrightarrow[\ell,F]{s}\sigma)\vdash_G M:s,\sigma}{\ell';F';\Gamma\vdash_G \mathsf{rec}f(x).M:\bot,(\tau\xrightarrow[\ell,F]{s}\sigma)}\;(\textsc{Fun})\qquad\frac{}{\ell;F;\Gamma\vdash_G ():\bot,\mathsf{unit}}\;(\textsc{Nil})$$

$$\frac{}{\ell;F;\Gamma\vdash_G tt:\bot,\mathsf{bool}}\;(\textsc{BoolT})\qquad\frac{}{\ell;F;\Gamma\vdash_G f\!f:\bot,\mathsf{bool}}\;(\textsc{BoolF})$$

$$\frac{\ell;F;\Gamma\vdash_G M:s,\mathsf{bool}\quad \ell;F;\Gamma\vdash_G N_i:s_i,\tau\quad s.\mathsf{r}\preceq_{F\cup G}s_0.\mathsf{w}\cup s_1.\mathsf{w}}{\ell;F;\Gamma\vdash_G(\mathsf{if}\ M\ \mathsf{then}\ N_0\ \mathsf{else}\ N_1):s\curlyvee s_0\curlyvee s_1\curlyvee(\bot,\top,t),\tau}\;(\textsc{Cond})$$

where

$$t\;=\;\begin{cases}\bot\ \text{ if } N_0,N_1\in\mathcal{T}\\ s.\mathsf{c}\text{ otherwise}\end{cases}$$

$$\frac{\begin{array}{ll}\ell;F;\Gamma\vdash_G M:s,\tau\xrightarrow[\ell',F']{s'}\sigma & \ell'\preceq_G\ell\quad s.\mathsf{t}\preceq_{F\cup G}s''.\mathsf{w}\\ \ell;F;\Gamma\vdash_G N:s'',\tau & s.\mathsf{r}\curlyvee s''.\mathsf{r}\preceq_{F\cup G}s'.\mathsf{w}\end{array}}{\ell;F\cup F';\Gamma\vdash_G(MN):s\curlyvee s'\curlyvee s''\curlyvee(\bot,\top,s.\mathsf{c}\curlyvee s''.\mathsf{c}),\sigma}\;(\textsc{App})$$

$$\frac{\ell;F;\Gamma\vdash_G M:s,\tau\quad \ell;F;\Gamma\vdash_G N:s',\sigma\quad s.\mathsf{t}\preceq_{F\cup G}s'.\mathsf{w}}{\ell;F;\Gamma\vdash_G M\,;N:(\bot,s.\mathsf{w},s.\mathsf{t})\curlyvee s',\sigma}\;(\textsc{Seq})$$

$$\frac{\ell;F;\Gamma\vdash_G M:s,\theta\quad s.\mathsf{r}\preceq_{F\cup G}\ell'}{\ell;F;\Gamma\vdash_G(\mathsf{ref}_{\ell',\theta}\,M):(\bot,s.\mathsf{w},s.\mathsf{t}),\theta\,\mathsf{ref}_{\ell'}}\;(\textsc{Ref})\qquad\frac{\ell;F;\Gamma\vdash_G M:s,\theta\,\mathsf{ref}_{\ell'}\quad \ell'\preceq_G\ell}{\ell;F;\Gamma\vdash_G(!\,M):s\curlyvee(\ell',\top,\bot),\theta}\;(\textsc{Deref})$$

$$\frac{\begin{array}{ll}\ell;F;\Gamma\vdash_G M:s,\theta\,\mathsf{ref}_{\ell'} & s.\mathsf{t}\preceq_{F\cup G}s'.\mathsf{w}\\ \ell;F;\Gamma\vdash_G N:s',\theta & s.\mathsf{r}\curlyvee s'.\mathsf{r}\preceq_{F\cup G}\ell'\end{array}}{\ell;F;\Gamma\vdash_G(M:=N):(\bot,s.\mathsf{w}\cup s'.\mathsf{w}\cup\ell',s.\mathsf{t}\curlyvee s'.\mathsf{t}),\mathsf{unit}}\;(\textsc{Assign})$$

$$\frac{\ell\curlywedge_G\ell';F;\Gamma\vdash_G M:s,\tau}{\ell;F;\Gamma\vdash_G(\ell'\ltimes M):s,\tau}\;(\textsc{Restric})\qquad\frac{\ell\curlyvee_G\ell';F;\Gamma\vdash_G M:s,\tau}{\ell;F;\Gamma\vdash_G(\mathsf{enable}\ \ell'\ \mathsf{in}\ M):s,\tau}\;(\textsc{Enable})$$

$$\frac{\ell';F;\Gamma\vdash_G M:s,\tau\quad \ell;F;\Gamma\vdash_G N:s',\tau}{\ell;F;\Gamma\vdash_G(\mathsf{test}\ \ell'\ \mathsf{then}\ M\ \mathsf{else}\ N):s\curlyvee s',\tau}\;(\textsc{Test})$$

$$\frac{\ell;F\cup F';\Gamma\vdash_G M:s,\tau\quad s.\mathsf{c}\preceq_{F\cup F'\cup G}c\quad s.\mathsf{t}\preceq_{F\cup F'\cup G}t}{\ell;F;\Gamma\vdash_G(\mathsf{flow}\ F'\ \mathsf{in}\ M):(c,s.\mathsf{w},t),\tau}\;(\textsc{Flow})$$

Fig. 3. The Type and Effect System

Convention. *In the type system, when the security effects occurring in the context of a judgement $\ell;F;\Gamma\vdash_G M:s,\tau$ involve the join operation $\curlyvee$, it is assumed that the join is taken w.r.t. $F\cup G$, i.e. it is $\curlyvee_{F\cup G}$. We recall that by $s.\mathsf{r}$ we mean $s.\mathsf{c}\curlyvee_{F\cup G}s.\mathsf{t}$.*

The typing system is given in Figure 3. Notice that this system is syntax-directed: there is exactly one rule per construction of the language. In particular, there is no subtyping rule. As an example, one can see for instance that the expression

$$(\mathsf{test}\ \{p\}\ \mathsf{then}\ (\mathsf{flow}\ p \prec q\ \mathsf{in}\ v_{\{q\}} := \mathbf{!}\, u_{\{p\}})\ \mathsf{else}\ ())$$

(similar to the one given in the Introduction) is typable. One should also notice that our typing rules allow the programmer to make a dynamic use of the test construct, as in

$$\begin{array}{l}\mathsf{let}\ x = \lambda y(\mathsf{test}\ \ell\ \mathsf{then}\ v_\ell := \mathbf{!}\, u_\ell\ \mathsf{else}\ ())\ \mathsf{in} \\ (\mathsf{if}\ M\ \mathsf{then}\ (\mathsf{enable}\ \ell\ \mathsf{in}\ x())\ \mathsf{else}\ x())\end{array}$$

Compared to the system of [1], which does not involve constructs for managing access control, the main difference is in the (Deref) rule, where we have the constraint that the level of the reference that is read should be less than the access level granted by the context. Notice that this constraint only involves the global flow policy G, not the local one F. This is the way to ensure that declassification does not modify the access rights. There is a similar constraint in the rule for typing application, where the access level required by the (body of the) function should indeed be granted by the context. It is easy to see that typing enjoys a "weakening" property, asserting that if an expression is typable in the context of some access right ℓ, then it is also typable in the context of a more permissive reading clearance:

Lemma 3.1. *If $\ell; F; \Gamma \vdash_G M : s, \tau$ and $\ell \preceq_G \ell'$ then $\ell'; F; \Gamma \vdash_G M : s, \tau$.*

Proof: by induction on the inference of the typing judgemnent. ❏

Similarly, one could show that relaxing (that is, extending) the global or local flow policy does not affect typability.

4 Type Safety

The proof of type safety follows the usual steps [25]: we prove the Subject Reduction property, showing that typing is preserved along reductions, while the effects are decreasing; then we prove that faulty expressions are not typable. This, together with Lemma 2.2, will entail type safety. In the proof of the Subject Reduction property we use the following observation:

Remark 4.1. *For any value $V \in \mathcal{V}al$, if V is typable with type τ in the context Γ, then for any ℓ, F and G we have $\ell; F; \Gamma \vdash_G V : \bot, \tau$.*

We shall also need the following lemma, where we use the flow policy $\lceil \mathbf{E} \rceil$ granted by the evaluation context $\mathbf{E}$, which is defined as follows:

$$\begin{array}{l}\lceil [] \rceil = \emptyset \\ \lceil \mathbf{E}[\mathbf{F}] \rceil = \begin{cases} \lceil \mathbf{E} \rceil \cup F & \text{if } \mathbf{F} = (\mathsf{flow}\ F\ \mathsf{in}\ []) \\ \lceil \mathbf{E} \rceil & \text{otherwise} \end{cases}\end{array}$$

Lemma 4.2. *If $\ell; F; \Gamma \vdash_G \mathbf{E}[M] : s, \tau$, then there exist s_0 and σ such that $\lfloor \mathbf{E} \rfloor_\ell; F \cup \lceil \mathbf{E} \rceil; \Gamma \vdash_G M : s_0, \sigma$, and if $\lfloor \mathbf{E} \rfloor_\ell; F \cup \lceil \mathbf{E} \rceil; \Gamma \vdash_G N : s_1, \sigma$ with $s_1 \preceq_G s_0$ then $\ell; F; \Gamma \vdash_G \mathbf{E}[N] : s', \tau$ for some s' such that $s' \preceq_G s$.*

Proof: by induction on $\mathbf{E}$. ❏

Proposition (Subject Reduction) 4.3. *If $\ell; F; \Gamma \vdash_G M : s, \tau$ and $\ell \vdash_G (M, \mu) \rightarrow (M', \mu')$ with $u_{\ell,\theta} \in \mathsf{dom}(\mu) \Rightarrow \ell; F; \Gamma \vdash_G \mu(u_{\ell,\theta}) : \bot, \theta$ then $\ell; F; \Gamma \vdash_G M' : s', \tau$ for some s' such that $s' \preceq_G s$.*

Proof Sketch: we have $M = \mathbf{E}[R]$ and $M' = \mathbf{E}[N]$ with $(R, \mu) \rightarrow (N, \mu')$, where R is a redex. We proceed by induction on the context $\mathbf{E}$. In the case where $\mathbf{E} = []$, the proof is as usual for the functional and imperative part of the language (we need some auxiliary properties, see [25]). Let us just examine the cases where a security construct is involved. In the cases of

$$\begin{aligned} \ell \vdash_G ((\ell' \ltimes V), \mu) &\rightarrow (V, \mu) \\ \ell \vdash_G ((\mathsf{enable}\ \ell'\ \mathsf{in}\ V), \mu) &\rightarrow (V, \mu) \\ \ell \vdash_G ((\mathsf{flow}\ F\ \mathsf{in}\ V), \mu) &\rightarrow (V, \mu) \end{aligned}$$

we use the Remark 4.1 above. The cases

$$\begin{aligned} \ell \vdash_G ((\mathsf{test}\ \ell'\ \mathsf{then}\ M'\ \mathsf{else}\ M''), \mu) &\rightarrow (M', \mu) \quad \text{with } \ell' \preceq_G \ell \\ \ell \vdash_G ((\mathsf{test}\ \ell'\ \mathsf{then}\ M''\ \mathsf{else}\ M'), \mu) &\rightarrow (M', \mu) \quad \text{with } \ell' \npreceq_G \ell \end{aligned}$$

are immediate (in the first case we use Lemma 3.1). Now if $\mathbf{E} = \mathbf{E}'[\mathbf{F}]$ we use the Lemma 4.2 above. ❏

Now we show that the faulty expressions, as defined in Lemma and Definition 2.2, are not typable.

Lemma 4.4. *The (ℓ, G)-faulty expressions are not typable in the context of access right ℓ and global flow policy G.*

Proof: let $M = \mathbf{E}[(!\, u_{\ell',\theta})]$ with $\ell' \npreceq_G \lfloor \mathbf{E} \rfloor_\ell$, and assume that $\ell; F, \Gamma \vdash_G M : s, \tau$. Then by Lemma 4.2 one would have $\lfloor \mathbf{E} \rfloor_\ell; F \cup \lceil \mathbf{E} \rceil; \Gamma \vdash_G (!\, u_{\ell',\theta}) : s', \sigma$ for some s' and σ, but this is only possible, by the (Deref) rule, if $\ell' \preceq_G \lfloor \mathbf{E} \rfloor_\ell$, a contradiction. The other cases are standard. ❏

An immediate consequence of these results and Lemma 2.2 is:

Theorem (Type Safety) 4.5. *Let M be a typable closed expression, with $\ell; F; \Gamma \vdash_G M : s, \tau$, and let μ be such that $u_{\ell,\theta} \in \mathsf{dom}(\mu) \Rightarrow \ell; F; \Gamma \vdash_G \mu(u_{\ell,\theta}) : \bot, \theta$. Then either the reduction of (M, μ) with respect to (ℓ, G) does not terminate, or there exist a value $V \in \mathcal{V}al$ and a memory μ' such that $\ell \vdash_G (M, \mu) \xrightarrow{*} (V, \mu')$ with $\ell; F; \Gamma \vdash_G V : \bot, \tau$.*

In particular, this shows that the dynamic checking of the reading clearance (by means of a "stack inspection" mechanism) is actually not needed regarding a typable program, which never attempts to access a reference for which it would not have the appropriate access right.

5 Secure Information Flow

In [1], the authors have proposed a generalization of the usual non-interference property that allows one to deal with declassification. The idea is to define a program as secure if it satisfies a "local non-interference" property, called the *non-disclosure policy*, which roughly states that the program is, at each step of its execution, non-interfering with respect to the current flow policy, that is the global flow policy extended by the one granted by the evaluation context. Technically, a program will be considered as secure if it is bisimilar to itself. As usual, this property relies on preserving the "low equality" of memory. Roughly speaking, two memories are equal up to level ℓ if they assign the same value to every location with security level lower than ℓ, with respect to a given flow policy. In order to deal with reference creation, we only compare memories on the domain of references they share – then the "low equality" of memory is actually not an equivalence (it is not transitive). Nevertheless, we keep the standard terminology. The "low equality" of memories, with respect to a current flow policy F, and to a security level ℓ regarded as "low," is thus defined:

$$\mu \doteq^{F,\ell} \nu \quad \Leftrightarrow_{\text{def}} \quad \forall u_{\ell',\theta} \in \mathsf{dom}(\mu) \cap \mathsf{dom}(\nu).\ \ell' \preceq_F \ell \ \Rightarrow\ \mu(u_{\ell',\theta}) = \nu(u_{\ell',\theta})$$

From an information flow point of view, the notion of a secure program actually depends on the default access right. For instance, the assignment $v_{\ell'} := \mathbf{!}u_\ell$ where $\ell \not\preceq_G \ell'$, which is usually taken as a typical example of an unsecure program, is indeed secure (in the sense of [6]) in the context of a default access level ℓ'' such that $\ell \not\preceq_G \ell''$ (but in that case this program attempts a confidentiality violation, as regards access control). Then our definition of secure programs is parameterized by a default access level, and, as in [1], by a global flow policy.

Definition (Bisimulation) 5.1. *A* (ℓ, G, ℓ')-bisimulation *is a symmetric relation $\mathcal{R}$ on expressions such that if $M \mathcal{R} N$ and $\ell \vdash_G (M, \mu) \to (M', \mu')$ with $M = \mathbf{E}[R]$ where R is a redex, and if ν is such that $\mu \doteq^{G \cup \lceil \mathbf{E} \rceil, \ell'} \nu$ and $u_{\ell'',\theta} \in \mathsf{dom}(\mu' - \mu)$ implies that u is fresh for ν, then there exist N' and ν' such that $\ell \vdash_G (N, \nu) \xrightarrow{*} (N', \nu')$ with $M' \mathcal{R} N'$ and $\mu' \doteq^{G,\ell'} \nu'$.*

Remarks and Notation 5.2.

(i) *For any ℓ, G and ℓ' there exists a (ℓ, G, ℓ')-bisimulation, like for instance the set $\mathcal{V}al \times \mathcal{V}al$ of pairs of values.*

(ii) *The union of a family of (ℓ, G, ℓ')-bisimulations is a (ℓ, G, ℓ')-bisimulation. Consequently, there is a largest (ℓ, G, ℓ')-bisimulation, which we denote $\approxeq^{\ell,G,\ell'}$. This is the union of all such bisimulations.*

One should observe that the relation $\approxeq^{\ell,G,\ell'}$ is not reflexive. Indeed, a process which is not bisimilar to itself, like $v_{\ell'} := \mathbf{!}\,u_{\ell''}$ where $\ell \not\preceq_G \ell'' \preceq_G \ell$, is not secure. As in [20], our definition states that a program is secure, with respect to a default access level and a given global flow policy, if it is bisimilar to itself:

Definition (The Non-Disclosure Policy) 5.3. *A process P satisfies the* non-disclosure policy *(or is* secure *from the confidentiality point of view) with respect to the default access level ℓ and the global flow policy G if it satisfies $P \approxeq^{\ell,G,\ell'} P$ for all ℓ'. We then write $P \in \mathcal{ND}(\ell, G)$.*

For explanations and examples regarding the non-disclosure policy, we refer to [1]. Our second main result is that the type system guarantees secure information flow, whatever the default access right is:

Theorem (Soundness). *If M is typable in the context of the access level ℓ, a global flow policy G and a local policy F, that is if for some Γ, s and τ we have $\ell; F; \Gamma \vdash_G M : s, \tau$, then M satisfies the non-disclosure policy with respect to $F \cup G$, that is $M \in \mathcal{ND}(\ell', F \cup G)$, for all ℓ'.*

The proof of this result is very similar to the one of Type Soundness in the revised version of [1]. As usual with bisimulation proofs, one has to find an appropriate candidate relation, that contains the pairs (M, M) of typable expressions, and which is closed with respect to the co-inductive property. The constructs for managing access control do not add much complexity to this proof.

6 Conclusion

We have shown a way of integrating access control and information flow control in the setting of a high-level programming language, involving a declassification construct. Our "state-oriented" approach, that we share with [2], differs from the "value-oriented" approach (that one has to adopt when dealing with purely functional languages) that is followed in [8,17,23] to deal with stack inspection, and [16,22] as regards information flow control (see also [18] for further references). We think that assuming that confidentiality levels are assigned to "information containers" is more in line with the usual way of dealing with confidentiality than assigning security levels to values, like boolean *tt* and *ff*, integers or functions. In this way, the safety property guaranteed by access control is quite simple and natural, and this also provides a natural restriction on the use of declassification, and, more generally, information flow.

References

1. Almeida Matos, A., Boudol, G.: On declassification and the non-disclosure policy. In: CSFW'05, pp. 226–240 (2005) Revised version accepted for publication in the J. of Computer Security, available from the authors web page.
2. Banerjee, A., Naumann, D.A.: Stack-based access control for secure information flow. J. of Functional Programming. special issue on Language-Based Security 15, 131–177 (2005)
3. Boudol, G.: On typing information flow. In: Van Hung, D., Wirsing, M. (eds.) ICTAC 2005. LNCS, vol. 3722, pp. 366–380. Springer, Heidelberg (2005)
4. Broberg, N., Sands, D.: Flow locks: towards a core calculus for dynamic flow policies. In: Sestoft, P. (ed.) ESOP 2006 and ETAPS 2006. LNCS, vol. 3924, pp. 180–196. Springer, Heidelberg (2006)
5. Chong, S., Myers, A.C.: Security policies for downgrading. In: 11th ACM Conf. on Computer and Communications Security (2004)
6. Cohen, E.: Information transmission in computational systems. In: 6th ACM Symp. on Operating Systems Principles, pp. 133–139 (1977)
7. Denning, D.E.: A lattice model of secure information flow. CACM 19(5), 236–243 (1976)
8. Fournet, C., Gordon, A.: Stack inspection: theory and variants. In: POPL'02, pp. 307–318 (2002)

9. Goguen, J.A., Meseguer, J.: Security policies and security models. In: IEEE Symp. on Security and Privacy, pp. 11–20 (1982)
10. Lampson, B.W.: A note on the confinement problem. CACM 16(10), 613–615 (1973)
11. Lucassen, J.M., Gifford, D.K.: Polymorphic effect systems. In: POPL'88, pp. 47–57 (1988)
12. Li, P., Zdancewic, S.: Downgrading policies and relaxed noninterference. In: POPL'05, pp. 158–170 (2005)
13. Myers, A.: JFlow: practical mostly-static information flow control. In: POPL'99 (1999)
14. Myers, A.C., Liskov, B.: A decentralized model for information flow control. In: ACM Symp. on Operating Systems Principles, pp. 129–142 (1997)
15. Myers, A.C., Sabelfeld, A., Zdancewic, S.: Enforcing robust declassification and qualified robustness. J. of Computer Security 14(2), 157–196 (2006)
16. Pottier, F., Conchon, S.: Information flow inference for free. In: ICFP'00, pp. 46–57 (2000)
17. Pottier, F., Skalka, C., Smith, S.: A systematic approach to static access control. ACM TOPLAS 27(2), 344–382 (2005)
18. Sabelfeld, A., Myers, A.C.: Language-based information-flow security. IEEE J. on Selected Areas in Communications 21(1), 5–19 (2003)
19. Sabelfeld, A., Myers, A.C.: A model for delimited information release. In: Futatsugi, K., Mizoguchi, F., Yonezaki, N. (eds.) ISSS 2003. LNCS, vol. 3233, Springer, Heidelberg (2004)
20. Sabelfeld, A., Sands, D.: Probabilistic noninterference for multi-threaded programs. In: CSFW'00 (2000)
21. Sabelfeld, A., Sands, D.: Dimensions and principles of declassification. In: CSFW'05, pp. 255–269 (2005)
22. Simonet, V.: The Flow Caml system: documentation and user's manual INRIA Tech. Rep. 0282 (2003)
23. Skalka, C., Smith, S.: Static enforcement of security with types. In: ICFP'00, pp. 34–45 (2000)
24. Volpano, D., Smith, G., Irvine, C.: A sound type system for secure flow analysis. J. of Computer Security 4(3), 167–187 (1996)
25. Wright, A., Felleisen, M.: A syntactic approach to type soundness. Information and Computation 115(1), 38–94 (1994)
26. Zdancewic, S.: Challenges for information-flow security. In: PLID'04 (2004)

Reasoning About Delegation and Account Access in Retail Payment Systems

Shiu-Kai Chin and Susan Older

EECS Department, Syracuse University, Syracuse, New York 13244, USA

Abstract. Delegation and trust are essential to the smooth operation of large, geographically distributed systems, such as the US electronic retail payment system. This system supports billions of electronic transactions—from routine banking and store purchases to electronic commerce on the Internet. Because such systems provide the electronic fabric of our networked information society, it is crucial to understand rigorously and precisely the basis for the delegation and trust relationships in them. In this paper, we use a modal logic for access control to analyze these relationships in the context of checks (and their electronic equivalents) as payment instruments. While not free from risk, the retail payment system effectively balances trust, delegation, and risk on billions of transactions. Our logic allows us to explore with rigor the details of trust, delegation, and risk in these transactions.

Keywords: Access control, delegation, trust, retail payment systems, modal logic.

1 Introduction

You may be deceived if you trust too much, but you will live in torment if you don't trust enough.—Frank Crane

Trust—by which we mean the willingness to adopt someone else's beliefs as one's own—is central to the operation of the electronic retail payment system in the US. Trusted delegates operate on behalf of consumers, banks, and financial networks throughout the retail payment system, which handles many billions of transactions yearly.

Systems such as the retail payment system exemplify critical systems that are large, geographically distributed, part of critical infrastructure, and widely used. The electronic retail payment system in particular uses delegation extensively and depends on trust relationships each and every step of the way. Systems engineers who build such systems are ultimately responsible for assuring that the system behaves securely—in this case allowing account access to only those who should have access. Providing this assurance means engineers need formal tools to describe and analyze trust, delegation, and access policies to derive and justify access-control decisions. These tools ideally are both simple and effective. An engineering artifact will often produce an effect that is not precisely understood and demands scientific analysis. In the case of large distributed systems, the retail payment system largely works in a trustworthy fashion. Our objective is to describe and analyze the retail payment system so that we know precisely why it works.

In this paper, we incorporate a formal accounting of delegation into a modal logic for access control, described initially in [1,2] and based on the work of Lampson, Abadi, Burrows and colleagues [3,4]. The extension itself is quite simple, but it seems to work well for analyzing fairly complicated systems. To demonstrate its suitability for use by engineers, we use the logic to formally explicate the policies, delegations, and trust assumptions on which the United States' Automated Clearing House (ACH) network [5] depends. The ACH network is used by depository financial institutions (e.g., banks) to settle large numbers of financial transactions on a daily basis. These transactions are handled electronically according to the rules of the ACH network, using *check truncation*: the physical checks are removed from the payment-processing process, and instead, information from checks is transmitted electronically via the ACH network. The original checks are usually destroyed shortly after they are received by banks. As mandated by the Check 21 Act [6], the substitute electronic checks have the same legal standing as the original paper checks.

The remainder of this paper is organized as follows. In Section 2 we describe our logic for reasoning about access control and introduce our extension for delegation. Section 3 presents how electronic checks are used in the ACH network. We conclude in Section 4.

2 A Logic for Reasoning About Access Control

To keep this paper self-contained, we provide a brief introduction to the access-control logic described in detail in [2,1]. In Section 2.4, we introduce a minor extension to capture delegation and its essential properties.

2.1 Overview of the Logic

Principal Expressions. We start out by introducing a collection of principal expressions, ranged over by P and Q. Letting A range over a countable set of simple principal names, the abstract syntax of principal expressions is given as follows:

$$P ::= A \ / \ P\&Q \ / \ P \mid Q$$

The principal $P\&Q$ ("P in conjunction with Q") represents an abstract principal who makes exactly those statements made by both P and Q; $P \mid Q$ ("P quoting Q") represents an abstract principal corresponding to principal P quoting principal Q.

Access Control Statements. The abstract syntax of statements (ranged over by φ) is defined as follows, where P and Q range over principal expressions and p ranges over a countable set of *propositional variables*:

$$\varphi ::= p \;/\; \neg\varphi \;/\; \varphi_1 \wedge \varphi_2 \;/\; \varphi_1 \vee \varphi_2 \;/\; \varphi_1 \supset \varphi_2 \;/\; \varphi_1 \equiv \varphi_2 \;/$$
$$P \Rightarrow Q \;/\; P \textsf{ says } \varphi \;/\; P \textsf{ controls } \varphi \;/\; P \textsf{ reps } Q \textsf{ on } \varphi$$

Informally, a formula $P \Rightarrow Q$ (pronounced "P speaks for Q") indicates that *every* statement made by P can also be viewed as a statement from Q. A formula P controls φ is syntactic sugar for the implication (P says φ) $\supset \varphi$: in effect, P is a trusted authority with respect to the statement φ. P reps Q on φ denotes that P is Q's delegate on φ; it is syntactic sugar for (P says (Q says φ)) $\supset Q$ says φ. Notice that the definition of P reps Q on φ is a special case of controls and in effect asserts that P is a trusted authority with respect to Q saying φ.

2.2 Semantics

The semantics of formulas is based on Kripke structures, as given by the following definitions.

Definition 1. *A* Kripke structure $\mathcal{M}$ *is a three-tuple* $\langle W, I, J\rangle$*, where:*

- W *is a nonempty set, whose elements are called* worlds*.*
- I : ***PropVar*** $\rightarrow \mathcal{P}(W)$ *is an* interpretation *function that maps each propositional variable p to a set of worlds.*
- J : ***PName*** $\rightarrow \mathcal{P}(W \times W)$ *is a function that maps each principal name A to a relation on worlds (i.e., a subset of* $W \times W$*).*

We extend J to work over arbitrary *principal expressions* using set union and relational composition as follows:

$$J(P\&Q) = J(P) \cup J(Q)$$
$$J(P \mid Q) = J(P) \circ J(Q),$$

where

$$J(P) \circ J(Q) = \{(w_1, w_2) \mid \exists w'.(w_1, w') \in J(P) \text{ and } (w', w_2) \in J(Q)\}$$

Definition 2. *Each Kripke structure* $\mathcal{M} = \langle W, I, J\rangle$ *gives rise to a function*

$$\mathcal{E}_{\mathcal{M}}[\![-]\!] : \textbf{\textit{Form}} \rightarrow \mathcal{P}(W),$$

where $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is the set of worlds in which* φ *is considered true.* $\mathcal{E}_{\mathcal{M}}[\![\varphi]\!]$ *is defined inductively on the structure of* φ*, as shown in Figure 1.*

Note that, in the definition of $\mathcal{E}_{\mathcal{M}}[\![P \textsf{ says } \varphi]\!]$*,* $J(P)(w)$ *is simply the image of world* w *under the relation* $J(P)$*.*

2.3 Inference Rules

The semantic functions $\mathcal{E}_{\mathcal{M}}$ provide a fully defined and fully disclosed interpretation for the formulas of the logic. This mathematical foundation enables us

$$\begin{aligned}
\mathcal{E}_{\mathcal{M}}[\![p]\!] &= I(p) \\
\mathcal{E}_{\mathcal{M}}[\![\neg\varphi]\!] &= W - \mathcal{E}_{\mathcal{M}}[\![\varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \wedge \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \vee \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!] \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] &= (W - \mathcal{E}_{\mathcal{M}}[\![\varphi_1]\!]) \cup \mathcal{E}_{\mathcal{M}}[\![\varphi_2]\!] \\
\mathcal{E}_{\mathcal{M}}[\![\varphi_1 \equiv \varphi_2]\!] &= \mathcal{E}_{\mathcal{M}}[\![\varphi_1 \supset \varphi_2]\!] \cap \mathcal{E}_{\mathcal{M}}[\![\varphi_2 \supset \varphi_1]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \Rightarrow Q]\!] &= \begin{cases} W, & \text{if } J(Q) \subseteq J(P) \\ \emptyset, & \text{otherwise} \end{cases} \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ says } \varphi]\!] &= \{w | J(P)(w) \subseteq \mathcal{E}_{\mathcal{M}}[\![\varphi]\!]\} \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ controls } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![(P \text{ says } \varphi) \supset \varphi]\!] \\
\mathcal{E}_{\mathcal{M}}[\![P \text{ reps } Q \text{ on } \varphi]\!] &= \mathcal{E}_{\mathcal{M}}[\![P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi]\!]
\end{aligned}$$

Fig. 1. Semantics

to provide a means to reason about access-control situations using a small core collection of sound inference rules and syntactic-sugar definitions (see Figure 2), along with a larger set of rules that can be formally derived from the core rules (see Figure 3 for a sample set of derived rules that we have found particularly useful).

A rule of form $\dfrac{H_1 \cdots H_n}{C}$ is sound provided that, *for all Kripke structures* $\mathcal{M} = \langle W, I, J\rangle$, *if* $\mathcal{E}_{\mathcal{M}}[\![H_i]\!] = W$ *for each* $i \in \{1, \ldots, n\}$, *then* $\mathcal{E}_{\mathcal{M}}[\![C]\!] = W$. The rules in Figures 2 and 3 are all sound, and become the basis for reasoning about access-control decisions. The Kripke structures are then only necessary if one wishes to add new inference rules and to verify their soundness.

2.4 Delegation and Its Properties

Delegation is an important relationship in networks where decisions and authorities are distributed in different locations. When principal P acts on behalf of principal Q, we say that P is Q's delegate. P acting as Q's delegate on the statement φ, denoted by P reps Q on φ, is defined as syntactic sugar:

$$P \text{ reps } Q \text{ on } \varphi \overset{\text{def}}{=} (P \text{ says } (Q \text{ says } \varphi)) \supset Q \text{ says } \varphi$$

Essentially, if P is Q's representative on statements φ, then P claiming Q has said φ is treated as if Q said φ herself.

There are three crucial properties of the delegation relationship that our logic (or any other formal system) must accurately capture:

1. A recognized delegate should in fact have the authority to act on behalf of the principals they represent. That is, if a given policy allows principals to delegate to others and recognizes that Bob is Alice's delegate, then Bob should be able to act on Alice's behalf.

$$\textit{Taut} \quad \frac{}{\varphi} \quad \text{if } \varphi \text{ is an instance of a prop-logic tautology}$$

$$\textit{Modus Ponens} \quad \frac{\varphi \quad \varphi \supset \varphi'}{\varphi'} \qquad \textit{Says} \quad \frac{\varphi}{P \text{ says } \varphi}$$

$$\textit{MP Says} \quad \frac{}{(P \text{ says } (\varphi \supset \varphi')) \supset (P \text{ says } \varphi \supset P \text{ says } \varphi')}$$

$$\textit{Speaks For} \quad \frac{}{P \Rightarrow Q \supset (P \text{ says } \varphi \supset Q \text{ says } \varphi)}$$

$$\textit{Quoting} \quad \frac{}{P \mid Q \text{ says } \varphi \equiv P \text{ says } Q \text{ says } \varphi}$$

$$\textit{\&Says} \quad \frac{}{P\&Q \text{ says } \varphi \equiv P \text{ says } \varphi \wedge Q \text{ says } \varphi}$$

$$\textit{Idempotency of} \Rightarrow \quad \frac{}{P \Rightarrow P} \qquad \textit{Monotonicity of} \mid \quad \frac{P' \Rightarrow P \quad Q' \Rightarrow Q}{P' \mid Q' \Rightarrow P \mid Q}$$

$$\textit{Associativity of} \mid \quad \frac{P \mid (Q \mid R) \text{ says } \varphi}{(P \mid Q) \mid R \text{ says } \varphi}$$

$$P \text{ controls } \varphi \quad \stackrel{\text{def}}{=} \quad (P \text{ says } \varphi) \supset \varphi$$

$$P \text{ reps } Q \text{ on } \varphi \stackrel{\text{def}}{=} P \mid Q \text{ says } \varphi \supset Q \text{ says } \varphi$$

Fig. 2. Core Inference Rules

2. Delegates generally should not be able to restrict the scope of their duties as a principal's representative. For example, suppose that Alice delegates to Bob the task of withdrawing \$500 from her checking account and depositing it to her savings account. Bob should not be able to withdraw the funds without also depositing them; to do so would be a violation of his responsibilities, not to mention theft.
3. The delegation relationship generally is not transitive: a delegate should not be able to pass on his responsibilities to someone else.

The first property—that recognized delegates should be able to act on behalf of the principals they represent—is reflected by the *Reps* rule stated below and in Figure 3:

$$\textit{Reps} \quad \frac{Q \text{ controls } \varphi \quad P \text{ reps } Q \text{ on } \varphi \quad P \mid Q \text{ says } \varphi}{\varphi}$$

This rule states that if Q is authorized to perform φ, P is recognized as Q's delegate on φ, and P requests φ on Q's behalf, then the request for φ should be

$$\textit{Conjunction} \quad \frac{\varphi_1 \quad \varphi_2}{\varphi_1 \wedge \varphi_2}$$

$$\textit{Simplification (1)} \quad \frac{\varphi_1 \wedge \varphi_2}{\varphi_1} \qquad \textit{Simplification (2)} \quad \frac{\varphi_1 \wedge \varphi_2}{\varphi_2}$$

$$\textit{Quoting (1)} \quad \frac{P \mid Q \textsf{ says } \varphi}{P \textsf{ says } Q \textsf{ says } \varphi} \qquad \textit{Quoting (2)} \quad \frac{P \textsf{ says } Q \textsf{ says } \varphi}{P \mid Q \textsf{ says } \varphi}$$

$$\textit{\&Says (1)} \quad \frac{P\&Q \textsf{ says } \varphi}{P \textsf{ says } \varphi \wedge Q \textsf{ says } \varphi} \qquad \textit{\&Says (2)} \quad \frac{P \textsf{ says } \varphi \wedge Q \textsf{ says } \varphi}{P\&Q \textsf{ says } \varphi}$$

$$\textit{Controls} \quad \frac{P \textsf{ controls } \varphi \quad P \textsf{ says } \varphi}{\varphi} \qquad \textit{Derived Speaks For} \quad \frac{P \Rightarrow Q \quad P \textsf{ says } \varphi}{Q \textsf{ says } \varphi}$$

$$\textit{Reps} \quad \frac{Q \textsf{ controls } \varphi \quad P \textsf{ reps } Q \textsf{ on } \varphi \quad P \mid Q \textsf{ says } \varphi}{\varphi}$$

$$\textit{Rep Says} \quad \frac{P \textsf{ reps } Q \textsf{ on } \varphi \quad P \mid Q \textsf{ says } \varphi}{Q \textsf{ says } \varphi}$$

Fig. 3. Derived Rules Used in this Paper

granted. This rule can be derived from the sound rules of Figure 2 and thus is sound itself.

The second and third properties both state things that should **not** happen. For that reason, it is necessary to verify that our definition of delegation **prohibits** the reduction or passing on of delegation duties. The following two rules, which would allow the undesired behavior, can easily be shown to be unsound with respect to the Kripke semantics:

$$\textit{Unsound Rule!} \quad \frac{P \textsf{ reps } Q \textsf{ on } \varphi_1 \wedge \varphi_2}{P \textsf{ reps } Q \textsf{ on } \varphi_1}$$

$$\textit{Unsound Rule!} \quad \frac{P \textsf{ reps } Q \textsf{ on } \varphi \quad Q \textsf{ reps } R \textsf{ on } \varphi}{P \textsf{ reps } R \textsf{ on } \varphi}$$

The original papers by Lampson, Abadi and colleagues [3,4] introduced a notion of delegation based on a "fictional delegation server", whose purpose was to co-sign or back up any of a delegate's authentic requests. Their notion of delegation was a universal one: if Bob is Alice's delegate, then Bob represents Alice on *all* statements, not only on specifically designated ones. Furthermore, access policies had to specifically name delegates in addition to the principals they represented (e.g., *Bob* for *Alice* controls φ).

Our definition of delegation is a much simpler one, but we believe that the soundness of the *Reps* rule and the lack of soundness of the two undesired rules provide supporting evidence of its correctness. The next section demonstrates its suitability for reasoning about fairly complicated situations, such as how checks work as payment instruments within the retail payment system.

3 Checking Using an Electronic Clearing House Network

In this section we describe a banking network that uses electronic credits and debits and the Automated Clearing House (ACH) network—a trusted third-party settlement service. Detailed descriptions of retail payment systems and the ACH network can be found in the Federal Financial Institutions Examination Council's handbook on Retail Payment Systems [7] and the National Automated Clearing House Association's guide to rules and regulations governing the ACH network[5].

We pay particular attention to patterns of description and patterns of reasoning. The patterns of description take the form of definitions—typically formal definitions of financial instruments, statements of jurisdiction, and policy statements. Patterns of reasoning take the form of derived (and thus inherently sound) inference rules that reflect the implicit access-control decisions being made in the retail payment system. We include the formal proofs that justify the access-control decisions being made. These proofs are simple and show explicitly the trust relationships upon which the decisions are being made. We believe the combination of simplicity, precision, clarity, and soundness are of tremendous benefit to systems engineers and certifiers who build and evaluate complicated systems where delegation is widely used.

As we will be focusing on how checks and endorsed checks are used, we give them formal definitions. We adopt the notational convention that atomic (indivisible) actions are surrounded by "$\langle$" and "$\rangle$". For example, $\langle \textit{debit \$100}, \textit{acct}_{Alice}\rangle$ is interpreted as the atomic proposition "*it would be a good idea to debit Alice's account by \$100.*" To save space, we also adopt the notational abbreviation P controls+says φ to denote the *two* statements: P controls φ and P says φ. We use P controls+says φ in describing policies, but use both P controls φ and P says φ in formal proofs.

Definition 3. *A check is an order from a principal (known as the payer) upon a bank to draw upon the payer's deposit of funds to pay a certain amount of money to another principal (known as the payee). If P is the payer and Q is the payee, we represent a check written by P to Q as follows:*

$$\textit{Signature}_P \textsf{ says } (\langle \textit{Pay} \text{ amt}, Q\rangle \wedge \langle \textit{debit} \text{ amt}, \textit{acct}_P\rangle)$$

The check associates P's *signature* with the statement to pay Q and to debit P's account. As our subsequent analysis will show, one must be able to associate statements made by $\textit{Signature}_P$ with P; this association is represented in the logic as $\textit{Signature}_P \Rightarrow P$.

Definition 4. *A check is* endorsed *when the payee signs the check issued to him or her. If P is the payer and Q is the payee, we represent a check written by P and endorsed by Q as follows:*

$$\textit{Signature}_Q \mid \textit{Signature}_P \textsf{ says } (\langle \textit{Pay} \text{ amt}, Q\rangle \wedge \langle \textit{debit} \text{ amt}, \textit{acct}_P\rangle)$$

The banking system uses clearing houses (or clearing corporations) to collect and settle individual transactions while minimizing the payments made between banks.

Definition 5. *[7] defines a* clearing corporation *as follows:*

> *A central processing mechanism whereby members agree to net, clear, and settle transactions involving financial instruments. Clearing corporations fulfill one or all of the following functions:*
> - *nets many trades so that the number and the amount of payments that have to be made are minimized,*
> - *determines money obligations among traders, and*
> - *guarantees that trades will go through by legally assuming the risk of payments not made or securities not delivered. This latter function is what is implied when it is stated that the clearing corporation becomes the "counter-party" to all trades entered into its system. Also known as a clearinghouse or clearinghouse association.*

To understand how a clearing house works, suppose that some depositors of $Bank_P$ and $Bank_Q$ exchange a total of two checks as follows during the day:

1. Bob, a $Bank_Q$ depositor deposits a \$100 check from Alice, a $Bank_P$ depositor.
2. Dan, a $Bank_P$ depositor deposits a \$250 check from Carol, a $Bank_Q$ depositor.

$Bank_P$ and $Bank_Q$ send the deposited checks to a clearing house to total up the transactions between them. The clearing house will let each bank know how much it owes (or is owed) to (or from) other banks to settle their accounts each banking day. In this example, $Bank_P$ and $Bank_Q$ settle by having $Bank_Q$ transfer \$150 to $Bank_P$. $Bank_P$ will credit \$250 to Dan's account and debit Alice's account by \$100. $Bank_Q$ will credit \$100 to Bob's account and debit \$250 from Carol's account. In the (hopefully) unlikely event that $Bank_Q$ is unable to cover its debts, the clearing house will pay $Bank_P$ what it is owed.

Another feature that provides the benefits of faster processing of checks to banks and consumers is the practice of *check truncation.*

Definition 6. *[7] defines* check truncation *as follows:*

> *The practice of holding a check at the institution at which it was deposited (or at an intermediary institution) and electronically forwarding the essential information on the check to the institution on which it was written. A truncated check is not returned to the writer.*

Banks and other financial institutions use electronic check conversion (ECC) to convert endorsed physical checks into legally equivalent check images in support of check truncation.

Definition 7. Electronic check conversion *is the process of using magnetic-ink character recognition (MICR) to capture information from a check's MICR line, including: the bank's routing number, account number, check number, check amount, and other information that are printed near the bottom of the check in magnetic ink in accordance with generally applicable industry standards.*

We represent $Bank_Q$'s ECC of an endorsed check as:

$$ECC_{Bank_Q} \textsf{ says } (Q \mid Signature_P) \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)$$

When electronically converted checks are used in the context of check truncation, this is known as electronic check presentment.

Definition 8. *[7] defines* electronic check presentment (ECP) *as follows:*

> *Check truncation methodology in which the paper check's MICR line information is captured and stored electronically for presentment. The physical checks may or may not be presented after the electronic files are delivered, depending on the type of ECP service that is used.*

$Bank_Q$'s presentation of the electronic check image can be represented as:

$$\begin{aligned} &Bank_Q \textsf{ says } (ECC_{Bank_Q} \textsf{ says} \\ &\qquad (Q \mid Signature_P) \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle))\end{aligned}$$

The above formula states that $Bank_Q$ is relaying the output of its ECC process. Presumably, $Bank_Q$ only makes this statement (i.e., vouches for its electronic check conversion process) when it believes the process is working correctly.

Figure 4 illustrates the use of a clearing house and check images. Bob, the payee, deposits Alice's check at his bank. Bob's bank does not present the check endorsed by Bob to Alice's bank directly. Rather, Bob's bank *truncates* the check, credits Bob's account, and sends an electronic version of the check (usually in a batch with other orders) to an Automated Clearing House (ACH) operator, who sends the image and information to Alice's bank to debit Alice's account. The ACH operator settles the accounts between Alice's and Bob's respective banks each day.

Clearing corporations such as the Federal Reserve Banks guarantee payments for depository financial institutions using services such as FedACH. Consequently, the Federal Reserve Banks take on the financial risk if a depository financial institution (DFI) defaults and has insufficient funds to settle. Hence, both ACH and the DFIs are signatories to transactions. Thus, ACH is not merely relaying information but assuming liability.

We represent the presentation of a check image created by $Bank_Q \mid ECC_{Bank_Q}$ by an ACH operator ACH as follows:

$$\begin{aligned} &((ACH \& Bank_Q) \mid ECC_{Bank_Q}) \mid Q \textsf{ says } (Signature_P \textsf{ says} \\ &\qquad\qquad (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle))\end{aligned}$$

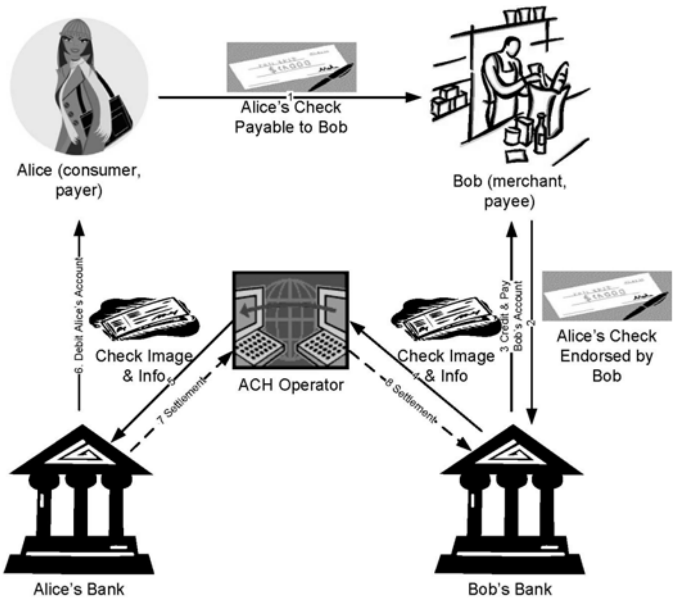


Fig. 4. Interbank Checking Using an Automated Clearing House

The operator is functioning as a clearing corporation and *counter signs* the check image. By so doing, the ACH operator assumes the risk of the transaction if $Bank_P$ defaults at settlement time.

In this system checks are cleared immediately without first checking balances. If there is an insufficient balance to cover the amount of the check, the check in question is returned to the depositor and the amount ultimately charged back to his or her account as a separate transaction. In Figure 4, if Alice's check bounces, then Alice's check (or a truncated version of her check) is returned by her bank to the ACH operator to debit Bob's bank the amount of the returned check.

Authorities, Jurisdiction, and Policies. The controlling authorities in this case include the bank owners with the addition of the Automated Clearing House (ACH) association, whose rules all members agree to follow as a condition of membership. Our analysis starts with $Bank_P$, as it is the paying bank.

$Bank_P$: At the individual account level, depositors are allowed to write checks. If there are insufficient funds in the account, another transaction will reverse the debit. Therefore, the policy allowing checks to be written is given below:

$$Bank_P\ \ Owner\ \textsf{controls+says}\ (P\ \textsf{controls}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))$$

The policy allows payees to be delegates of the payers indicated on checks:

$$Bank_P\ \ Owner\ \textsf{controls+says}\ (Q\ \textsf{reps}\ P\ \textsf{on}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))$$

Applying the *Controls* inference rules to the above statements produces the following policy statements for $Bank_P$:

$$P \;\textsf{controls}\; (\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle)$$
$$Q \;\textsf{reps}\; P \;\textsf{on}\; (\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle)$$

Because $Bank_P$ is part of the ACH network, it recognizes ACH as a counter signer with the ACH network banks that use ECP:

$$\begin{aligned}&Bank_P\ \mathit{Owner}\;\textsf{controls+says}\\&\quad((ACH\&Bank_Q) \mid ECC_{Bank_Q})\;\textsf{reps}\;(Q \mid Signature_P)\;\textsf{on}\\&\qquad\qquad(\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle)\end{aligned}$$

Using the *Controls* inference rule, we derive $Bank_P$'s policy regarding check images and information forwarded to it by ACH:

$$\begin{aligned}&((ACH\&Bank_Q) \mid ECC_{Bank_Q})\;\textsf{reps}\;(Q \mid Signature_P)\;\textsf{on}\\&\qquad\qquad(\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle)\end{aligned}$$

ACH: The ACH operator only accepts transactions from ACH members. In this example, the policies for the ACH operator regarding $Bank_P$ and $Bank_Q$ are as follows:

$$Bank_P\;\textsf{controls}\;\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle$$

The above states that the ACH operator will accept a payment from $Bank_P$ as part of the settlement process. The next formula states that $Bank_Q$ is allowed to present electronically converted checks to the operator:

$$\begin{aligned}&Bank_Q\;\textsf{reps}\;ECC_{Bank_Q}\;\textsf{on}\\&\qquad(Q \mid Signature_P)\;\textsf{says}\;(\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle))\end{aligned}$$

$Bank_Q$: The controlling authority for $Bank_Q$ is $Bank_Q$'s owner. The following statements are a result of recognizing $Bank_P$ as a banking partner as part of the ACH network (this would be determined from the MICR line). The first policy states that checks drawn upon accounts in $Bank_P$ may be deposited in $Bank_Q$'s accounts:

$$\begin{aligned}&Bank_Q\ \mathit{Owner}\;\textsf{controls+says}\\&\quad((Q \mid Signature_P\;\textsf{says}\;(\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge \langle \mathit{debit}\ \mathrm{amt}, \mathit{acct}_P\rangle)) \supset\\&\qquad\qquad(\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle \wedge (Q\;\textsf{controls}\;\langle \mathit{credit}\ \mathrm{amt}, \mathit{acct}_Q\rangle)))\end{aligned}$$

We are assuming here that funds are immediately available (i.e., there is no float time). The second policy states that $Bank_Q$ recognizes ACH's settlement statement:

$$Bank_Q\ \mathit{Owner}\;\textsf{controls+says}\;(ACH\;\textsf{controls}\;\langle \mathit{Pay}\ \mathrm{amt}, Q\rangle)$$

Applying the *Controls* inference rule to the above statements produces the following policies for $Bank_Q$:

$$((Q \mid Signature_P \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \supset (Q \textsf{ controls } \langle credit \text{ amt}, acct_Q\rangle \wedge \langle Pay \text{ amt}, Q\rangle))$$

and

$$ACH \textsf{ controls } \langle Pay \text{ amt}, Q\rangle$$

Operating Rules. There are five access-control decisions to be made, corresponding to the arrows labeled 3–8 in Figure 4. The first decision (arrow 3) is made by $Bank_Q$. This decision corresponds to Bob's request to deposit Alice's check, credit his account by the same amount, and have the funds made available to him. This decision is made by the *ACH Check Deposit* rule, whose proof is in Figure 5:

$$\textit{ACH Check Deposit}\quad \frac{\begin{array}{c} Signature_Q \mid Signature_P \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle) \\ Signature_Q \textsf{ says } \langle credit \text{ amt}, acct_Q\rangle \\ Signature_Q \Rightarrow Q \\ (Q \mid Signature_P \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \supset \\ \langle Pay \text{ amt}, Q\rangle \wedge (Q \textsf{ controls } \langle credit \text{ amt}, acct_Q\rangle) \end{array}}{\langle Pay \text{ amt}, Q\rangle \wedge \langle credit \text{ amt}, acct_Q\rangle}$$

The second decision (arrow 4) is also made by $Bank_Q$, which must decide whether or not to electronically present the check endorsed by Q to the ACH operator. If the check is endorsed by a depositor Q of $Bank_Q$, $Signature_Q$ is Q's signature, and the check itself passes whatever integrity check is used by the bank, then the check is converted to its electronic version and passed on to the ACH operator. This decision is made by the *ACH Check Presentation* rule, which is proved in Figure 6:

$$\textit{ACH Check Presentation}\quad \frac{\begin{array}{c} Signature_Q \mid Signature_P \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle) \\ Signature_Q \Rightarrow Q \end{array}}{Bank_Q \textsf{ says } (ECC_{Bank_Q} \textsf{ says } (Q \mid Signature_P) \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle))}$$

The third decision (arrow 5) is made by the ACH operator to counter sign the electronically converted check and present it to $Bank_P$. This decision uses the *ACH Counter Sign* rule, whose proof is in Figure 7:

$$\textit{ACH Counter Sign Rule}\quad \frac{\begin{array}{l} Bank_Q \textsf{ says } (ECC_{Bank_Q} \textsf{ says} \\ \quad (Q \mid Signature_P) \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \\ Bank_Q \textsf{ reps } ECC_{Bank_Q} \textsf{ on} \\ \quad (Q \mid Signature_P) \textsf{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \end{array}}{\begin{array}{l} (ACH \& Bank_Q) \textsf{ says } (ECC_{Bank_Q} \textsf{ says } ((Q \mid Signature_P) \textsf{ says} \\ \quad (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle))) \end{array}}$$

1.	$Signature_Q \mid Signature_P \text{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)$	Endorsed Check
2.	$Signature_Q \text{ says } \langle credit \text{ amt}, acct_Q\rangle$	Deposit Slip
3.	$Signature_Q \Rightarrow Q$	Signature on File
4.	$(Q \mid Signature_P \text{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \supset \langle Pay \text{ amt}, Q\rangle \wedge (Q \text{ controls } \langle credit \text{ amt}, acct_Q\rangle)$	$Bank_Q$ Policy
5.	$Signature_P \Rightarrow Signature_P$	Idempotency of $\Rightarrow$
6.	$Signature_Q \mid Signature_P \Rightarrow Q \mid Signature_P$	Monotonicity of $\mid$
7.	$Q \mid Signature_P \text{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)$	6, 1 Derived Speaks For
8.	$\langle Pay \text{ amt}, Q\rangle \wedge (Q \text{ controls } \langle credit \text{ amt}, acct_Q\rangle)$	7, 4 Modus Ponens
9.	$\langle Pay \text{ amt}, Q\rangle$	8 Simplification (1)
10.	$Q \text{ controls } \langle credit \text{ amt}, acct_Q\rangle$	8 Simplification (2)
11.	$Q \text{ says } \langle credit \text{ amt}, acct_Q\rangle$	3, 2 Derived Speaks For
12.	$\langle credit \text{ amt}, acct_Q\rangle$	10, 11 Controls
13.	$\langle Pay \text{ amt}, Q\rangle \wedge \langle credit \text{ amt}, acct_Q\rangle$	9, 12 Conjunction

Fig. 5. Proof of ACH Check Deposit

The fourth decision (arrows 6 and 7) is made by $Bank_P$ to debit the appropriate account and pay toward settlement. This rule (*ACH Check Funding*) is proved in Figure 8:

$$\textit{ACH Check Funding}\quad \frac{\begin{array}{l} (ACH\&Bank_Q) \text{ says } (ECC_{Bank_Q} \text{ says } ((Q \mid Signature_P) \text{ says} \\ \quad (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle))) \\ ((ACH\&Bank_Q) \mid ECC_{Bank_Q}) \text{ reps } Q \text{ on} \\ \quad (Signature_P \text{ says } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle)) \\ Signature_P \Rightarrow P \\ P \text{ controls } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle) \\ Q \text{ reps } P \text{ on } (\langle Pay \text{ amt}, Q\rangle \wedge \langle debit \text{ amt}, acct_P\rangle) \end{array}}{(Bank_P \text{ says } \langle Pay \text{ amt}, Q\rangle) \wedge \langle debit \text{ amt}, acct_P\rangle}$$

The final decision (arrow 8) is made by the ACH operator using the *ACH Check Settlement Rule*, which is proved in Figure 9.

$$\textit{ACH Check Settlement Rule}\quad \frac{\begin{array}{c} Bank_P \text{ says } \langle Pay \text{ amt}, Q\rangle \\ Bank_P \text{ controls } \langle Pay \text{ amt}, Q\rangle \end{array}}{ACH \text{ says } \langle Pay \text{ amt}, Q\rangle}$$

Risks. All of the risks of paper-based checking are present in the ACH system. There is an additional risk incurred here, because in practice the ACH system does not look up signatures to ensure that the signature on a check matches a signature on file. Instead, checking a signature occurs only when a customer complains about a fraudulent check.

Despite the potential risks, the ACH system is largely trustworthy. The ACH system is highly automated and runs on exceptions, so that large numbers of transactions are cleared daily. Transactions are largely assumed to be legitimate, and the evidence in terms of the number of checks returned unpaid supports this. The 2004 Federal Reserve Payments Study [8] reported:

In 2000, the number of checks returned unpaid was 0.6 percent of checks paid by depository institutions, compared to 0.5 percent in 2003. The value per returned check has remained relatively unchanged: \$756 [in 2003] compared to \$747 ... [in 2000].

1. $Signature_Q \mid Signature_P\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)$	Endorsed Check
2. $Signature_Q \Rightarrow Q$	Signature on File
3. $Signature_P \Rightarrow Signature_P$	Idempotency of $\Rightarrow$
4. $Signature_Q \mid Signature_P \Rightarrow Q \mid Signature_P$	Monotonicity of $\mid$
5. $Q \mid Signature_P\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)$	4, 1 Derived Speaks For
6. $ECC_{Bank_Q}\ \textsf{says}\ (Q \mid Signature_P\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))$	6 Says
7. $Bank_Q\ \textsf{says}\ ECC_{Bank_Q}\ \textsf{says}\ (Q \mid Signature_P\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))$	7 Says

Fig. 6. Proof of ACH Check Presentation

1. $Bank_Q\ \textsf{says}\ (ECC_{Bank_Q}\ \textsf{says}\ (Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	Presented Check
2. $Bank_Q\ \textsf{reps}\ ECC_{Bank_Q}\ \textsf{on}\ (Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	ACH policy
3. $Bank_Q \mid ECC_{Bank_Q}\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	1 Quoting (2)
4. $ECC_{Bank_Q}\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	2, 3 Rep Says
5. $ACH\ \textsf{says}\ (ECC_{Bank_Q}\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))))$	4 Says
6. $ACH\&Bank_Q\ \textsf{says}\ (ECC_{Bank_Q}\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))))$	5, 1 &Says (2)

Fig. 7. Proof of ACH Counter Sign Rule

1. $(ACH\&Bank_Q)\ \textsf{says}\ (ECC_{Bank_Q}\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle))))$	Check presented by ACH
2. $((ACH\&Bank_Q) \mid ECC_{Bank_Q})\ \textsf{reps}\ Q\ \textsf{on}\ (Signature_P\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	$Bank_P$ policy
3. $Signature_P \Rightarrow P$	Signature on File
4. $P\ \textsf{controls}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)$	$Bank_P$ policy
5. $Q\ \textsf{reps}\ P\ \textsf{on}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)$	$Bank_P$ policy
6. $(ACH\&Bank_Q \mid ECC_{Bank_Q})\ \textsf{says}\ ((Q \mid Signature_P)\ \textsf{says}\ (\langle Pay\ \text{amt}, Q\rangle \wedge \langle debit\ \text{amt}, acct_P\rangle)))$	1 Quoting (2)

Fig. 8. Proof of ACH Check Funding

1. $Bank_P\ \textsf{says}\ \langle Pay\ \text{amt}, Q\rangle$	$Bank_P$ authorizing payment to Q
2. $Bank_P\ \textsf{controls}\ \langle Pay\ \text{amt}, Q\rangle$	ACH's policy to rely/trust in $Bank_P$'s authorization
3. $\langle Pay\ \text{amt}, Q\rangle$	2, 1 Controls
4. $ACH\ \textsf{says}\ \langle Pay\ \text{amt}, Q\rangle$	3 says

Fig. 9. Proof of ACH Check Settlement Rule

The ACH rules [5] and Check 21 regulations [6] add consistency and uniformity of operations and formats that supports delegation and trust. Speeding up the processing of checks due to check truncation reduces the float time for checks and works against frauds such as check kiting. Guarding against fraud ultimately depends on the consumer's awareness and diligence in monitoring transactions. The ability, willingness, and speed with which consumers are able to detect "exceptions" varies and is likely the largest risk in the payment system.

4 Conclusions

Systems engineering is difficult in part because of the myriad interacting components and the often crucial and undocumented assumptions about the context in which a component, subsystem or system will operate. (A particularly stunning example was the loss of the Mars Climate Orbiter due to Lockheed Martin engineers assuming an English interpretation of data while NASA engineers assumed a metric interpretation). In systems where delegates are used extensively and access-control decisions are made automatically, it is crucial to understand and precisely document how and why these decisions are made. Understanding of complicated systems is best served when the underlying formal system of reasoning is simple yet effective. We believe our logic for delegation and trust has these properties and is a useful tool for systems engineers and system certifiers.

Regarding the check-based retail payment system itself, using the access-control logic brings the intricacies of trust and delegation to the fore. These intricacies include precise statements about who is trusted on precisely what statements. Using the logic and delegation definitions, we are able to formally justify what amounts to the access-control decisions made in the retail payment system. Not only does this provide insight into how the system works, it also provides a precise specification for how the check-based system should behave. While we have focused entirely on paper and electronic checks, a similar description and analysis should hold for other payment instruments, such as credit cards, debit cards, and stored-value cards.

Our long-term goal is to provide to systems engineers a similar combination of accessibility, usability, and rigor that hardware engineers enjoy with digital logic. In particular, digital logic is the foundation of hardware design and verification. It is introduced at the introductory undergraduate level to explain core concepts with precision. At more advanced levels, digital logic provides the formal basis for computer-aided design tools such as verifiers and synthesizers.

We have used our access-control logic to describe a variety of systems, including everyday situations such as airport security, web-based services such as CORBA [1], control of physical memory [9,10], and role-based access control [2]. We have incorporated our logic into courses at both the undergraduate and graduate levels [11,10]. In our experience, this logic is accessible to rising juniors and seniors in computer science and computer engineering. These experiences lead us to believe that the goal of giving engineers simple, effective, and mathematically sound tools for assuring security is feasible and within reach.

References

1. Kosiyatrakul, T., Older, S., Humenn, P.R., Chin, S.K.: Implementing a calculus for distributed access control in higher order logic and hol. In: Gorodetsky, V., Popyack, L.J., Skormin, V.A. (eds.) MMM-ACNS 2003. LNCS, vol. 2776, pp. 32–46. Springer, Heidelberg (2003)
2. Kosiyatrakul, T., Older, S., Chin, S.K.: A modal logic for role-based access control. In: Gorodetsky, V., Kotenko, I.V., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 179–193. Springer, Heidelberg (2005)

3. Lampson, B., Abadi, M., Burrows, M., Wobber, E.: Authentication in Distributed Systems: Theory and Practice. ACM Transactions on Computer Systems 10(4), 265–310 (1992)
4. Abadi, M., Burrows, M., Lampson, B., Plotkin, G.: A Calculus for Access Control in Distributed Systems. ACM Transactions on Programming Languages and Systems 15(4), 706–734 (1993)
5. National Automated Clearing House Association 13665 Dulles Technology Drive, Suite 300, Herndon, VA 20171: 2006 ACH Rules: A Complete Guide to Rules and Regulations Governing the ACH Network (2006)
6. 108th Congress (Check 21 act) Public Law Number 108–100, 117 Stat (2003). Public Law 108–100 was the 100th law passed by the 108th Congress. It was published in vol. 117, p. 1177 (2003) of the United States Statutes at Large at available at http://www.federalreserve.gov/paymentsystems/truncation/
7. Federal Financial Institutions Examination Council: Retail Payment Systems: IT Examination Handbook (2004) Available under IT Booklets on the FFIEC IT Handbook InfoBase web page at http://www.ffiec.gov/
8. Federal Reserve System: The 2004 Federal Reserve Payments Study: Analysis of Noncash Payments Trends in the United States: 2000–2003 (2004) Available at www.frbservices.org/Retail/pdf/2004PaymentResearchReport.pdf
9. Saltzer, J., Schroeder, M.: The protection of information in computer systems. In: Proceedings of IEEE 1975, IEEE Computer Society Press, Los Alamitos (1975)
10. Chin, S.K., Older, S.: A rigorous approach to teaching access control. In: Proceedings of the First Annual Conference on Education in Information Security, ACM, New York (2006)
11. Older, S., Chin, S.K.: Using Outcomes-based Assessment as an Assurance Tool for Assurance Education. Journal of Information Warfare 2(3), 86–100 (2003)

Performance Evaluation of Keyless Authentication Based on Noisy Channel

Valery Korzhik[1], Viktor Yakovlev[1], Guillermo Morales-Luna[2], and Roman Chesnokov[1]

[1] State University of Telecommunications, St.Petersburg, Russia
korzhik@spb.lanck.net
[2] Computer Science Department, CINVESTAV-IPN, Mexico City, Mexico
gmorales@cs.cinvestav.mx

Abstract. We consider a cryptographic scenario of two honest parties which share no secret key initially, but their final goal is to generate an information-theoretical secure key. In order to reach this goal they use assistance of some trusted center (as a satellite) that broadcasts a random string to legal users over noisy channels. An eavesdropper is able to receive also this string over another noisy channel. After an execution of the initialization phase, legal parties use discussion over noiseless public channels existing between them. The eavesdropper can intervene in the transmission and change the messages transmitted by legal parties. Thus, it is necessary to provide authentication of these messages. Otherwise the legal parties may agree a false key with the eavesdropper instead. In this paper we develop a concept of authentication based on noisy channels and present a performance evaluation of authentication procedures both for non-asymptotic and asymptotic cases.

Keywords: Authentication, Bhattacharyya distance, error correcting codes, wiretap channels.

1 Introduction

1.1 Model for Key Distribution in Presence of Active Eavesdropper

Let us consider the model of key distribution between legal users Alice (A) and Bob (B) in the presence of an active adversary Eve (E) assuming that initially the legal users had no shared secret keys (Fig. 1).

The *key distribution protocol* (KDP) consists of two phases: the *initialization phase* and the *key generation phase.*

In the initialization phase A, B, and E receive random i.i.d. sequences $X = (x_i)_{i=1}^k, Y = (y_i)_{i=1}^k$, $Z = (z_i)_{i=1}^k \in \{0,1\}^k$, respectively, such that for each i, $p_m = \Pr(x_i \neq y_i)$ and $p_w = \min\{\Pr(x_i \neq z_i), \Pr(y_i \neq z_i)\}$. One of the methods to provide legal users A, B with the sequences X, Y is to generate the truly random sequence $S = (s_i)_{i=1}^k \in \{0,1\}^k$, by some trusted party, say source S, and then to transmit it to the legal users A and B over noisy channels

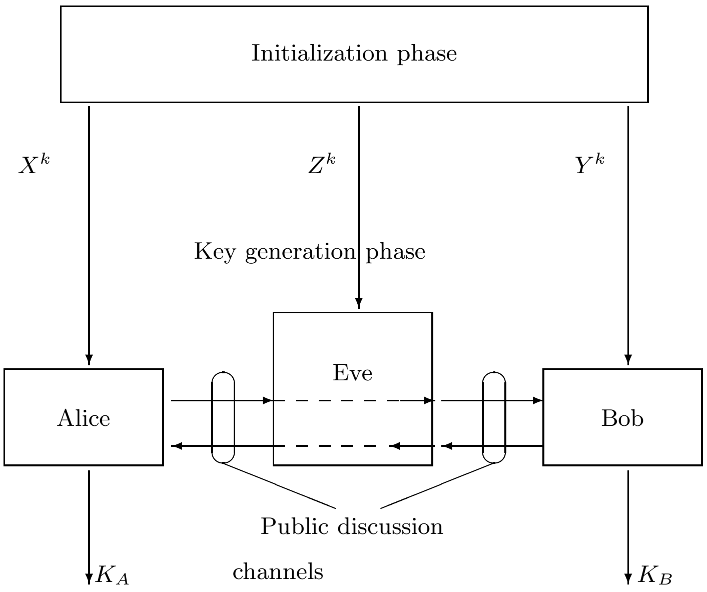


Fig. 1. Model of the Key Distribution Protocol

(*source model* [1,2]). Let us assume A and B receive sequences X, Y over *binary symmetric channels* without memory (BSC) with error probabilities $\pi_A = \Pr(x_i \neq s_i)$, $\pi_B = \Pr(y_i \neq s_i)$, while the adversary E receives the sequence Z over a BSC with the error probability of the wiretap $\pi_E = \Pr(z_i \neq s_i)$. One may imagine that the source S is some friendly satellite transmitter or some remote transmitter on Earth or even a natural space source as a pulsar (although the last case is rather exotic since it would be very complex and expensive to arrange the directional antenna devices and a powerful signal processor for such weak signals).

After the execution of the initialization phase, the source model can be reduced to the *channel model* where user A sends the sequence X to user B who receives it as Y, whereas E receives X as Z. Then the error probability on the *main virtual* BSC between A and B is $p_m = \pi_A + \pi_B - 2\pi_A\pi_B$ and the probability of the *wiretap virtual* BSC from B to E is $p_w = \pi_B + \pi_E - 2\pi_B\pi_A$.

In KDP's initialization phase it is natural to assume that the adversary E is unable to intervene the transmission from S to A and B. But this is not the case when legal users are exchanging information over a *public discussion channel* (PDC) at the *key generation phase.* We note that the use of PDC is necessary to send check symbols for agreement test of the strings X and Y and even for hash function transmission [3]. E can receive all information transmitted over the PDC. We assume that the PDC's between legal users and E are noiseless BSC's if E does not intervene in transmission. However E can replace this information as desired and therefore it is necessary to authenticate signals transmitted over PDC to detect any intervention of E and to reject suspicious messages.

1.2 Authentication Based on Noisy Channels

In order to solve the above stated problem it is necessary firstly to design a *keyless message authentication* based on noisy channels. In [2] a special code type has been proposed aiming to this problem: the *authentication codes* (AC). Let us describe the authentication procedure: in an initialization phase the legal users share the strings X, Y over BSC ($\Pr(x_i \neq y_i) = p_m$) and agree an error correcting binary but not necessary linear (n,k)-code V of length n consisting of 2^k codewords in order to authenticate any message of length k.

The authenticator $a = (a_1, a_2, \ldots, a_n)$ of a message m is formed as follows: for each i, let v_i be the i-th bit of the codeword in V corresponding to m and let $a_i = x_i$ if $v_i = 1$ or let it undefined otherwise. After receiving a pair $(\tilde{m}, \tilde{a})$, user B forms similarly $\tilde{\tilde{a}}$ for the message $\tilde{m}$ using his string Y and compares $\tilde{\tilde{a}}$ with $\tilde{a}$. If the number of bit disagreements is less or equal than a given threshold Δ then message $\tilde{m}$ is assumed authentic, otherwise it is rejected.

The AC's [2] efficiency can be characterized by two probabilities:

P_f: The probability of *false removal* of the message although the adversary E does not intervene at all.

P_d: The probability of *deception* false information: the event in which E has forged the message and this fact is not detected by B.

In sections 2, 3 we present an AC's performance evaluation for non-asymptotic and asymptotic (as $k \to +\infty$) cases. In section 4 we consider another authentication method not based on AC's. Section 5 states conclusions and open problems.

2 Performance Evaluation of AC's

As remarked in [2], AC's efficiency does not depend directly on the ordinary minimum code distance of V but on the *minimum asymmetric semidistance* d_{01}:

Definition 1. *The* asymmetric semidistance d_{01} *of the code V is the minimal number of* $0,1$*-symbol transitions among any pairs of distinct codewords in V:*

$$d_{01} = \min\left\{d_{01}(v,v') = \#\left\{i \leq n \mid v_i = 0 \ \& \ v'_i = 1\right\} \mid v \neq v', \ v, v' \in V\right\}$$

In order to provide equality in authentication efficiency among all messages we restrict our consideration to *constant Hamming-weight codes.*

Theorem 1. *Let V be an (n,k)-AC with constant Hamming weights τ for nonzero codewords and asymmetric semidistance d_{01}. Then the probabilities P_f and P_d in the authentication procedure over a noisy wire-tap channel described in section 1.2, for any deception strategy of the adversary, are upper bounded as:*

$$P_f \leq \sum_{i=\Delta+1}^{\tau} \binom{\tau}{i} p_m^i (1-p_m)^{\tau-i} \tag{1}$$

$$P_d \leq \sum_{i=0}^{\Delta} \binom{d_{01}}{i} p_w^i (1-p_w)^{d_{01}-i} \cdot \sum_{j=0}^{\Delta-i} \binom{\tau-d_{01}}{j} p_m^i (1-p_m)^{\tau-d_{01}-j} \tag{2}$$

where p_m is the error probability in main channel, p_w is the error probability in the wire-tap channel, and Δ is a threshold used in the authentication procedure.

Proof. We note that the length of authenticators for any message is equal to τ. Relation (1) is obvious because its right side is exactly the probability of occurrence of more than Δ disagreements between the strings of length τ assuming that each bit disagreement is independent of any other and the probability of a disagreement in every bit is equal to p_m. In order to prove (2) we note firstly that the best adversary's substitution attack to decept a message m', having known a correct pair (m, w), is to find the codewords v and v' in the AC V corresponding to m and m' and to compare them. If it happens that $v_i = v'_i = 1$ (where v_i, v'_i are the i-th bits of v and v') then the best strategy is to select as the i-th bit of the authenticator w' the i-th bit in w, otherwise the i-th bit of her received string Z. We observe that the disagreement probability among authenticators bits in w and w' is p_m at those positions i where $v_i = v'_i = 1$ and is p_w at those where $v_i = 0$, $v'_i = 1$. Since we assume $p_m < p_w$, in order to decrease P_d it is necessary to increase the number of positions where $v_i = 0$, $v'_i = 1$. This fact entails inequality (2) because d_{01} is the minimum number of such positions and the probabilities of bits disagreements in authenticators w and w' are p_m whenever $v_i = v'_i = 1$ and p_w whenever $v_i = 0$, $v'_i = 1$. □

Unfortunately, the estimation of P_f and P_d by (1) and (2), for large values of τ, is a hard problem. Therefore we present below their upper Chernoff's bounds.

Theorem 2. *Under the conditions of Theorem 1:*

$$P_f \le \left(\frac{1-p_m}{p_m}\frac{\Delta}{\tau-\Delta}\right)^{-\Delta}\left((1-p_m)\left(1+\frac{\Delta}{\tau-\Delta}\right)\right)^{\tau} \qquad (3)$$

$$P_d \le e^{-\nu\Delta}\left(p_w e^{\nu} + (1-p_w)\right)^{d_{01}}\left(p_m e^{\nu} + (1-p_m)\right)^{\tau-d_{01}} \qquad (4)$$

where $\nu = \ln x$ and x is the root of the following quadratic equation

$$p_m p_w(\tau-\Delta)\,x^2 + [p_m\tau(1-p_w) - d_{01} - (p_m+p_w)]\,x + p_w - p_m p_w - 1 = 0 \quad (5)$$

which provides a minimum at the right side of (4).

Proof. From the facts in the proof of Theorem 1, we have

$$P_f = \Pr\left(\sum_{i=1}^{\tau}\xi_i > \Delta\right) \qquad \text{and} \qquad P_d = \Pr\left(\sum_{i=1}^{d_{01}}\eta_i + \sum_{i=1}^{\tau-d_{01}}\xi_i \le \Delta\right),$$

where $H_0 = (\eta_i)_i$, $\Xi_0 = (\xi_i)_i$ are sequences of i.i.d. binary random variables with

$\Pr(\xi_i = 1) = p_m$, $\Pr(\xi_i = 0) = 1-p_m$, $\Pr(\eta_i = 1) = p_w$, $\Pr(\eta_i = 0) = 1-p_w$.

Applying Chernoff's bounds [4] to the right side of (4) we get:

$$\Pr\left(\sum_{i=1}^{\tau}\xi_i > \Delta\right) \le e^{-s\Delta}\left[E\left(\Xi_{1s}\right)\right]^{\tau} \qquad (6)$$

where $\Xi_{1s} = \left(e^{s\xi_i}\right)_i$, and s is a real number satisfying

$$\tau\frac{E\left(\Xi_{2s}\right)}{E\left(\Xi_{1s}\right)} = \Delta \tag{7}$$

where $\Xi_{2s} = \left(\xi_i e^{s\xi_i}\right)_i$. It is easy to see that

$$E\left(\Xi_{2s}\right) = p_m e^s \quad , \quad E\left(\Xi_{1s}\right) = p_m e^s + (1 - p_m) \tag{8}$$

Substituting last relations into (6) and (7) we get, respectively

$$P_f \leq e^{-s\Delta}\left[p_m e^s + (1 - p_m)\right]^{\tau} \quad , \quad \Delta = \tau\frac{p_m e^s}{p_m e^s + (1 - p_m)} \tag{9}$$

The solution of last equation is

$$s = \ln\frac{1 - p_m}{p_m}\frac{\Delta}{\tau - \Delta}$$

and substituting this value into (9) gives the proof of (3). For (4), we use also Chernoff's bound

$$P_d \leq e^{-s\Delta}\left[E\left(H_{1s}\right)\right]^{d_{01}}\left[E\left(\Xi_{1s}\right)\right]^{\tau - d_{01}} \tag{10}$$

where $H_{1s} = \left(e^{s\eta_i}\right)_i$, and s is a real number satisfying

$$d_{01}\frac{E\left(H_{2s}\right)}{E\left(H_{1s}\right)} + (1 - d_{01})\frac{E\left(\Xi_{2s}\right)}{E\left(\Xi_{1s}\right)} = \Delta \tag{11}$$

where $H_{2s} = \left(\eta_i e^{s\eta_i}\right)_i$. It is easy to see that $E\left(H_{2s}\right) = p_w e^s$, $E\left(H_{1s}\right) = p_w e^s + (1 - p_w)$ Substituting last relations and (8) into (10) and (11) we get, respectively

$$\begin{aligned} P_d &\leq e^{-s\Delta}\left[p_w e^s + (1 - p_w)\right]^{d_{01}}\left[p_m e^s + (1 - p_m)\right]^{\tau - d_{01}} \\ \Delta &= d_{01}\frac{p_w e^s}{p_w e^s + (1 - p_w)} + (\tau - d_{01})\frac{p_m e^s}{p_m e^s + (1 - p_m)} \end{aligned} \tag{12}$$

Let us denote e^s by x in last equation. Then after some simplifications we get the quadratic equation (5) with respect to the new variable x. Substituting the root of this equation into (12) we complete the theorem's proof. □

Theorem 2 gives simpler (and simultaneously tight) relations for a performance comparison of different AC's. But eq's. (3) and (4) are not convenient to estimate the asymptotic behavior of the probabilities P_f and P_d as $P_f = P_d$. In order to solve this problem we use the notion of Bhattacharyya distance.

Theorem 3. *Under the condition of Theorem 1 the following upper bounds for the probability $P_e = \frac{1}{2}\left(P_f + P_d\right)$ holds as $n \to +\infty$:*

$$P_e \leq \frac{1}{2}\exp\left[-\frac{1}{4}\frac{d_{01}(p_m - p_w)^2}{\left(\frac{2\tau}{d_{01}} - 1\right)p_m(1 - p_m) + p_w(1 - p_w)}\right] \tag{13}$$

Proof. Let us deal an authentication procedure as hypothesis testing. Let $\Lambda = \sum_{i=1}^{\tau} \zeta_i$ where $\zeta_i = \begin{cases} 0 & \text{if } \tilde{\tilde{a}} = \tilde{a} \\ 1 & \text{otherwise} \end{cases}$ (we recall that B decides the authentication to be correct if $\Lambda \leq \Lambda_0$, for a threshold Λ_0, otherwise authentication is rejected).

If the adversary does not intervene at all (let us call it *hypothesis* H_0) then $(\zeta_i)_i$ is a sequence of binary i.i.d. random values with $\Pr(\zeta_i = 1) = p_m$ and $\Pr(\zeta_i = 0) = 1 - p_m$. In the case in which the adversary uses an optimal substitution attack (let us call it *hypothesis* H_1) we may assume that in the worst case (for the verifier B) $(\zeta_i)_i$ is also a sequence of binary i.i.d. random values and $\zeta_i = \eta_i$ for $i \leq d_{01}$ and $\zeta_i = \xi_i$ for $i > d_{01}$ and

$$\Pr(\xi_i = 1) = p_m\ ,\ \Pr(\xi_i = 0) = 1 - p_m\ ,\ \Pr(\eta_i = 1) = p_w\ ,\ \Pr(\eta_i = 0) = 1 - p_w$$

(actually the variables η_i can take not necessarily the first d_{01} positions but we assume this for simplicity and without any generality loss). Assuming, in line with the De Moivre-Laplace Theorem [5], that Λ has asymptotically (as $n \to +\infty$) a normal (Gaussian) distribution for both hypothesis, we have

$$H_0 \sim N(e_0, \sigma_0^2) \quad , \quad H_1 \sim N(e_1, \sigma_1^2) \tag{14}$$

where $e_0 = E(\Lambda|H_0) = \tau p_m$, $\sigma_0^2 = \mathrm{Var}(\Lambda|H_0) = \tau p_m(1-p_m)$, $e_1 = E(\Lambda|H_1) = d_{01}p_w + (\tau - d_{01})p_m$ and $\sigma_1^2 = \mathrm{Var}(\Lambda|H_1) = d_{01}p_w(1-p_w) + (\tau - d_{01})p_m(1-p_m)$.

According with [6], the Bhattacharyya distance (BD) is defined between two probability distributions P_{H_0} and P_{H_1} over a space Ω as

$$D_B(P_{H_0}, P_{H_1}) = -\ln\left(\int_\Omega \sqrt{P_{H_0}(\omega)P_{H_1}(\omega)}\, d\omega\right)$$

If P_{H_0}, P_{H_1} are one-dimensional Gaussian distributions, see (14), then [6],

$$D_B(P_{H_0}, P_{H_1}) = \frac{1}{4}\frac{(e_0 - e_1)^2}{\sigma_0^2 + \sigma_1^2} + \frac{1}{2}\ln\left(\frac{\sigma_0^2 + \sigma_1^2}{2\sigma_0\sigma_1}\right)$$

Substituting (14) into last equation, since $\frac{1}{2}\ln\left(\frac{\sigma_0^2+\sigma_1^2}{2\sigma_0\sigma_1}\right) \geq 0$, we get:

$$\begin{aligned} D_B(P_{H_0}, P_{H_1}) &\geq \frac{1}{4}\frac{d_{01}^2(p_m - p_w)^2}{2\tau p_m(1-p_m) + d_{01}\left(p_w(1-p_w) - p_m(1-p_m)\right)} \\ &= \frac{1}{4}\frac{d_{01}(p_m - p_w)^2}{\left(\frac{2\tau}{d_{01}} - 1\right)p_m(1-p_m) + p_w(1-p_w)} \end{aligned} \tag{15}$$

Also [6], the *Bayesian error probability* $P_e = \frac{1}{2}(P_f + P_d)$ under the condition of optimal hypothesis testing is bounded as $P_e \leq \frac{1}{2}e^{-D_B\left(P_{H_0}, P_{H_1}\right)}$. This fact and inequality (15) entail the theorem's assertion. □

It is a very hard problem to find d_{01} for any constant weight AC V. But there exists a very simple method [2] to design such a $(2n_0, k)$-AC V with a known d_{01} given a linear (n_0, k)-code V' with ordinary minimum code distance $d = d_{01}$. Namely, let us change each symbol 1 by 10 and each 0 by 01 in every codeword in V'. The non-linear $(2n_0, k)$-code V is a constant weight $(\tau = n_0)$ code.

3 Asymptotic Code Rate for AC's

The statement to be proved would follow directly from Theorem 2 but since we applied De Moivre-Laplace approximation in its proof and the Bayesian error probability P_e for the efficiency of hypothesis testing, we give another proof.

Theorem 4. *For any real numbers $\varepsilon, \delta \in [0,1]$ there exist an integer n_0 and a $(2n_0, k)$-AC V providing $\frac{k}{n_0} < 1 - \varepsilon$, and $P_f < \delta$, $P_d < \delta$, for all $n_0 > n_0'$.*

Proof. In line with the verification algorithm in section 1.2 we get that the probabilities of false alarm P_f and undetected deception P_d satisfy:

$$P_f = \Pr\left(\rho_H(a,a') > \Delta\right)\ ,\ P_d = \Pr\left(\rho_H'(a,a') \le \Delta\right) \tag{16}$$

where $\rho_H'(a,a')$ is the Hamming distance between the received authenticator a and the authenticator a' produced by verifier B under the assumption that adversary E has intervened using optimal strategy (see section 1.2), $\rho_H(a,a')$ is the analogous distance without the intervention of E, and Δ is a given threshold.

Let us assume that the $(2n_0, k)$-AC V has been constructed from a linear (n_0, k)-code V' with ordinary minimum code distance $d = d_{01}$.

Let us normalize the Hamming distances and the threshold as $\rho_{H0} = \frac{1}{n_0}\rho_H$, $\rho_{H0}' = \frac{1}{n_0}\rho_H'$, $\Delta_0 = \frac{1}{n_0}\Delta$. Then (16) is rewritten as

$$P_f = \Pr\left(\rho_{H0}(a,a') > \Delta_0\right)\ ,\ P_d = \Pr\left(\rho_{H0}'(a,a') \le \Delta_0\right) \tag{17}$$

The random differences among a and a' of length n_0 are Bernoulli trials with parameter p_m, assuming E does not intervene in the PDC, and p_w in at least d authenticator symbols and p_m in $(n_0 - d)$ symbols if E tries to forge with an optimal strategy. An application of the Chebishev's bounds [5] to (17) gives

$$P_f \le \frac{\operatorname{Var}\left(\rho_{H0}(a,a')\right)}{\left[\Delta_0 - E\left[\rho_{H0}(a,a')\right]\right]^2}\ ,\ P_d \le \frac{\operatorname{Var}\left(\rho_{H0}'(a,a')\right)}{\left[\Delta_0 - E\left[\rho_{H0}'(a,a')\right]\right]^2} \tag{18}$$

Clearly, since we are dealing with Bernoulli trials,

$$\begin{aligned}
E\left[\rho_{H0}(a,a')\right] &= p_m \\
E\left[\rho_{H0}'(a,a')\right] &= \frac{d}{n_0}p_w + \left(1 - \frac{d}{n_0}\right)p_m \\
\operatorname{Var}\left(\rho_{H0}(a,a')\right) &= \frac{p_m(1-p_m)}{n_0} \\
\operatorname{Var}\left(\rho_{H0}'(a,a')\right) &= \frac{1}{n_0}\left(\frac{d}{n_0}p_w(1-p_w) + \left(1 - \frac{d}{n_0}\right)p_m(1-p_m)\right)
\end{aligned}$$

Above relations with (18) give

$$P_f \le \frac{1}{n_0}\frac{p_m}{\left(\Delta_0 - p_m\right)^2} \tag{19}$$

$$P_d \le \frac{1}{n_0}\frac{\frac{d}{n_0}p_w(1-p_w) + \left(1 - \frac{d}{n_0}\right)p_m(1-p_m)}{\left(\Delta_0 - \frac{d}{n_0}p_w(1-p_w) - \left(1 - \frac{d}{n_0}\right)p_m(1-p_m)\right)^2} \tag{20}$$

Selecting (n_0, k_0)-codes with *Varshamov-Gilberts bound* [7], we get asymptotically

$$1 - R \leq g\left(\frac{d}{n_0}\right) \tag{21}$$

where

$$R = \frac{k_0}{n_0}$$

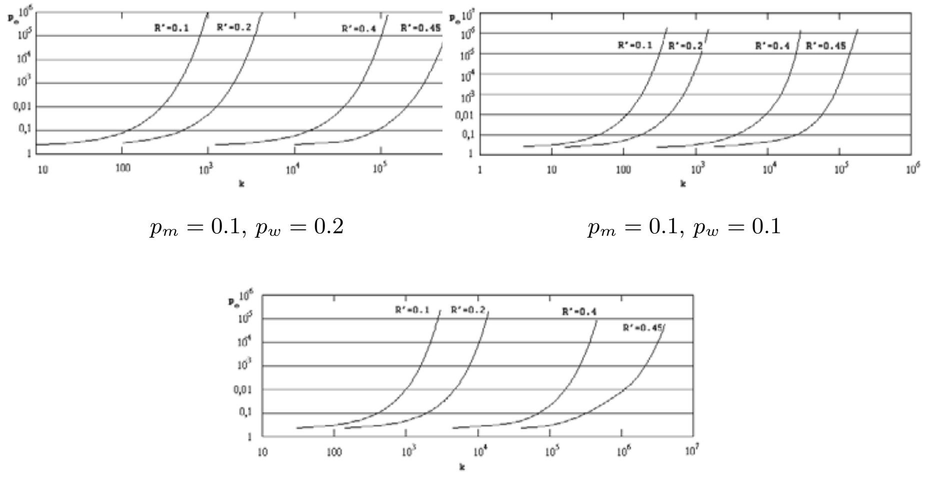
is the rate of the error correction (n_0, k_0)-code used in authentication and

$$g : [0,1] \to [0,1]\ ,\ p \mapsto -p\ln p - (1-p)\ln(1-p)$$

is the *entropy function.* If we take $R \geq 1 - \varepsilon$, then inequality (21) provides a constant ratio $\frac{d}{n_0}$ for any code length n_0. Substituting $\frac{d}{n_0}$ into (20) we can select n_0 enough large such that both $P_f < \delta$, $P_d < \delta$ hold.

This finishes the proof of the theorem. $\square$

We note that (19) and (20) are relevant to the asymptotic case but Chebyshev's inequality is very crude for P_e estimation within the finite length of the AC's. Therefore we exemplify AC performance evaluation using Bhattacharyya's distance bound (13). In Fig. 2 we plot P_e versus the authenticated message length k given different AC rates $R' = k/2n$ and wire-tap channel parameters p_m, p_w. The semidistances d_{01} in (13) were calculated as $d_{01} = d$ where d is the minimum code distance of the (n,k)-code satisfying Varshamov-Gilbert bound.



$p_m = 0.1$, $p_w = 0.2$ $p_m = 0.1$, $p_w = 0.1$

$p_m = 0.1$, $p_w = 0.05$

Fig. 2. The error probability P_e versus message length k

Theorem 4 renders immediately the fact that for large enough message length k the AC rate approaches 0.5 if AC has been constructed as described above.

But the following interesting question arises: is it possible to design an AC with a code rate larger than 0.5? In order to clarify this problem let us consider a *nonlinear constant weight code* with a given minimum code distance d.

Lemma 1. *Any nonlinear constant weight code C with minimum code distance d has always the minimum semidistance d_{01} equal to $d/2$.*

Proof. Consider any pair of distinct codewords $c, c' \in C$. It is easy to see that

$$\#\{i|\, c_i = 0 \& c'_i = 1\} + \#\{i|\, c_i = 1 \& c'_i = 0\} = d(c, c'),$$

and also that both relations

$$\begin{aligned}\#\{i|\, c_i = 0 \& c'_i = 1\} + \#\{i|\, c_i = 1 \& c'_i = 1\} = \tau \\ \#\{i|\, c_i = 1 \& c'_i = 0\} + \#\{i|\, c_i = 1 \& c'_i = 1\} = \tau\end{aligned}$$

do hold where τ is the weight of the codewords at C, and $d(c, c')$ is the Hamming distance between c and c'. From above relations, it follows

$$d_{01}(c, c') = \#\{i|\, c_i = 0 \ \& \ c'_i = 1\} = \#\{i|\, c_i = 1 \ \& \ c'_i = 0\} = d_{10}(c, c')$$

and $d_{01}(c, c') = \frac{1}{2} d(c, c')$. This relation means that whenever $d_{01}(c, c') = d$, for distinct $c, c' \in C$, then for the same words we will also have $d_{01}(c, c') = d/2$ and this is indeed the minimum semidistance for the code C. □

By *Johnson's bound* [8] the number $A(n, d, \tau)$ of codewords in any nonlinear constant τ-weight code of length n with minimum code distance d satisfies:

$$A(n, d, \tau) \le \binom{n}{\tau - \frac{d}{2} + 1} \binom{\tau}{\tau - \frac{d}{2} + 1}^{-1}. \tag{22}$$

Then the code rate of such code is upper bounded as

$$R_C \le \frac{1}{n} \log_2 \left[\binom{n}{\tau - \frac{d}{2} + 1} \binom{\tau}{\tau - \frac{d}{2} + 1}^{-1} \right] = R_0. \tag{23}$$

In order to prove that there do exist authentication codes with $R > 0.5$ and $P_f, P_d \to 0$ as $n \to +\infty$ it is necessary to prove that the right side of (23) is at least greater or equal then 0.5 for large enough n, some τ and $d = \alpha n$, where $\alpha > 0$ is some constant, assuming that there exist codes subject to Johnson's bound. Let us approximate the involved binomial coefficients by the relation [9] $\frac{\sqrt{\pi}}{2} G \le \binom{n}{\lambda n} \le G$ where $G = \frac{1}{\sqrt{2\pi\lambda n \mu}} \lambda^{-\lambda n} \mu^{-\mu n}$, with $\mu = 1 - \lambda$, and $\mu, \lambda \ne 0$. Then, substituting these bounds into (23) we get asymptotically (as $n \to +\infty$) after simple but tedious transformations

$$R_0 \approx \frac{\sqrt{\pi}}{2}\, g\,(x - a) - x\, g \left(1 - \frac{a}{x}\right) \tag{24}$$

where $a = \frac{d}{2n}$, $x = \frac{\tau}{n}$ and g is the entropy function.

It can be easily shown by (24) that R_0 is greater than 0.5 for small values of the parameter $a < 5 \times 10^{-4}$. This means that if there exists a family of nonlinear constant weight codes satisfying Johnson's bound (22) then the code rate of the AC's designed with such codes can be greater than 0.5 in some cases. But the existence of codes satisfying Johnson's bound is still an open problem. We can try, however, to increase the code rates of authentication codes refusing from the authentication algorithm presented in section 1.2.

4 Authentication Based on Bit-Wise Method

The use of an AC with some given semidistance d_{01} as was proposed in [2] (see also section 1.2.) appears to be not so natural. Let us try to consider another technique of unconditionally secure authentication based on the *bit-wise authentication method.*

In this setting a single message bit (either 0 or 1) is authenticated by one authentication bit with the use of a two-bit key, as shown in Fig. 3, where the key bits are labeling the graph's edges.

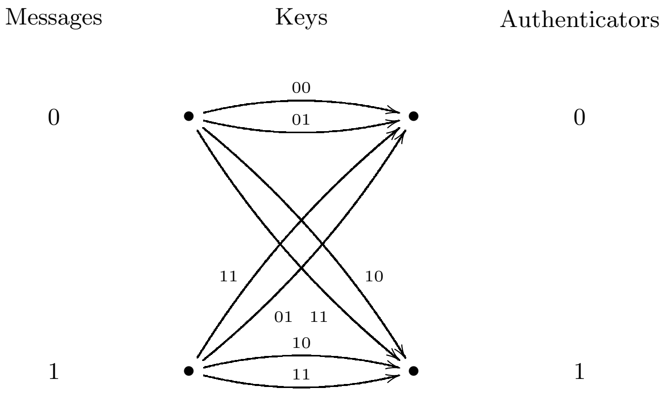


Fig. 3. Graph of a single message bit authentication

In order to authenticate a message of length k, the legal users agree an error correcting binary systematic linear (n, k)-code V with some chosen minimum code distance d. After the initialization phase, user A encodes the message m into the codeword $v \in V$, then A calculates the authenticator for each of the n bits in this codeword, by the rule sketched in Fig. 3 using her string X as the key. Next, A appends the authenticator to the codeword and sends this sequence to user B over the noiseless channel. As soon as B receives an authenticated codeword, he calculates his authenticator in the same manner as A did it but using his own string Y and then he compares both received and found authenticators.

If the Hamming distance between them is less or equal to some thresholds then he accepts message as authentic, otherwise he rejects it as forged.

Let us denote by (v_i, a_i) the pair of bits produced by A on the i-th position of the codeword and the authenticator corresponding to message m. The adversary E generates some false message m' and encodes it within the same code V. If the adversary does not intervene at all during the information transmission then, as can be seen in Fig. 3, the probability for B to get an incorrect authenticator $a'_i \neq a_i$ is equal to p_m. Then it is easy to see that the probability of false message rejection coincides exactly with relation (1).

Having received a correct pair (v_i, a_i), the adversary always knows one of the key bits used by A in the authentication procedure of the i-th bit. For him, the probability to "guess" the unknown key bit is p_w. If the false i-th bit v'_i of the codeword corresponding to the false message m', which E aims to send to B, coincides with v_i, then the best strategy for E is to select $a'_i = a_i$ due to the assumption $p_m < p_w$. If $v'_i \neq v_i$ then the best strategy for E is to select for the unknown key bit its corresponding in E's string Z. In the last case the probability for B to produce a different a'_i than the bit sent to him by E is p_w. The best case from E's point of view is the situation when the false codeword v' differs from the original codeword v in the minimum number of positions. But this number cannot be smaller than d under the condition of a shared code V.

This means that by allowing error occurrences in d authenticator positions with probability p_w and in $\tau - d$ positions with probability p_m, then an upper bound for the probability of successful deception is got. On the other hand, the same condition arises in the case of using the ordinary AC considered in section 2. Therefore the deception probability of the false message will coincide exactly with relation (2). This means that the new construction based on bit-wise methods gives the same performance evaluation as the use of an AC on the base of the same error correcting code and changing in it 0 to 01 and 1 to 10. Since this method does not show any advantages in comparison with the method based on AC it is meaningless to consider it in deeper details.

We remark however that the construction considered in this section can be generalized as follows: let us divide any block of length n in code V into sub-blocks of length $n_1 > 1$ and let us authenticate every such subblock using two or more bit-key and an one-bit authenticator. In this setting we can diminish the total length of codeword authenticator but performance evaluation of this authentication scheme requires more knowledge about the used error correcting code than just its minimum code distance.

5 Conclusion

We have analyzed the performance evaluation of keyless authentication based on noisy channels. This procedure is very important in executing a key distribution protocol (KDP) over noisy channels in presence of an active adversary. The design and implementation of quantum computing as well as the design of super fast multiprocessor conventional computers pose a serious threat of breaking some computationally secure cryptosystems, thus the use of KDP based on noisy channels is very promising because it can provide with perfect one-time pad unconditionally secure ciphers. In this paper we investigated the efficiency of the known authentication codes [2] and propose a new authentication procedure. Our main contribution is the formula proofs to calculate the probabilities of false rejection of authenticated message and undetected deception of forgery by adversary. The upper bounds based on Chernoff's inequality and Bhattacharyya distance have been proved also, allowing to calculate the probabilities quite simply and for large length of AC especially important in practice.

Initially we showed that there exists a very simple construction of AC providing asymptotically a code rate close to 0.5. Moreover we have shown that using nonlinear constant weight codes, the code rate greater than 0.5 is achieved if there exist codes satisfying Johnson's bounds. The design of such constructive codes poses an interesting open problem as far as an investigation of the maximum possible code rate of AC based on noisy channels.

Acknowledgements. Dr. Morales-Luna acknowledges Mexican *Consejo Nacional de Ciencia y Tecnología* (Conacyt) for partial support and José Luis Enríquez, of CINVESTAV-IPN Mathematics Department, for his help during the preparation of this paper.

References

1. Ahlswede, R., Csiszar, I.: Common randomness in information theory and cryptography – part I: Secret sharing. IEEE Transactions on Information Theory 39(4), 1121–1132 (1993)
2. Maurer, U.: Information-theoretically secure secret-key agreement by not authenticated public discussion. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 209–223. Springer, Heidelberg (1997)
3. Yakovlev, V., Korzhik, V., Morales-Luna, G.: Key distribution protocols based on noisy channels in presence of an active adversary: Conventional and new versions with parameter optimization. IEEE Transactions on Information Theory (Submitted to the special issue on Information Security) (2007)
4. Proakis, J.: Digital Communications, 4th edn. Mc Graw Hill Publishing Co. New York (2001)
5. Papoulis, A.: Probability, Random Variables and Stochastic Processes, 4th Rev. edn. McGraw-Hill Publishing Co. New York (2002)
6. Kailath, T.: The divergence and Bhattacharyya distance measures in signal selection. IEEE Trans. Commun. Tech. 15, 52–60 (1967)
7. MacWilliams, F., Sloane, N.: The Theory of Error-Correcting Codes. North-Holland, Amsterdam (1977)
8. Pless, V., Huffman, W.C., Brualdi, R.A.: Handbook of Coding Theory. North-Holland, Amsterdam (1998)
9. Peterson, W., Weldon, E.: Error-Correcting Codes, 2nd edn. MIT Press, Cambridge (1972)

Avoiding Key Redistribution in Key Assignment Schemes

Harry Rowe* and Jason Crampton

Royal Holloway, University of London
{a.h.rowe,jason.crampton}@rhul.ac.uk

Abstract. A key assignment scheme is a model for enforcing an information flow policy using cryptographic techniques. Such schemes have been widely studied in recent years. Each security label is associated with a symmetric encryption key: data objects are encrypted and authorised users are supplied with the appropriate key(s). However, updates to encryption keys pose a significant problem, as the new keys have to be issued to all authorised users. In this paper, we propose three generic approaches to key assignment schemes that remove the problem of key redistribution following key updates. We analyse the overheads incurred by these approaches and conclude that these overheads are negligible in practical applications.

Keywords: key assignment schemes, key redistribution, hierarchical access control.

1 Introduction

There are a number of situations in which access control is implemented using cryptographic techniques. Such an approach is useful when the (protected) objects are: read often, by many users; written once, or rarely, by the owner of the data; and transmitted over unprotected networks. Fu et al. [1] cite content distribution networks, such as Akamai and BitTorrent, as examples where this kind of technique might be useful.

A key assignment scheme is a form of cryptographic access control [2] that seeks to implement a no-read-up information flow policy [3]. Users and objects are assigned security labels; a user may have read access to an object if and only if the user's security label is at least as high as that of the object. Such a policy can be implemented by associating a symmetric encryption key with each security label and encrypting objects with the appropriate key. A user is given (or can derive) the encryption key for each label less than or equal to the user's security label.

The data owner encrypts protected objects and supplies authorised users with the appropriate keys. Hence, many users may have a copy of the same key,

* The research of Harry Rowe is supported by Microsoft Research through its PhD Scholarship programme.

enabling each user to decrypt a number of different protected objects. One of the main practical problems with key assignment schemes arises when a user's authorisation is revoked: each of the keys that this user has been given or can derive is "compromised". Hence it is necessary to create new encryption keys and to securely redistribute the new keys to all remaining authorised users. (Of course all objects also need to be re-encrypted.)

In this paper we examine the assumptions that have typically been made when designing key assignment schemes. In particular, we are interested in the practical implications of the design choices. We conclude that there are number of ways in which we can design key assignment schemes so that key redistribution is not required. Of course, one would expect such a scheme to have certain disadvantages compared to existing schemes; there is always a trade-off. However, we believe that in purely practical terms, the additional overheads incurred by our schemes will be perfectly acceptable in practice, and are more than offset by the fact that we no longer need to be concerned with key distribution.

The rest of this paper is organised as follows: Sect. 2 reviews existing KASs and explores their shortcomings. We introduce our new schemes in Sect. 3 and analyse their performance. Sect. 4 discusses the optimality of our preferred scheme and we discuss proposed optimisations for existing KASs in Sect. 5. Finally, Sect. 6 concludes this work.

2 Preliminaries

A *partially ordered set* (or *poset*) is a pair $(L, \leqslant)$, where $\leqslant$ is a reflexive, anti-symmetric, transitive binary relation on L. We may write $x \geqslant y$ whenever $y \leqslant x$. We say x *covers* y, denoted $y \lessdot x$, if $y < x$ and there does not exist $z \in L$ such that $y < z < x$. L is a *total order* if for all $x, y \in L$, either $x \leqslant y$ or $y \leqslant x$. The *Hasse diagram* of a poset is the directed graph $(L, \lessdot)$ [4]. A simple Hasse diagram is shown in Fig. 1(a). We will write e to denote the cardinality of the covering relation $\lessdot$ (that is, the number of edges in the Hasse diagram of L), and e^* to denote the cardinality of the partial order relation $\leqslant$. In general e and e^* are $\mathcal{O}\left(l^2\right)$, where l is the cardinality of L, although in certain special cases (as when L is a total order, for example) e is $\mathcal{O}(l)$ and e^* is $\mathcal{O}\left(l^2\right)$.

Henceforth we adopt the following conventions: x, when used as input to some (cryptographic) function, will denote a string identifying the security label x; H denotes a hash function; E denotes a symmetric encryption algorithm; and $E_k(m)$ denotes the encryption of message m with key k.

Definition 1. *An* information flow policy *is a tuple* $(L, \leqslant, U, O, \lambda)$*, where:*

- $(L, \leqslant)$ *is a (finite) partially ordered set of security labels;*
- U *is a set of users;*
- O *is a set of objects;*
- $\lambda : U \cup O \to L$ *is a security function that associates users and objects with security labels.*

Such policies were of particular interest in the mid-1970s, and formed part of the famous Bell-LaPadula security model. The basic operation of the policy is that a user u can read an object o if $\lambda(u) \geqslant \lambda(o)$. Clearly, one way of implementing such a policy is to encrypt an object with security label $y \in L$ with a key $k(y)$ and provide all users with security label $x \geqslant y$ with the key $k(y)$. Henceforth, we will represent an information flow policy $(L, \leqslant, U, O, \lambda)$ as a pair $(L, \leqslant)$ with the tacit understanding that U, O and λ are given. We assume that the policy will be implemented by encrypting objects and distributing keys to users, enabling them to decrypt the objects to which they should have access.

2.1 Key Assignment Schemes

Most key assignment schemes seek to minimise the number of keys that need to be distributed to users. This entails either making certain additional information public or providing each user with additional secret information (or both). A recent paper formalised the characteristic features of a key assignment scheme [2]. We summarise this approach below.

In general, a *key assignment scheme* (or *scheme*) for an information flow policy $(L, \leqslant)$ defines three algorithms, the first two being run by the *scheme administrator*, the third by end users:

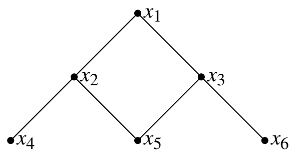
- makeKeys returns a labelled set of encryption keys $(\kappa(x) : x \in L)$, which we denote by $\kappa(L)$;
- makePublicData returns some set of data *Pub* that is made public by the scheme administrator;
- getKey takes $x, y \in L$, $\kappa(x)$ and the public data, and returns $\kappa(y)$ whenever $y \leqslant x$.

The paper also identified five generic constructions for key assignment schemes and compared these schemes according to the following criteria: amount of public storage required, amount of private storage required, complexity of key derivation, and complexity of key updates. It was concluded that there were two generic constructions that offered the best trade-offs with regard to these criteria. We discuss these constructions in the next two sections. In both schemes a user u has a single key $\kappa(\lambda(u))$.

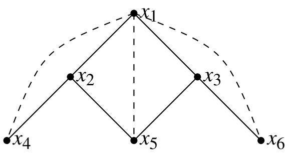
IKE KAS. An *iterative key encrypting* (IKE) key assignment scheme (KAS) uses public information to enable a user to iteratively derive keys for which he is authorised. An IKE KAS has the following characteristic features.

- $\kappa(x)$, $x \in L$, can be chosen at random from the key space;
- Public information $Pub_{\mathrm{I}} = \{E_{\kappa(x)}(\kappa(y)) : y \lessdot x : x, y \in L\}$;
- If $y \lessdot x$, a user with security label x can use $\kappa(x)$ to obtain $\kappa(y)$ by decrypting the public information $E_{\kappa(x)}(\kappa(y))$;

  If $y < x$, there exists a path $y = z_0 \lessdot z_1 \cdots \lessdot z_n = x$, $n \geqslant 1$, so a user with security label x can successively derive keys $\kappa(z_{n-1}), \ldots, \kappa(z_0) = \kappa(y)$.



(a) Edges used in an IKE KAS



(b) Edges used in a DKE KAS

Fig. 1. The directed graphs used in key assignment schemes; the transitive relationships in the partially ordered set are denoted by broken lines

In all the KASs we consider in this paper, each datum of public information corresponds to an edge in some graph. In the case of an IKE KAS, this graph is $(L, \lessdot)$. Hence, the public information storage requirements for a KAS is proportional to the total number of edges in the graph, and we use the terms synonymously. The edges used in an IKE KAS are illustrated in Fig. 1(a),

DKE KAS. A *direct key encrypting* (DKE) KAS uses public information to enable a user to directly derive keys for which he is authorised. Such a scheme is shown in Fig. 1(b), and has the following characteristic features.

– $\kappa(x)$, $x \in L$ can be chosen at random from the key space;
– Public information $Pub_{\mathrm{D}} = \{E_{\kappa(x)}(\kappa(y)) : y < x : x, y \in L\}$;
– If $y < x$, a user with security label x can use $\kappa(x)$ to obtain $\kappa(y)$ by decrypting the public information $E_{\kappa(x)}(\kappa(y))$.

The Atallah, Frikken and Blanton (AFB) KAS. The basic AFB scheme, proposed by Atallah et al. [5], is an example of an IKE KAS, which is secure against collusion, simple to set up and maintain, allows easy modifications to the hierarchical structure, and is efficient in terms of its storage requirements and key derivation time. In our notation, the AFB scheme can be represented in the following way:

– keys are chosen at random from $\{0,1\}^m$, where m is a positive integer;
– $Pub = \{\kappa(y) - H(\kappa(x), y) : y \lessdot x : x, y \in L\}$, where $H : \{0,1\}^* \to \{0,1\}^m$ is a hash function.

Note that the AFB scheme can be transformed into a DKE KAS by setting $Pub = \{\kappa(y) - H(\kappa(x), y) : y \leqslant x : x, y \in L\}$, allowing a single step key derivation.

2.2 Implementation Considerations

In order to meaningfully evaluate a KAS, it is necessary to consider its proposed applications. Without reference to particular implementations, the expected purpose of a KAS is to facilitate controlled access to data resources, such as a *cryptographically protected file system* (CPFS) hosted on a remote file server.

The use of cryptography to enforce access control policies is most applicable in systems with distributed components whose security cannot be guaranteed and in which the data must be transmitted over untrusted communication channels.[1]

Thus, we assume that (i) objects must be decrypted, and (ii) a publicly accessible resource server exists, hosting an information store of significant size. With these points in mind, it becomes clear that certain properties of KASs are less important than might be expected from a purely abstract analysis.

Key Derivation Time. The *key derivation time* is the average number of key derivations the user is required to compute to obtain a particular key. In an IKE KAS, this is the average path length in the Hasse diagram of the poset $(L, \leqslant)$, which is $\mathcal{O}(l)$ in general. Naively, this would seem to be a significant disadvantage when compared to the DKE KAS, where key derivation always requires a single step.

However, the process of deriving a key requires the use of only the most efficient cryptographic primitives – typically one-way hash functions as in the AFB scheme [5] – and involves decrypting keys of ~ 256 bits. Once the key is obtained, slower symmetric primitives would be used to decrypt the actual data objects, the size of which will be orders of magnitude larger than the encryption keys. That is, the time taken to derive encryption keys, even in an IKE KAS, will be negligible compared to the time taken to actually decrypt the data object.

Public Storage Requirements. The *public storage requirements* for a KAS can be expressed as the number of edges required to be stored publicly: this is proportional to e^* in the case of a DKE KAS and proportional to e in the case of an IKE KAS. The difference between e^* and e is $\mathcal{O}\left(l^2\right)$ in general, so this could be regarded as an important factor in deciding whether to implement an IKE KAS or DKE KAS.

However, the edges in question are the same size as the keys, ~ 256 bits. If we take an extreme example, where L is a totally ordered set of 100 security levels, the DKE KAS would require no more than ~ 150 kB of storage space ($\binom{100}{2}$ edges + 100 node labels), compared with the IKE KAS requirements of ~ 3 kB. The actual cost, in real terms, to the organisation of storing the extra 150 kB of information is negligible, particularly when compared with the overall storage cost necessarily incurred by the target file system. Thus, the extra public storage requirements incurred by a DKE KAS are effectively irrelevant to the analysis.

Complexity of Key Updates. Whether through key compromise, user revocation or routine security precautions, it will be necessary to change some or all of the keys at certain points in time. Updating the public information is a relatively simple task, involving changes to a single public information store.

The complexity is essentially identical in an IKE KAS and a DKE KAS. In order to revoke a single user with security level x, we must update $\kappa(y)$ for all

[1] Significant en/decryption overheads are inevitably incurred and are likely to be unacceptable in a localised system, in which all components are trusted and where secure channels exist between each component.

$y \leqslant x$ (since we must assume that the user may have derived and cached any of these keys). Hence, every user with a security label $y \leqslant x$ will need to receive the new key for y.

Note, that this is still a major improvement on a *trivial key assignment scheme* (TKAS) [2], in which user u with security label x is given the set of keys $\{\kappa(y) : y \leqslant x\}$. In a TKAS, revoking u means that any user v such that $\lambda(v) \geqslant x$ or $\lambda(v) \leqslant x$ will require at least one new key. The advantage of having information (edges) stored publicly (as in IKE and DKE KASs), is that each user has a single key and only users v such that $\lambda(v) \leqslant x$ require a new private key when $\kappa(x)$ is updated; the remainder of the key update process is achieved by updating the public information.

2.3 Remaining Difficulties and Motivation

It is clear that there are advantages in using a KAS to control access to a CPFS over traditional methods of access control in distributed file systems. Somewhat ironically, less cryptographic overhead is incurred in a CPFS, since there is no need to establish a secure channel (which requires the server to encrypt the data and the client to decrypt) as opposed to a single decryption by the client when using a CPFS. There is also no need for an online *authorisation server* (AS) and (logically distinct) *reference monitor*, thus removing these potential bottlenecks and shrinking the *trusted computing base* [6] to just the client machine.

Although issues such as key derivation time and public storage requirements are largely irrelevant in practical terms, as we have explained in the previous sections, the private key redistribution required following key updates is a serious consideration in practical CPFS implementations. Although the scale of this problem is reduced in schemes such as the AFB KAS, it is still far more complex (from the scheme administrator's point of view) than simply removing a particular user's authorisation from the AS (which has no effect on other legitimate users).

Secure key distribution has always been a fundamental issue in applied cryptography. The development of asymmetric cryptographic techniques in the 1970s was expected to remove or at least simplify this problem. However, it has become apparent that the robust implementation of such techniques require a dedicated infrastructure to guarantee the authenticity of public keys. Public key infrastructures are costly to implement and manage. Our motivation is therefore to design a KAS with similar security and efficiency properties as existing schemes, but one which obviates all private key redistribution.

3 Avoiding Key Redistribution in KASs

We assume that each user $u \in U$ shares a symmetric key, $k(u)$, with the scheme administrator. This is a reasonable assumption, given that we must have *a priori* knowledge of all the users in the system in order for them to be assigned to security labels, and we would require some way of securely transmitting $\kappa(\lambda(u))$ to u for *any* KAS. In our scheme, we publish information tailored to allow each user to derive their permitted encryption keys. We can do this in several ways, balancing key derivation time with the public storage requirements, analogous to the tradeoff between the IKE and DKE KASs.

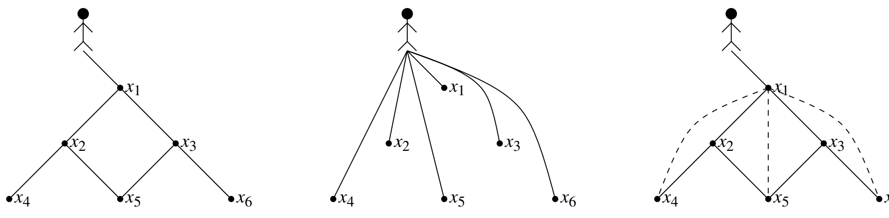
3.1 User-Based KASs

In all our schemes the set of encryption keys $\{\kappa(x) : x \in L\}$ and the set of private user keys $\{k(u) : u \in U\}$ can be chosen at random from some appropriate key space. The public information requirements of each scheme are described in the following sections.

User-based Iterative Key Encrypting KAS (UIKE KAS). As the name implies, this scheme is analogous to an IKE KAS as only strictly necessary edges are included in the public information. The "graph" of the scheme includes an edge from each user to their security level and the edges in the covering relation on L.

- $Pub_{\mathrm{UI}} = \{E_{k(u)}(\kappa(\lambda(u))) : u \in U\} \cup \{E_{\kappa(x)}(\kappa(y)) : y \lessdot x : x, y \in L\}$;
- The user first obtains $\kappa(\lambda(u))$ by decrypting $E_{k(u)}(\kappa(\lambda(u))) \in Pub_{\mathrm{UI}}$ with $k(u)$. Then,
  - if $y \lessdot \lambda(u)$, u can use $\kappa(\lambda(u))$ to obtain $\kappa(y)$ by decrypting $E_{\kappa(\lambda(u))}(\kappa(y)) \in Pub_{\mathrm{UI}}$;
  - if $y < \lambda(u)$, there exists a path $y = z_0 \lessdot z_1 \cdots \lessdot z_n = \lambda(u)$, $n \geqslant 1$, so that u can successively derive keys $\kappa(z_{n-1}), \ldots, \kappa(z_0) = \kappa(y)$.

In other words, the user uses their secret key to derive the key to their particular security level and then traverses the graph, decrypting the key for each node in turn to obtain the desired key, as illustrated in Fig. 2(a).



(a) UIKE KAS (b) UDKE KAS (c) HKE KAS

Fig. 2. The directed graphs used in our user-based key assignment schemes: the transitive relationships in the partially ordered set are denoted by broken lines; the user has security label x_1

User-based Direct Key Encrypting KAS (UDKE KAS). This scheme is analogous to a DKE KAS as the user can derive any (permitted) key in a single step. The "graph" of the scheme comprises an edge from each user to each of the security levels for which they are authorised.

- $Pub_{\mathrm{UD}} = \{E_{k(u)}(\kappa(y)) : y \leqslant \lambda(u) : u \in U, y \in L\}$;
- $\kappa(y)$ is obtained by decrypting $E_{k(u)}(\kappa(y)) \in Pub_{\mathrm{UD}}$ using $k(u)$.

This scheme allows the user to derive any (permitted) key in a single step, as demonstrated in Fig. 2(b). Although this is a desirable feature, the size of Pub_{UD} is potentially very large.

Hybrid Key Encrypting KAS (HKE KAS). This scheme is a compromise between the previous two, balancing key derivation steps with public storage requirements.

- $Pub_{\mathrm{H}} = \{E_{k(u)}(\kappa(\lambda(u))) : u \in U\} \cup \{E_{\kappa(x)}(\kappa(y)) : y < x : x, y \in L\}$;
- The user first obtains $\kappa(\lambda(u))$ by decrypting $E_{k(u)}(\kappa(\lambda(u))) \in Pub_{\mathrm{H}}$ using $k(u)$. Having derived $\kappa(\lambda(u))$, he can then obtain any $\kappa(y), y < \lambda(u)$, by decrypting $E_{\kappa(\lambda(u))}(\kappa(y)) \in Pub_{\mathrm{H}}$.

Informally, it takes one derivation to "hop" onto the graph of $(L, \leqslant)$, from where any (authorised) node can be reached in one step, as shown in Fig. 2(c).

3.2 Performance Evaluation

Key Derivation Time. The UDKE KAS retains the advantage of the DKE KAS in that key derivation still requires a single computation. The UIKE KAS adds one extra step to the original IKE KAS key derivation. The HKE KAS generally takes two steps for any node (the exception being the key for the user's own security level), making the key derivation time approximately equal to the derivation time for the UDKE KAS. When we compare like with like (that is, UIKE KAS with IKE KAS, etc.) there is no significant difference in key derivation time between the original and the new user-based schemes.

Public Storage Costs. There is no doubt that our schemes require more public storage than the original schemes. If we have n users and l security levels, and assume that $n \gg l$, then the UIKE KAS public storage is $\mathcal{O}(n+e) = \mathcal{O}\left(n+l^2\right) = \mathcal{O}(n)$, compared with the original IKE KAS storage of $\mathcal{O}(e)$. The UDKE KAS fares even worse, requiring $\mathcal{O}(nl)$ edges, as opposed to $\mathcal{O}(e^*) = \mathcal{O}\left(l^2\right)$ for the original DKE KAS, (a significant difference, given that $n \gg l$). This increase in the size of Pub_{UD}, was the main motivation for the introduction of the HKE KAS; the size of Pub_{H} is $\mathcal{O}(n+e^*) = \mathcal{O}(n)$.

If we consider an extreme example in which L is a total order containing 100 security levels with 1000 users assigned to each level, the number of edges required for the HKE KAS is proportional to $\frac{1}{2}99.100 + 100000 \simeq 10^5$. If we use 256 bit keys/edges, this implies storage space requirements of ~ 3 MB. Whereas this could be significant in some systems, if we consider that the point of our entire scheme is to facilitate access to a remote file server, and would expect a file system of a size vastly exceeding this 3 MB cache to be stored, it becomes clear why we believe the cost of storing Pub_{H} is negligible in practical terms. The storage requirements for the UIKE KAS are proportional to $99 + 100000 \simeq 10^5$, so there is almost no advantage to be gained by using the UIKE KAS. The public storage requirements for UDKE KAS are proportional to $1000(100 + \cdots + 1) \simeq 5 \times 10^6$ edges, corresponding to storage requirements of $\sim$ 150 MB. Even by modern standards, storing this amount of public information, even on a dedicated file server, may not be considered unimportant.

The increase in public storage for UDKE KAS may seem important when we consider the relative sizes of the public information for each scheme: however, as we have explained in Sect. 2.2, the relevance of this performance parameter is doubtful given the applications to which we expect such schemes to be applied. Nevertheless, although we believe that the size of Pub_{UD} is likely to be acceptable for most applications, we would generally recommend the HKE KAS, with its vastly reduced storage requirements achieved at the cost of merely an extra key derivation step.

Administrative Complexity. Whether through key compromise, user revocation or routine security measures, it will inevitably be necessary to update the keys used to encrypt the data on the file server. Of these, the most complex procedure from an administrative point of view is handling the revocation of a user u_{R} with a security label $\lambda(u_{\mathrm{R}}) = x$, say. We write $\uparrow x$, to denote the set $\{y \in L : x \leqslant y\}$, we write $\downarrow x$, to denote the set $\{y \in L : x \geqslant y\}$, and we write $\kappa(\downarrow x)$ to denote $\{\kappa(y) : y \leqslant x\}$. Since we must assume that u_{R} has cached a copy of all the keys to which he had been previously entitled, we must update all $\kappa(\downarrow x)$ (and re-encrypt all affected objects).

In a standard IKE KAS or DKE KAS, this has the effect of changing all the edges incident to any of the affected nodes, meaning changes to the public information. Far more importantly, from a practical perspective, the private key for any $u \in U$ such that $\lambda(u) \leqslant x$ has to be updated, thereby requiring key redistribution.

If we now consider our schemes, the keys $\kappa(\downarrow x)$ still have to be updated, the data objects re-encrypted and the relevant edges updated (with precisely the same users affected) but, crucially, no key (re)distribution is required. The administrator simply doesn't update (or deletes) the entries $E_{k(u_{\mathrm{R}})}(\kappa(\downarrow x)) \in Pub_{\mathrm{UD}}$ in a UDKE KAS or $E_{k(u_{\mathrm{R}})}(\kappa(x)) \in Pub_{\mathrm{UI}}$ (respectively Pub_{H}) in a UIKE KAS (HKE KAS) meaning that u_{R} can no longer derive the new keys and the revocation is accomplished.

3.3 Discussion

In all our schemes, the burden of administration has shifted from the difficult task of redistributing new private keys to geographically distributed users in a secure, confidential manner, to the relatively simple task of updating encrypted information held on a public server. It is this crucial point that differentiates our schemes from all previous KASs in the literature. Although our schemes are based on existing generic constructions, and our ideas can probably be used to improve other schemes, we feel that our contribution is to change the way KASs are generally thought about by researchers and system designers; we anticipate that our ideas are likely to have the most impact on the implementation of KASs.

The only remaining concern is how to ensure that at least the authorised users have the necessary public information when they need it. We feel that a *pull* model, whereby users retrieve their data (edges) as and when they require, is particularly suitable, bearing in mind the system under consideration (users requesting objects), rather than a *push* model, where the central administrator seeks to actively broadcast any data changes to users who may not be online. In fact, the edges could be stored on the resource server itself, with the file system calls modified to return the object and the necessary edge(s), reducing overheads still further.

4 Is HKE KAS the Best Two-Step Scheme?

A number of attempts have been made to find a trade-off between the number of key derivation steps required and the amount of public storage, see [5] for example. The general approach is to construct a new graph $G = (L, R)$, where $R \subseteq L \times L$ and for all $x, y \in L$ such that $y \leqslant x$, there is a path no longer than p (for some fixed integer p) between x and y in G. Then the public information is defined to be $\{E_{\kappa(x)}(\kappa(y)) : (x, y) \in R\}$.

In this section we consider applying this kind of approach to our user-based KASs. In particular, we investigate whether there exists a two-step KAS with smaller public storage requirements than the HKE KAS. In the HKE KAS, the amount of public storage is proportional to the sum of the "internal" edges arising from the DKE KAS used for L and the "external" edges arising from the edges between each user and his security level. As we have already noted, the amount of public storage is therefore $\mathcal{O}\left(l^2 + n\right)$, where $n = |U|$.

What we now try to do is use analogous methods to reduce the number of internal edges at the expense of a small increase in the number of external edges. Suppose, then, that we can partition $(L, \leqslant)$ into $L_1, \ldots, L_m$ such that each L_i has a unique maximal element (with respect to the original partial order). We call these maximal elements *anchor points*, and denote the set of anchor points by $A \subseteq L$.

We can define a DKE KAS for each sub-poset $(L_i, \leqslant)$ with public information Pub_i. Finally, we can construct a user-centric KEKAS with the following properties:

- the public information contains $\bigcup_{i=1}^{m} Pub_i$;
- for each user u there is an element of public information for the pair $(u, \lambda(u))$ and for every pair (u, a) such that $a \in A$ and $a \leqslant \lambda(u)$.

Then a user can get to any anchor point $a_i \leqslant \lambda(u)$ in one hop and to any other node in L_i in a further hop. Hence, a user can derive the key for any $y \leqslant \lambda(u)$ in two steps. Writing l_i for $|L_i|$, the public storage required for each scheme is

$$\mathcal{O}\left(\sum_{i=1}^{m} l_i^2 + \sum_{u \in U} |{\downarrow}(\lambda(u)) \cap A|\right).$$

A formal comparison of the storage requirements for the standard HKE KAS and the modified scheme proposed above is beyond the scope of this paper. However, the following provides some justification for believing that in most practical instances, the modified scheme is unlikely to better than the HKE KAS.

In the general case, we note that the reduction in the number of internal edges is fixed (assuming that L and $L_1, \ldots, L_m$ are fixed). In any "optimised" two-step solution, adding a single user must add more than one external edge (on average) in order to allow the user to reach every relevant anchor point. While the HKE KAS may well have more internal edges for a small number of users, adding a user only ever requires a single external edge to be added. It is clear then, that in a realistic application, where users will significantly outnumber security levels, so that the total number of edges in the scheme is dominated by the number of external edges, the HKE KAS will be an optimal solution. We include a simple example in Appendix A to demonstrate this.

5 Related work

5.1 Existing KASs

We refer the interested reader to a recent paper for a comprehensive survey of existing KASs in the literature [2], and note that none of these schemes eliminate private key redistribution. Atallah et al. touch upon the idea of eliminating key redistribution [5, Section 5], noting that this could be achieved in principle, but choose to focus their research entirely on other extensions such as trying to optimise the number of edges required (which we discuss in the next section).

5.2 Optimised KASs

In our evaluation of KASs (Sect. 2.2) we only considered the DKE KAS and IKE KAS. We can view these schemes as representing the extreme points on a continuum of possible key encrypting KASs (KEKASs). The IKE KAS represents the KEKAS with the minimum possible storage requirements, at the expense of maximising the average key derivation time. The DKE KAS represents the other end of the scale, with a single-step key derivation achieved at the expense of maximising the public storage requirements.

Recent work by Atallah et al. [7] has developed methods to construct optimal p-step KEKASs (as described briefly in Sect. 4). Whilst their work is intriguing from an abstract mathematical point of view, we believe it has only limited applicability in terms of finding an "optimal" KEKAS (OKE KAS).

In order to develop an OKE KAS, it is first necessary to find a meaningful way to compare the various possibilities. Unfortunately, it is not at all clear how to compare, say, a four-step solution with its particular storage requirements to, say, a three-step solution which has greater storage needs.

More importantly, as we have explained, the trade-off between key derivation time and public storage is not very important in practical terms. In any given organisation, the actual gains made by using an OKE KAS over either a IKE KAS or a DKE KAS, will be negligible, since neither of these factors are likely to be of any practical significance.

Finally, in almost all situations, system designers are guided by the principle of trying to maximise the runtime performance. Slightly counter-intuitively, the greater the number of edges the organisation stores publicly, the smaller the number of edges each user has to retrieve. The overhead is shifted from a runtime performance hit to an off-line computation by the administrator. Thus, even if a suitable objective function could be devised and a cost-benefit analysis conducted, the DKE KAS is still overwhelmingly likely to be selected as the best overall option, as it requires the user to download the minimum possible number of edges and perform just a single computation at runtime in order to obtain any (permitted) key.

In the previous section, we introduced our new, user-based schemes and pointed out that we favoured the two-step HKE KAS over the single-step UDKE KAS, which would clearly be faster at runtime. The reason for this discrepancy is straightforward. The public storage requirements in the user-based schemes scale with $n = |U|$, as opposed to the original schemes which scale with $l = |L|$. Since we expect that in most organisations $n \gg l$, the size of Pub_{UD} can become sufficiently large for it to be considered non-trivial (as explained in Sect. 3.2). Once this happens, it is necessary to consider ways of reducing the requirements; the HKE KAS provides a natural way to construct a two-step solution which significantly reduces the public storage requirements.

6 Conclusion and Future Research

We have introduced three new KASs, the UIKE KAS, the UDKE KAS and the HKE KAS. In this work, we have developed a unique, pragmatic approach to evaluating KASs, ignoring the negligible public storage requirements and key derivation times, focusing instead on the issue of private key updates and the resulting requirement for key redistribution. We believe this single issue to be overwhelmingly the most important factor in any KAS analysis.

Our schemes address this point by eliminating all private key updates, transforming the problem of key redistribution following key updates into a simple data update process. Moreover, our ideas can be implemented on top of any existing provably secure KAS, thus automatically inheriting the same security guarantees.

There is an interesting parallel we can draw between our ideas and the concept of a CPFS. In all remote file systems it is necessary to protect sensitive data by encrypting it, at least for transmission purposes. The CPFS approach merges the cryptography layer into the file system itself. Nothing is fundamentally changed by moving the protection mechanism, except that our approach to security shifts completely into the area of KASs, making sure that only authorised users have the necessary keys. By using this approach, we need no longer worry about the consequences of the server being compromised or the security of the communication channel between the client and the server, since the *trusted computing base* now comprises only the client machine.

In existing KASs, a secure, private key redistribution mechanism is assumed. This would almost certainly be in the form of a shared secret between each user and the scheme administrator, who would then send the new key to the user encrypted with the shared secret to maintain confidentiality.[2] Our contribution is to merge this key distribution mechanism into the KAS itself, by publishing the encrypted keys along with the rest of the public KAS information. Architecturally, little has fundamentally changed, but by using this approach, we reduce the overall complexity of the scheme, since all administrative tasks can be handled with simple updates to a public server.

In future work, we would like to implement our schemes, investigating whether our assumptions about performance characteristics are realistic. It is also worth investigating whether the HKE KAS is indeed an optimal two-step solution in realistic applications and the conditions under which it becomes optimal for arbitrary hierarchies and user distributions.

Another significant obstacle barring the widespread implementation of CPFSs, is the problem of mass re-encryption of data following updates to the encryption keys. Certain approaches have been proposed, including delaying the re-encryption, known as the *lazy update* approach [8]. In future work we would like to consider ways to avoid this problem entirely.

References

1. Fu, K., Kamara, S., Kohno, Y.: Key regression: Enabling efficient key distribution for secure distributed storage. In: Proceedings of the Network and Distributed System Security (NDSS 2006) (2006)
2. Crampton, J., Martin, K., Wild, P.: On key assignment for hierarchical access control. In: Proceedings of 19th Computer Security Foundations Workshop, pp. 98–111 (2006)
3. Denning, D.: A lattice model of secure information flow. Communications of the ACM 19(5), 236–243 (1976)
4. Davey, B., Priestley, H.: Introduction to Lattices and Order. Cambridge University Press, Cambridge, United Kingdom (1990)
5. Atallah, M.J., Frikken, K.B., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: Proceedings of 12th ACM Conference on Computer and Communications Security, pp. 190–202. ACM Press, New York (2005)
6. US Department of Defense: Trusted computer system evaluation criteria. Technical Report 5200.28-STD, DoD (1985)
7. Atallah, M.J., Blanton, M., Frikken, K.B.: Key management for non-tree access hierarchies. In: Proceedings of the 11th ACM Symposium on Access Control Models and Technologies, pp. 11–18. ACM Press, New York (2006)

[2] Of course this could also be achieved in other ways such as using asymmetric cryptographic techniques. Our point is to demonstrate the advantages of eliminating the requirement to maintain a separate private key distribution mechanism, regardless of how this is implemented.

8. Backes, M., Cachin, C., Oprea, A.: Secure key-updating for lazy revocation. In: Proceedings of 11th European Symposium on Research in Computer Security, pp. 327–346 (2006)

Appendix A: A Simple Example

Let us now apply the analysis of Sect. 4 to a simple example, in which L is a total order and equal numbers of users are assigned to each security label. Then we can divide L into k sub-chains of length $m = l/k$. The DKE KAS for each sub-chain requires $\frac{1}{2}m(m-1)$ edges. Hence there are $\frac{k}{2}m(m-1) = \frac{1}{2}l(m-1)$ internal edges in the new scheme. The number of internal edges for the standard HKE KAS is $\frac{1}{2}l(l-1)$. Hence the number of internal edges is reduced by $\frac{1}{2}l\left(l-1-(m-1)\right) = \frac{1}{2}l(l-m)$.

We now consider what extra external edges are needed in the new scheme. Users assigned to a label in the highest subchain have k edges (one for their security label and $k-1$ for each of the other anchor points). Similarly, users assigned to a label in the next highest subchain have $k-1$ edges. Hence, in total, we require n edges to connect each user to his respective security level and

$$\frac{n}{l}\sum_{i=1}^{k-1} i = \frac{n}{2l}k(k-1) = \frac{n}{2l}\frac{l}{m}\left(\frac{l}{m}-1\right) = \frac{n}{m^2}(l-m)$$

additional edges to connect users to relevant anchor points. The proposed scheme improves on the basic HKE KAS only if the number of additional external edges is less than the reduction in internal edges. Hence, we need to consider whether $\frac{n}{m^2}(l-m) < \frac{1}{2}l(l-m)$. We may assume that $l > m$ (otherwise $k = 1$ and the modified scheme is equivalent to the original HKE KAS). Hence, we have that the modified scheme offers an improvement if $lm^2 > n$. For most practical schemes, $n \gg l$ and this inequality is unlikely to hold. Now, the largest m can be is $l/2$, so there is only an improvement if $n < l^3/4$. If $l = 10$, for example, we only require 250 users for the optimal choice to be $k = 1$ (which represents the basic HKE KAS).

Fern : An Updatable Authenticated Dictionary Suitable for Distributed Caching

E. Freudenthal, D. Herrera, S. Gutstein, R. Spring, and L. Longpré

University of Texas at El Paso,
{efreudenthal,daherrera,smgutstein,rcspring,longpre}@utep.edu

Abstract. Fern is an updatable cryptographically authenticated dictionary developed to propagate identification and authorization information within distributed systems. Fern incrementally distributes components of its dictionary as required to satisfy client requests and thus is suitable for deployments where clients are likely to require only a small fraction of a dictionary's contents and connectivity may be limited.

When dictionary components must be obtained remotely, the latency of lookup and validation operations is dominated by communication time. This latency can be reduced with locality-sensitive caching of dictionary components. Fern dictionary's components are suitable for caching and distribution via autonomic scalable locality-aware Content Distribution Networks (CDNs) and therefore can provide these properties without requiring the provisioning of a dedicated distribution infrastructure. Competitive approaches require either the sequential transfer of two-to-three times more vertices or the replacement of a greater number of already distributed vertices when updates occur.

Keywords: binary trie, authenticated dictionary, distributed systems, content distribution network, Merkle tree.

1 Introduction

The maintenance of consistency between Certificate Authorities (CAs) and access controllers has been a persistent problem in distributed systems. A variety of approaches have been implemented including online validation, limitation of certificate lifetimes, and dissemination of certificate revocation lists. These approaches have complementary advantages, and hybrid implementations are common. For example, by issuing certificates with limited lifetimes, a CA can limit the number of unexpired certificates that access controllers must reject. This *certificate revocation list* (CRL) can be disseminated directly to all access controllers, or to a set of trusted proxies who provide online validation services. The dissemination of complete CRLs can impose onerous communication, storage, and computational costs to access controllers that only reference a small subset of CA's certificates. Similarly, the provisioning of trusted proxies increases the communication and computational cost, and possibly the latency of authorization decisions. Furthermore the determination of whether an online software system has security properties suitable for secure online transactions is notoriously difficult.

Distributed updatable online authenticated dictionaries based on skiplists[4,1] and self-balancing trees[9] have recently been proposed as a source of authorization information in distributed systems. These structures disseminate name-to-value mappings that can provide evidence of identity (i.e., that some public key identity K_A is an Id associated with a person named "Alice") or that a set of certificates has been revoked. Composed cryptographic hashes are embedded within these dictionaries, permitting their distribution via untrusted proxies to access controllers who can efficiently validate mappings or determine their absence. The integrity of any search path within these data structure can be verified with a single public-key root certificate. Furthermore, these partial copies of the dictionary's structure can be cached and therefore utilized to answer future queries that share common paths. These properties can be exploited by proxies and security-sensitive clients that incrementally obtain and validate portions of their search structure as required to answer queries.

The latency of a search within a distributed authenticated dictionary is dominated by the latency of obtaining vertices in a search path. So, it is useful to minimize these *necessarily serialized* operations. Path length in a search tree corresponds to leaf depth. Expected leaf depth is $Ulog_bN$ where b is the tree's branching factor. U (typically below 2) accounts for the structure's expected lack of balance.

Modifications to mappings stored within an authenticated dictionary typically cause changes to the dictionary's search structures. Since components of an authenticated dictionary may be cached, it is desirable if these updates are limited to the ancestors of the changed components and thus require only a few updates. Thus, algorithms that rely on structural modification to achieve good performance (e.g. self-balancing trees) reduce this cacheability. Updates to a randomized skiplist do not result in structural changes unrelated to the updated vertex's original search path, but the expected number of objects in a skiplist's search path is $3log_2N$ for a skiplist containing N items.

Fern utilizes a randomized search structure with empirical path lengths of $1.1 \log_2 n$. Given the high latency of inter-host communication in distributed systems, *the constants do matter*, and the number of vertices that must be transferred to lookup a value stored within Fern is comparable to the most aggressive algorithms based upon self-balancing trees and far lower than skip-lists. Like skiplists, but unlike balanced trees, Fern does not require restructuring and thus is also well suited for distributed caching.

2 Fern

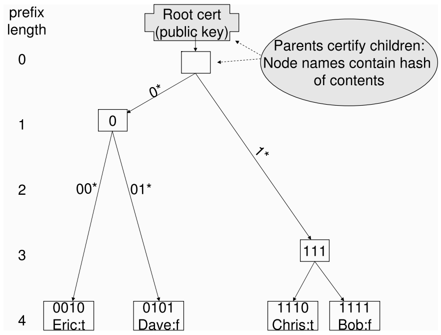
As illustrated on the left side of Fig. 1, Fern's internal structure is a binary Merkle-trie with path compression. Like the authenticated authorization framework suggested by Tamassia et. al. in [9], vertices from Fern's Merkle-trie are suitable for distribution by peer-to-peer (P2P) Content Distribution Networks (CDNs) to clients and access controllers who construct, maintain, and validate search paths that serve as evidence of authorizations upon which they depend.

Following the notation of Martel et. al.[5], we refer to validated search paths as *verification objects* (VO).

ID:value mappings are stored in leaf vertices whose search key is equal to its ID's SHA-1 hash (when used to disseminate set membership, the *value* field can be empty.) Following the model of a Merkle-tree[7], internal nodes also contain SHA-1 hashes of each immediate descendant (hashes are not indicated in Figure 1). Incorporation of hashes within all internal verices permits access controllers to to validate the integrity of individual search paths without obtaining the entire trie. The current root and its hash is stored in a *root certificate* signed with the originator's public key.

A Fern VO for a particular ID contains a root certificate and the search path containing its mapping. If the ID is not stored within Fern, it contains a search path that demonstrates that the referenced ID is not in the trie.

Fern publishes root certificates and vertices using the CoralCDN scalable, self-organizing, and locality-aware content distribution network. This helps to minimize network congestion that arises from distributing vertices and permits a large number of access controllers to be served by a single server with modest computational and network resources.



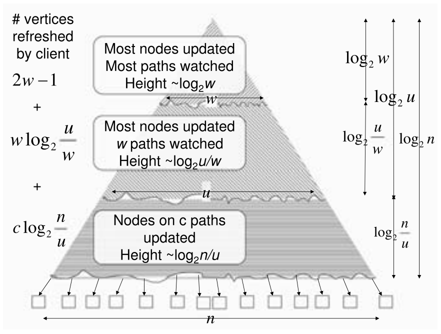

# vertices refreshed by client
$2w-1$
+
$w\log_2\frac{u}{w}$
+
$c\log_2\frac{n}{u}$
Most nodes updated Most paths watched Height ~$\log_2 w$
w
Most nodes updated w paths watched Height ~$\log_2 u/w$
u
Nodes on c paths updated Height ~$\log_2 n/u$
n
$\log_2 w$
$\log_2 u$
$\log_2\frac{u}{w}$
$\log_2 n$
$\log_2\frac{n}{u}$

Fig. 1. On left: Fern trie for 4-bit keys containing leaves (0010, 0101, 1110, and 1111). On right: Number of nodes that need to be updated when the number of watched data entries is smaller than the number of updated data entries (see Section 3).

In contrast to Fern, which uses 160-bit SHA-1 hashes, the trie depicted in Figure 1 only uses four-bit search keys. ID:value mappings are stored in leaf vertices with search-keys corresponding to the ID's hash digest. In this figure, the name "Bob" (mapped to f) is depicted as having a hash equal to 1111.

Routing decisions within this trie are made one bit at a time, in decreasing order of significance. Each internal vertex is labeled with the search prefix corresponding to the common prefix shared by all its descendants. Vertices within tries that have no branching are collapsed as is illustrated by the absence of vertices corresponding to the prefixes 1, 10, 11, and 110.

A client with interest in an ID's mapping stored within Fern obtains the set of trie nodes that comprise the path from the root to the desired leaf. Monte Carlo experiments indicate that leaves within Fern's binary trie are generally at a depth of $1.1 \log_2 n$ where n is the number of mappings stored within the trie, equal to the most agressive self-balancing algorithms.

Each internal vertex of a Fern authorization trie contains node identifiers (NIDs) of its children. NIDs are used as a locator by clients requesting particular nodes. Like SFS-RO's[3] self-certifying pathnames, Fern's NIDs include cryptographic hashes of the referenced node's contents and are also used to verify the integrity of a nodes obtained from insecure channels.

A Fern CA periodically publishes a signed, time-limited *root certificate*. Using an asymmetric cipher Fern clients and access controllers know the CA's public-key identity and use it to determine the integrity of root certificates.

A VO for a referenced identifier, I, thus contains all Fern-nodes between the Fern-root and I. A full certification path can be included within the payload of a single message transmitted from one Fern client to another. Alternatively, since each internal node of a Fern-trie contains the NIDs of immediate children serving as locators, a Fern client can obtain a certification path for I by obtaining intermediate nodes as it traverses the path from the root toward the leaf node containing I. A search-path that comprises a complete VO can be transmitted directly between clients, or instead be constructed by a client that searches for the node corresponding to the ID's hash.

3 Analysis: Number of Refresh Queries

Fern search-tries are well balanced due to their use of a good hash function (e.g. SHA-1) to evenly distribute a set of elements over an ordered range which is large compared to the number of elements being distributed.

The hash value of a Merkle-tries's leaf and all of its ancestors changes whenever the leaf's contents are updated. In this section, we consider the number of vertices that must be obtained by a client who is monitoring mappings stored within a set of $w = |W|$ *watched* leaves within a dictionary storing a total of n mappings in the event that $u = |U|$ leaves are *updated*. Let $c = |U \cap W|$ be the number of vertices in W whose mappings have changed. Assume that $w \leq u$ (this should be commonly the case.) Monte Carlo experiments support these results, especially as w, u and n become large. A technical report version of this paper [2] provides a more formal analysis and also examines the case where $w > u$.

As illustrated on the left side of Figure 1, our analysis divides the trie into three regions. The upper region, which extends to depth $\lfloor \log_2 w \rfloor$ has approximately w vertices at its (approximate) lower edge. We assume that the hash function uniformly distributes keys, that most of these (approximately) w vertices are ancestors to the w leaves mapping members of W. Since $u \geq w$, it is likely that most of these (approximately) w vertices members are also ancestors of the members of $|U|$. Thus our client is likely to require updates for all (approximately) $2w - 1$ vertices in the upper region.

The middle region of this trie extends to depth $\lfloor \log_2 u \rfloor$. Like our analysis of the upper region, it is likely that most vertices above depth $log_2 u$ contain nodes that are ancestors of the u updated leaves and most vertices in this region are likely to have been updated. However, only w paths of length $\lfloor \log_2(u/w) \rfloor$ through this middle region are likely to be ancestors of vertices being *watched*, and only $w\lfloor \log_2(u/w) \rfloor$ of these vertices are likely to be needed by the client.

The lower region extends down from (approximately) level $\lfloor \log_2 u \rfloor$ and has approximately $\lceil \log(n/u) \rceil$ levels. However, the approximately $c\lceil \log_2(n/u) \rceil$ vertices along the c paths to leaves that are both watched and updated will be obtained by clients. Thus we expect that approximately $(2w-1)+w\lfloor \log_2(u/w) \rfloor + c\lceil \log_2(n/u) \rceil$ nodes that will require updating. In practice, the shift from one section to the next may not occur exactly at the same level on every path. The analysis for $w > u$ is symmetric, so the same result holds with a switch between w and u.

We have begun evaluation of Fern upon the Planetlab[8] global distributed testbed. The plot on the left side of Figure 2 conducted upon hundreds of hosts distributed globally provides empirical evidence supporting our analytical model of refresh queries. In this experiment, $k = 2058$, $w = 60$, and $u = 200$. Each host's set of watched mappings was chosen with a distribution corresponding to the routing table of a Kademlia[6] DHT. Plots indicate the average number of Fern vertices obtained for clients distributed globally upon Planetlab. As predicted by the analytical model, refresh queries tend to terminate around depths $\log_2 u = 7$ and $\log n = 11$. In these experiments, the average number of vertices obtained by clients from CoralCDN is 154. Our analytical approximation prediction is 146.

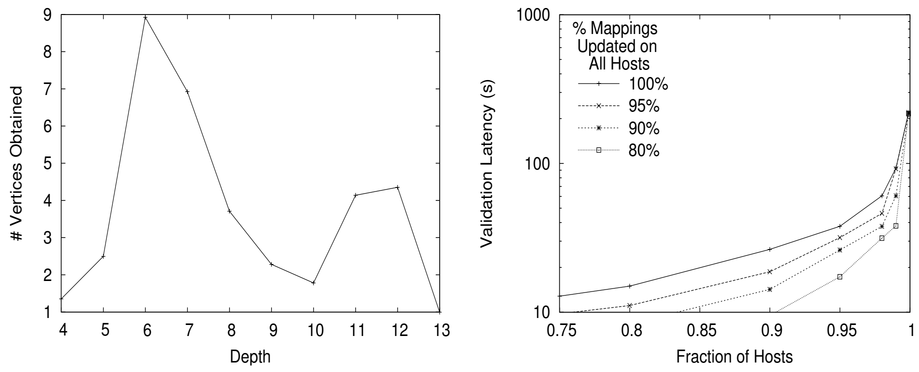


Fig. 2. Measured results from Executions on Planetlab: Left side: Average Depth of Refresh Queries. Right side: Cumulative distribution of refresh query times.

The plot on the right side of Figure 2 depicts the cumulative distribution of update latencies for watched mappings in an experiment with 324 hosts distributed in 37 countries. Fern vertices were disseminated using CoralCDN. In this experiment $n = 2048$ and each host watched $w = 37$ mappings chosen with mapping corresponding to its view of a Kademlia routing table. Roots were published at five minute intervals, and thirty-two mappings were changed during each interval. Each plot depicts the cumulative distribution of times required for a fraction of the hosts to obtain a fraction of the updated mappings. Note that 0.9 (90%) of the hosts obtained 90% of their 37 watched mappings in $\sim$ 10s, and 0.95 of the hosts obtained all of their 37 watched mappings in $\sim$ 30s.

4 Conclusion

We have described a data structure for updatable authenticated dictionary with low maintenance and with better latency than current techniques.

Acknowledgments. This work was supported in part by the U.S. Army Research Laboratory, Survivability/Lethality Analysis Directorate, Information and Electronic Protection Division, Information Warfare Branch.

References

1. Anagnostopoulos, A., Goodrich, M., et al.: Persistent authenticated dictionaries and their applications. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 373–393. Springer, Heidelberg (2001)
2. Freudenthal, E., Herrera, D., et al.: Fern: An updatable authenticated dictionary suitable for distributed caching. Technical Report 06-45, Computer Science Department, University of Texas at El Paso (2006)
3. Fu, K., Kaashoek, M.F., Mazieres, D.: Fast and secure distributed read-only file system. Computer Systems 20(1), 1–24 (2002)
4. Goodrich, M., Shin, M., Tamassia, R., Winsboro, W.: Authenticated dictionaries for fresh attribute credentials. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 332–347. Springer, Heidelberg (2003)
5. Martel, C., Nuckolls, G., et al.: A general model for authenticated data structures. Technical Report CSE-2001, Stubblebine Labs (2001)
6. Maymounkov, P., Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
7. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (1990)
8. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services, `http://planet-lab.org`
9. Tamassia, R., Triandopoulos, N.: Efficient content authentication over distributed hash tables. Technical report, Brown University (2005)

Class of Provably Secure Information Authentication Systems

N.A. Moldovyan and A.A. Moldovyan

Specialized Center of Program Systems “Spectr”,
Kantemirovskaya Str. 10, St. Petersburg 197342, Russia
nmold@cobra.ru,
www. cobra.ru

Abstract. Electronic messages authentication issue is of significant importance for computer systems. A number of public key cryptosystems based on the composite modulus ($n=pq$, where p and q are large primes) has been proposed to provide information authentication and only for one of them (that has been proposed by M. Rabin) security has been proved formally. In this paper we generalize the M. Rabin’s public key encryption and digital signature schemes and present formal proof of the security of the class of public key cryptosystems based on difficulty of the factorization problem.

Keywords: Information authentication, provably secure cryptosystems, digital signature, public encryption, public key cryptosystem.

1 Introduction

Systems designed for information authentication in computer networks and information systems are based on public key cryptosystems. The RSA public key cryptosystem introduced by R.L. Rivest, A. Shamir, and L.M Adleman in 1978 [1] has become the first world wide used digital signature and public-key encryption system. In RSA the public key is represented by pair of numbers (n, e), where $n = pq$ is the product of two randomly chosen distinct prime numbers and e is a random number that is relatively prime with Euler phi function $\varphi(n) = (p - 1)(q - 1)$. The triple ($p$, q, d), where $d = e^{-1} \bmod \varphi(n)$, is secret. Data ciphering with RSA is described as follows: $C = M^{e} \bmod n$ (public-key encryption) and $M = C^{d} \bmod n$ (decryption), where $M < n$ is a plaintext and C is ciphertext. RSA signature (S) generation and verification are performed as follows: $S = M^{d} \bmod n$ and $M = S^{e} \bmod n$, correspondingly.

Usually the signed documents are comparatively long. In such cases instead to sign a document M we sign the hash function value $H = F_H(M)$ corresponding to M: $S = H^{d} \bmod n$. The RSA security is based on difficulty of factoring modulus n, which depends on the structure of primes p and q. At present the requirements to the primes p and q are well clarified [2, 3]. However there exists a fundamental problem concerning the RSA and many other public key cryptosystems, which consists in strict formal proof of their security.

Among different cryptosystems [1,4,5] based on difficulty of factorization problem there is known only one of them that is provably secure. It has been proposed by M. Rabin [4]. Data ciphering with M. Rabin's cryptosystems is described as follows:

$$C = M^2 \bmod n \quad \text{(public-key encryption)} \quad \text{and} \quad M = \sqrt{C} \bmod n \quad \text{(decryption)},$$

where $M \in \{M_1, M_2, M_3, M_4\}$ is a plaintext, M_1, M_2, M_3, M_4 are four different values of the square root modulo n, C is the ciphertext. Peculiarity of M. Rabin's public key cryptosystem is the four different variants of the $\sqrt{C}$ value from which the correct plaintext should be selected. Procedures of M. Rabin's signature (S) generation and verification are performed, correspondingly, as decryption and encryption:

$$S = \sqrt{M} \bmod n \quad \text{(generation)} \quad \text{and} \quad M = S^2 \bmod n \quad \text{(verification)}.$$

In this paper we generalize the M. Rabin's cryptosystem and show existence of a class of provably secure cryptosystems based on difficulty of factorizing the composite modulus.

This work is organized as follows: in Section 2 a class of digital signature scheme based on difficulty of factorization problem is described. In such cryptosystems the public encryption and digital signature verification consists in exponentiation of the message M to the kth power modulo n, where k is a natural number such that k divides $\varphi(n)$. In Section 3 we prove formally that the proposed cryptosystems are as secure as difficulty of the modulus factorization. In Section 4 we consider the most computationally efficient system that relates to the case $k = 3$. Finally, conclusions are presented in Section 5.

2 Class of Public Key Cryptosystems

In the considered class of the public key cryptosystems we use a modulus of the form $n = pq$, where p and q are strong primes that are easy to be generated using Gordon's algorithm [2,6]. The primes p and q are supposed to be of large size $|p| \approx |q| \geq 512$ bits, where $|W|$ denotes the binary representation length of the integer W. Gordon's algorithm allows to generate strong primes p and q for which the numbers $p - 1$ and $q - 1$ contain different large prime devisors γ' and γ'', respectively (for example, primes γ' and γ'' that have length from 160 to 384 bits). The public key is the number n. The secret is the (p, q) pair of numbers.

In this section we consider the Rabin-like cryptosystems in which the data ciphering is performed as follows:

$$C = M^k \bmod n \quad \text{(public-key encryption)} \quad \text{and} \quad M = \sqrt[k]{C} \bmod n \quad \text{(decryption)},$$

where $M < n$, $M \in \{M_1, M_2, \ldots, M_z\}$ is a plaintext, $M_1, M_2, , \ldots, M_z$ are different values of the kth degree root modulo n, C is the ciphertext, and k divides at least one of the two values $p - 1$ and $q - 1$. Thus, we should select one correct value of the plaintext from z different variants. The signature (S) generation and verification are performed, correspondingly, as decryption and encryption:

$$S = \sqrt[k]{M} \bmod n \quad \text{(generation)} \quad \text{and} \quad M = S^k \bmod n \quad \text{(verification)}.$$

In order to provide uniqueness of the decryption procedure result it is possible to use the redundancy of the plaintext or to append a checksum $G=F(M')$, where F is a specified function, to the message M'. In the last case the encrypted plaintext has the structure $M = M' \| G$, where $\|$ denotes the concatenation operation. The last method is also efficient, while generating the digital signature. In practical applications it is supposed that the signature should be calculated as $S = \sqrt[k]{M' \| G} \bmod n$ and signature verification should contain the following two steps: i) calculate $M' \| G = S^k \bmod n$ and ii) check the equality $G=F(M')$. We also should take into account that in general case the $M' \| G$ value is not residue of the kth degree modulo n. To overcome this problem it is possible to use different approaches. For example, a random 16-bit number R can be concatenated to the $M' \| G$ value. With probability $1/z$ the $M' \| G \| R$ value is residue of the kth degree modulo n. On the average, trying z different values R we will have possibility to generate a signature $S = \sqrt[k]{M' \| G \| R} \bmod n$ corresponding to the message M'. (If the M' message length is large, then we will sign the hash function value $H' = F_H(M')$: $S = \sqrt[k]{H' \| R} \bmod n$.)

Security of the considered class of cryptosystems is based on secrecy of the primes p and q. Signature generation and decryption procedures can be easily performed using the values p and q. However for a person that does not know the secret correct decryption or signature formation are computationally infeasible procedures.

Suppose $t = \gcd(p-1, k)$ and $u = \gcd(q-1, k)$. In this case we have t different values of the kth degree root modulo p: $\{M^{(p)}_1, M^{(p)}_2, \ldots, M^{(p)}_t\}$ and u different values of the kth degree root modulo q: $\{M^{(q)}_1, M^{(q)}_2, \ldots, M^{(q)}_u\}$. Each pair $(M^{(p)}_i, M^{(q)}_j)$, where $i \in \{1, 2, \ldots, t\}$ and $j \in \{1, 2, \ldots, u\}$, defines a value of the kth degree root modulo n which can be calculated using the Chinese Remainder Theorem, i. e., in the considered particular case, using the following formula:

$$M_s = \left[q\left(q^{-1} \bmod p\right) M_i^{(p)} + p\left(p^{-1} \bmod q\right) M_j^{(q)} \right] \bmod n \,. \qquad (1)$$

Thus, in general case we have $z = tu$ different values of the kth degree root modulo n: $\{M_1, M_2, \ldots, M_z\}$. There are known computationally efficient algorithms to calculate roots modulo primes. Sufficiently efficient procedures correspond to the case

$$p \equiv (t^2 - t + 1) \bmod t^2 \quad \text{and} \quad q \equiv (u^2 - u + 1) \bmod u^2.$$

The most efficient procedures of finding the kth degree root modulo p and modulo q correspond to the case $t = u = k$ (i. e. we have $k \mid p-1$ and $k \mid q-1$) and

$$p \equiv q \equiv (k^2 - k + 1) \bmod k^2.$$

If we generate and use the secret primes that relates to the last case, then we will have possibility to find the kth degree root modulo p and modulo q using, respectively, the formulas:

$$\sqrt[k]{a} = a^{\frac{p+k-1}{k^2}} \bmod p \quad \text{and} \quad \sqrt[k]{a} = a^{\frac{q+k-1}{k^2}} \bmod q \,,$$

where a is the kth degree residue modulo p and modulo q. If $t = \gcd(k, p-1) = 1$ or $u = \gcd(k, q-1) = 1$, then we can calculate integer $x = k^{-1} \bmod p-1$ and $\sqrt[k]{a} = a^x \bmod p$ or integer $x' = k^{-1} \bmod q-1$ and $\sqrt[k]{a} = a^{x'} \bmod q$, respectively.

3 Provable Security

Security of all public key cryptosystems described in Section 2 is based on difficulty of factorizing the modulus n. Provable security of this class of encryption and authentication systems is defined by the following theorem:

Theorem 1. *Security of the cryptosystems included in the class described in Section 2 is polynomially equivalent to difficulty of factorizing modulus n.*

The theorem states that breaking any of the considered cryptosystems is polynomially as hard as difficulty of factorizing modulus. To prove Theorem 1 means to show that i) the factorization of the modulus n provides possibility to decipher cryptograms and to sign messages using a polynomial algorithm and ii) an attack breaking a cryptosystem corresponding to the considered class provides possibility to factorize the modulus n using an algorithm complexity of which depends polynomially on both the difficulty of the attack and the value n.

The first part of the proof is evident. Factorizing n we get secret that provides possibility to decrypt ciphertexts and to sign messages. Both of these procedures are performed with polynomial algorithms, i. e. algorithms having polynomial complexity. Let us prove the second statement.

Proff of case ii). Suppose we know an algorithm for breaking a cryptosystem from the considered class. This means that for arbitrary value a that is the kth degree residue modulo n we can calculate at least one of the kth degree roots modulo n. The last value can be used to factorize modulus n with a polynomial algorithm as follows.

1. Select an arbitrary value M_s and calculate $a = M_s^k \bmod n$.
2. Calculate $M'_{s'} = \sqrt[k]{a} \bmod n$. With high probability $M'_{s'} \neq M_s^k$ (probability of this event is equal to $1 - z^{-1}$). If $M'_{s'} = M_s^k$, then go to step 1.
3. Calculate $\gcd(M'_{s'} - M_s, n)$. With high probability $\gcd(M'_{s'} - M_s, n) = p$ or $\gcd(M'_{s'} - M_s, n) = q$. (Probability that $\gcd(M'_{s'} - M_s, n) \neq 1$ is equal to $(t+u-2)z^{-1}$). If $\gcd(M'_{s'} - M_s, n) = 1$, then go to step 1.

Thus, to factorize n we should on the average execute the algorithm about $z(t+u-2)^{-1}$ times, i. e. complexity of factorizing n can be polynomially expressed via complexity of the considered attack and the value n. Finally, we should prove the formulas used at step 3 of the algorithm. At this step we have two different roots of the kth degree modulo n: $M'_{s'}$ and M_s. Accordingly to (1) these roots can be presented as follows:

$$M_s = \left[q\left(q^{-1} \bmod p\right) M_i^{(p)} + p\left(p^{-1} \bmod q\right) M_j^{(q)} \right] \bmod n, \qquad (2)$$

$$M'_{s'} = \left[q\left(q^{-1} \bmod p\right)M_{i'}^{(p)} + p\left(p^{-1} \bmod q\right)M_{j'}^{(q)}\right] \bmod n, \tag{3}$$

where $i', i \in \{1, 2, \ldots, t\}$ and $j', j \in \{1, 2, \ldots, u\}$. Due to random selection of the value M_s at the first step of the algorithm we have the following probabilities: $\Pr(i = i') = t^{-1}$ and $\Pr(j = j') = u^{-1}$. In the case $j = j'$ we have

$$M'_{s'} - M_s = p\left(p^{-1} \bmod q\right)\left(M_{j'}^{(q)} - M_{j}^{(q)}\right) \bmod n \Rightarrow$$

$$\Rightarrow p \mid M'_{s'} - M_s \Rightarrow \gcd\left(M'_{s'} - M_s, n\right) = p.$$

In the case $i = i'$ we have

$$M'_{s'} - M_s = q\left(q^{-1} \bmod p\right)\left(M_{j'}^{(p)} - M_{j}^{(p)}\right) \bmod n \Rightarrow$$

$$\Rightarrow q \mid M'_{s'} - M_s \Rightarrow \gcd\left(M'_{s'} - M_s, n\right) = q,$$

i. e. the algorithm described above works correctly. If the difficulty of the hypothetical attack W_{attac} is sufficiently lager then the difficulty of calculating values $\gcd(M'_{s'} - M_s, n)$ and $\gcd(M'_{s'} - M_s, n)$, then the average complexity of the algorithm is proportional to W_{attac}, i. e. $W_{\text{alg}} \approx const \cdot W_{\text{attac}}$, where $const = z \cdot (t + u)^{-1}$.

Thus, we have proved that an efficient attack breaking a cryptosystem from the considered class can be transformed into efficient algorithm factorizing the modulus n. Theorem 1 is proved.

Plaintext encryption, ciphertext decryption, signature generation, and signature verification procedures have complexity depending on the value z. In general case, for larger values z we have the larger complexity of the procedures. Therefore the most attractive cryptosystems of the considered class correspond to cases relating to small values z.

4 Cryptosystem with Minimum Value *z*

While performing decryption in the considered class of public key cryptosystems we compute z different variants of the plaintexts from which one is to be selected as the correct plaintext. Therefore it is attractively to use the system with minimum value z. Such system corresponds to the case $t = 3$ (i. e. $3 \mid p - 1$) and $u = 1$ (or equivalently $t = 1$ and $u = 3$). In the case $t = k = 3$ and $u = 1$ (or equivalently $t = 1$ and $u = k = 3$) we have the minimum complexity of the encryption, decryption, signature generation, and verification procedures. In such cases to generate a public key one should generate such p that $3|p - 1$ and such q that number 3 does not divide $q - 1$. In this system the public encryption and signature verification are performed accordingly to the following formulas, respectively:

$$C = M^3 \bmod n \quad \text{and} \quad S^3 \bmod n = H' \| R, \quad \text{where} \quad H' = F_H(M).$$

To decipher the cryptogram or to generate a signature to the document M requires calculation of the roots of the third degree modulo n. This procedure has the minimum complexity in the case $p \equiv 7 \bmod 9$. We have three different roots $\{M^{(p)}{}_1, M^{(p)}{}_2, M^{(p)}{}_3\}$ modulo p and only one root $M^{(q)}{}_1$ modulo q. The value $M^{(q)}{}_1$ is calculated as follows: $M_1^{(q)} = C^{3^{-1} \bmod (q-1)} \bmod q$. The values $M^{(p)}{}_1, M^{(p)}{}_2, M^{(p)}{}_3$, and $M^{(q)}{}_1$ define three different roots modulo n which can be calculated using formula (1). In the case of M. Rabin's cryptosystem we have $t = k = 2$, $u = k = 2$, and $z = 4$. Complexity of the signature generation and verification in the considered system is about the same as in the M. Rabin's cryptosystems. In the case $z = 3$ we have minimum complexity of the decryption procedures.

5 Conclusion

This paper introduces a class of provably secure public key cryptosystem which can be used for secure information authentication, public key distribution, and public encryption. This class generalizes the M. Rabin's cryptosystem based on the RSA modulus. Analogously to RSA, new cryptosystem gets its security from difficulty of factorizing modulus and from difficulty of finding kth root ($k \geq 2$) modulo a composite number n. The main difference between the described class of provably secure cryptosystems and the RSA system consists in that in RSA we have $\gcd(e, \varphi(n)) = 1$, but in the mentioned class we have $\gcd(k, \varphi(n)) \geq 2$. Actually, the fact that $k \mid \varphi(n)$, where $k \geq 2$, leads to existence of different values of the kth root modulo n and the last fact has been used in the formal security proof.

The described class of provably secure public key cryptosystems includes M. Rabin's cryptosystem as a particular case corresponding to the case $t = u = k = 2$ and $z = 4$. The minimum value z is equal to 3. This case provides minimum complexity of the information authentication and encryption procedures.

References

1. Rivest, R.L., Shamir, A., Adleman, L.M.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
2. Menezes, A.J., Vanstone, S.A.: Handbook of Applied Cryptography, p. 780. CRC Press (1996)
3. Pieprzyk, J., HardjonoTh., S.J.: Fundumentals of Computer Security, p. 677. Springer, Berlin (2003)
4. Rabin, M.O.: Digitalized signatures and public key functions as intractable as factorization. – Technical report MIT/LCS/TR-212, MIT Laboratory for Computer Science (1979)
5. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
6. Gordon, J.: Strong primes are easy to find. In: Beth, T., Cot, N., Ingemarsson, I. (eds.) Advances in Cryptology. LNCS, vol. 209, pp. 216–223. Springer, Heidelberg (1985)

A New Modeling Paradigm for Dynamic Authorization in Multi-domain Systems[⋆,⋆⋆]

Manoj Sastry[1], Ram Krishnan[2], and Ravi Sandhu[3]

[1] manoj.r.sastry@intel.com
[2] rkrishna@gmu.edu
[3] sandhu@gmu.edu

Abstract. The emergence of powerful, full-featured *and* small form-factor mobile devices enables rich services to be offered to it's users. As the mobile user interacts with multiple administrative domains, he acquires various attributes. In such dynamic usage scenarios, attributes from one domain are interpreted and used in another domain. This motivates the need for dynamic authorization at the time of interaction. In this paper, we investigate the requirements of multi-domain interactions and explore a new paradigm for modeling these requirements using the UCON model for Usage Control [5]. We propose extensions to UCON in order to accommodate dynamic authorizations requirements.

Keywords: Authorization, Multi-domain, UCON, Attribute-based Access Control.

1 Introduction

The advent of small form-factor, high performance computing devices and high bandwidth ubiquitous networks is enabling users to be connected anytime, anywhere with access to rich services. As users become increasingly mobile, they transcend multiple security domains[1]. Context acquired in one domain can be interpreted and used at other domains for access decisions. Our objective in this paper is to investigate dynamic requirements for multi-domain interactions and explore a new paradigm for modeling these properties. Traditional attribute-based access control models have two major limitations: a. In a single domain setting, attributes are typically pre-defined. b. In a multi-domain setting, such models require extensive a-priori agreement of attribute semantics across these systems. We use the term Dynamic Authorization in this paper to collectively refer to the components required for supporting just-in-time authorization.

2 Characteristics of Multi-domain Interactions

In this section, we identify some of the desirable characteristics for user interactions with multi-domain systems using a concrete example. Alice walks into a

[⋆] Copyright ©2007 Intel Corporation.
[⋆⋆] A detailed version is available at http://www.list.gmu.edu/confrnc/misconf/DA.pdf
[1] For simplicity, we will often abbreviate 'security domain' simply as 'domain'.

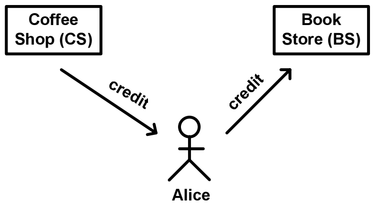


Fig. 1. Coffee Shop Example

Coffee Shop (CS) and engages in a transaction worth \$100. As an appreciation, the CS provides a 'credit' worth \$10. This 'credit' could be used at various other stores like the Bookstore (BS). Alice later uses this 'credit' towards purchasing a book at the BS. Fig. 1 illustrates this scenario. In this example, 'credit' is the context acquired by Alice from the CS and this affects access decision at the BS. We now identify three key characteristics of multi-domain interactions:

1. Multi-domain interactions: Subjects and Objects interact with multiple systems and this is a key characteristic in mobile commerce. E.g. Alice interacts with the CS and the BS which are administratively different domains.
2. Information could be dynamic and transcend systems: Due to mobility, information or context may move from one system to another and could affect access decisions at other systems. E.g. Alice obtained a 'credit' from the CS system and used it to purchase a book from the BS system.
3. No prior configuration: In order to interpret information across multiple domains, systems may have to exchange semantics of this information. But in mobile scenarios, information may be dynamically created and hence a-priori agreement of semantics is not desirable. It must be interpreted at authorization time. E.g. The CS issued 'credit' to Alice. The following day, CS may issue 'coupon' which may be semantically different from 'credit'.
   Further, the following characteristics are also desirable:

3 New Modeling Paradigm for Dynamic Authorization

Our new paradigm is to propose modeling requirements for the three key characteristics discussed earlier: 1. Multi-domain interactions, 2. Information could be dynamic and transcend systems, 3. No prior configuration. We believe that these three characteristics are missing from current approaches to dynamic authorization. Characteristics 1 and 2 brings in a notion of "Multi-Domain Attributes" which are attributes that need to be interpreted across multiple domains. Characteristic 3 brings in a notion of "Dynamic Attributes" which are created dynamically and are not pre-defined. In the coffee shop scenario, the 'credit' attribute was dynamically created by the coffee shop just for that day when Alice interacted with the system. Hence the bookstore cannot write authorization policies to use 'credit' ahead of time. The bookstore needs to interpret the semantics of 'credit' just when Alice uses it to buy a book. Here 'credit' is also an attribute that can be used at multiple domains (CS and BS). Thus it is a Dynamic, Multi-domain attribute. Note that Dynamic Attributes are new-born attributes (name-value) as opposed to the notion of *attribute value changing dynamically.*

4 The Extended UCON$_{ABC}$ Model

We now examine the major components of the UCON model: attributes, obligations, conditions and authorizations. [5] discusses the UCON model in great detail. Fig. 2 shows an extended UCON model or EUCON that accommodates multi-domain interactions. In the following subsections, we explore each of the EUCON components in detail to support dynamic authorization.

4.1 EUCON Attributes

In UCON, attributes are properties of subjects and objects which are used for usage decisions. We now investigate and classify EUCON attributes.

We can classify attributes based on time at which an attribute is defined:

Pre-defined Attributes: These are similar to the conventional notion of attributes. The semantics of these attributes are pre-defined by the administrator.

Dynamic Attributes: These are attributes that are defined just-in-time. E.g. The CS system might define new incentives like 'credit' at different times on different days dynamically. For instance, the CS could create a 'coupon' attribute on the following day which has a different meaning than a dollar value like 'credit'.

We can also classify attributes based on scope as follows:

Local Attributes: These are attributes whose semantics can be interpreted only within the domain where it was defined. It has no meaning or visibility anywhere outside the system in which it is defined. E.g. The CS system may have a Local Attribute called 'id' which may have no meaning outside the CS system.

Multi-domain Attributes: Multi-domain Attributes are attributes whose semantics can be interpreted across multiple domains. E.g. The book store was able to interpret the semantics of 'credit' that was issued by CS.

This classification gives us four possible combinations as follows:

Pre-defined Local Attributes (PLA): PLA's are exactly the same as how current attribute-based models (including UCON) define attributes. Traditionally, PLA's have served the purpose of access control in a single system (or domain).

Pre-defined Multi-domain Attributes (PMA): Current approaches to access control in distributed systems have the notion of PMA's. This involves prior agreement of attribute semantics across all the domains *a-priori*. As discussed earlier, this is clearly not flexible and is not suitable for dynamic scenarios.

Dynamic Local Attributes (DLA): DLA's allow systems to dynamically create attributes interpretable within the same system. Typically such an action is deemed as an administrative task. However, we believe emerging next-generation applications (like context-aware applications) would demand DLA's. In the coffee shop scenario, on a different day CS may create a new 'discount' attribute that could be used by Alice in the coffee shop itself in the future. This 'discount' may not exist all the time. Note that DLA's may or may not be persistent.

Dynamic Multi-domain Attributes (DMA): DMA is fundamentally a new approach to modeling emerging usage scenarios. As discussed earlier, systems may define attributes dynamically that needs to be interpreted at multiple domains. This requires authorization policies to be created dynamically. In the coffee shop scenario, a new attribute called 'credit' was dynamically created at some time and Alice received it. Further, Alice was able to use this attribute at the bookstore. The bookstore dynamically interpreted the semantics of 'credit' by interacting with the coffee shop and authorized purchasing a book with 'credit'. DMA could apply to both subjects and objects. In the coffee shop scenario, it is

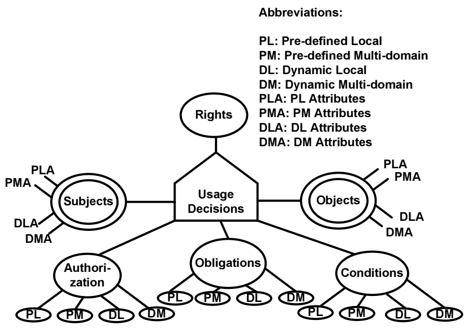


Fig. 2. Extended UCON Components

clear that 'credit' is the DMA of a subject (Alice). Here is another scenario to appreciate the generality of DMA's for subjects:

Airport Security: In airport scenario, a passenger interacts with multiple systems. Further each system (security, shops, airlines, etc.) may define their own attributes dynamically. For example, suppose that the security check-in system in an airport and the airline systems are multi-domain systems with no a-priori configuration. When Alice checks-in through the security system, she obtains a DMA called "cleared=true". This DMA could then be used by Alice at the airline's boarding system to board the airplane.

Following is an example of DMA's for objects:

Airport Security: Following the airport security example discussed for subjects, when Alice checks in at airport security, all the objects that she carries (e.g. luggage, laptop, etc.) could obtain a DMA "cleared=true". Alice can use this DMA at the airline system in order to board the flight with her objects.

4.2 EUCON Authorizations

In UCON, the authorization component contains rules based on subject and object attributes. We discussed that attributes could be classified into four different categories. Because authorization involves constructing rules based on subject and object attributes, we have a similar notion for EUCON authorizations:

Pre-defined Local Authorization: Current UCON's authorization would fall under this category. These rules have served the needs of traditional systems.

Pre-defined Multi-domain Authorization: This involves constructing rules based on Pre-defined Multi-domain Attributes. Current approaches to authorization in multi-domain systems take this approach. Attributes are pre-defined and authorization rules are constructed at multiple domains based on these attributes.

Dynamic Local Authorization: This involves constructing rules based on Dynamic Local Attributes. In the coffee shop scenario, a dynamic local authorization rule could be constructed so that subjects (e.g. Alice) who obtained 'credit' cannot obtain another incentive say 'coupon' at the same time.

Dynamic Multi-domain Authorization: This involves constructing dynamic authorization rules by interpreting Dynamic Multi-domain Attributes. E.g. The bookstore needs to interpret 'credit' dynamically and construct dynamic multi-domain authorization rules. Exactly how such policies are constructed is an enforcement level issue and restrictions should not be made in the policy model.

4.3 EUCON Obligations and Conditions

In UCON, obligations are actions a subject needs to perform before an access can be granted. For example, a subject may be obligated to 'agree' to a license before an object can be accessed. In UCON, conditions are system level factors that need to hold for access to be granted. For example, a server's load should be below a threshold value in order to accept new client connections.

Similar to attributes, in EUCON we can classify both obligations and conditions as: Pre-defined Local, Pre-defined Multi-domain, Dynamic Local, Dynamic Multi-domain. Pre-defined local and dynamic local obligations and conditions are similar to their attribute counter-parts. We discuss the other two below:

Pre-defined Multi-domain Obligations (and conditions): These are pre-defined obligations (and conditions) interpretable across multiple systems. Note that these are obligations (and conditions) on using Multi-domain Attributes.

Dynamic Multi-domain Obligations: These are obligations defined dynamically and are interpreted at multiple systems at authorization time. Again note that these are obligations on using Multi-domain Attributes at different systems. E.g. Suppose that there are two coffee shops: the coffee shop that issued 'credit' – CS and a coffee shop located within the book store – CS@BS. When Alice uses her 'credit' at BS, there could be an obligation that Alice needs to engage in a transaction with the CS@BS before 'credit' could be used at the BS.

Dynamic Multi-domain Conditions: These are conditions on using Multi-domain Attributes at different systems. Following Dynamic Multi-domain Obligations, say that Alice fulfills her obligation. The BS could then dynamically discover a condition on using 'credit' that current 'credit' usage on all coffee shop systems has not exceeded $1000 and the 'credit' expires on 05-25-2007.

5 Related Work

Many related work exists in the arena of dynamic authorization ([2], [1], [4], [6], [3]). We only discuss two of them here. In [1], a Contextual Attribute-Based Access Control model is proposed. The authors define Transaction Attributes (TA) as attributes that a subject obtains as part of a transaction. These TA's would fall under our Pre-defined Multi-domain category. In [2], the authors identify requirements for access control in open environments similar to ours. They survey extensions that have been proposed in general in different access control models. However our modeling paradigm of creating and interpreting attributes dynamically across multiple systems is substantially different.

6 Conclusion and Future Work

In this paper, we explored a new paradigm for modeling dynamic authorizations in multi-domain systems. Current access control models including UCON predefine their components and we demonstrated with compelling usage scenarios that such static definitions would not serve the needs of mobile and dynamic multi-domain interactions. We proposed extensions to the UCON model to express dynamic authorization policies. A formal EUCON model for multi-domain interactions needs to be specified. Enforcement and Implementation models supporting dynamic authorization need to be studied.

References

1. Covington, M., Sastry, M.: A Contextual Attribute-based Access Control Model. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. LNCS, vol. 4278, Springer, Heidelberg (2006)
2. Damiani, E., Vimercati, S., Samarati, P.: New Paradigms for Access Control in Open Environments. In: 5th IEEE Intl. Symposium on Signal Processing and Information, IEEE Computer Society Press, Los Alamitos (2005)
3. Freudenthal, E., Pesin, T., Port, L., Keenan, E., Karamcheti, V.: drbac: Distributed Role-based Access Control for Dynamic Coalition Environments. In: Proceedings of 22nd ICDCS, pp. 411–420 (2002)
4. Lepro, R.: Cardea: Dynamic Access Control in Distributed Systems. SYSTEM 3, 4 (2003)
5. Park, J., Sandhu, R.: The UCON$_{ABC}$ Usage Control Model. TISSEC 7, 57–64 (2004)
6. Hayton, R.J., Bacon, J.M., Moody, K.: Access Control in an Open Distributed Environment. In: IEEE Symposium on Security and Privacy, pp. 3–14. IEEE Computer Society Press, Los Alamitos (1998)

Synthesis of Non-interferent Distributed Systems

Franck Cassez[1], John Mullins[2], and Olivier H. Roux[1]

[1] IRCCyN/CNRS
BP 92101 1 rue de la Noë 44321 Nantes Cedex 3 France
[2] École Polytechnique de Montréal
P.O. Box 6079, Station Centre-ville, Montreal (Quebec), Canada, H3C 3A7

Abstract. In this paper, we focus on distributed systems subject to security issues. Such systems are usually composed of two entities: a high level user and a low level user that can both do some actions. The security properties we consider are non-interference properties. A system is non-interferent if the low level user cannot deduce any information by playing its low level actions. Various notions of non-interference have been defined in the literature, and in this paper we focus on two of them: one trace-based property (SNNI) and another bisimulation-based property (BSNNI).

For these properties we study the problems of synthesis of a high level user so that the system is non-interferent. We prove that a most permissive high level user can be computed when one exists.

Keywords: Non-Interference, Controller Synthesis.

1 Introduction

Security in Distributed Systems. Nowadays computing environments allow users to employ programs that are sent or fetched from different sites to achieve their goal, either in private or in an organization. Such programs may be run as a code to do simple calculation task or as interactive communicating programs doing IO operations or communications. Sometimes they deal with secret information such as private data of a user or as classified data of an organization. Similar situations may occur in any computing environments where multiple users share common computing resources. One of the basic concerns in such context is to ensure programs not to leak sensitive data to a third party, either maliciously or inadvertently. This is one of the key aspects of the security concerns, that is often called *secrecy*.

Non-Interference. The *information flow analysis* addresses this concern by clarifying conditions when a flow of information in a program is safe (*i.e.* high level information never flows into low level channels). These conditions referred to as *non-interference* properties, capture any causal dependency between high level and low level behaviours. Their characterization has appeared rapidly out of the scope of the common safety/liveness classification of system properties considered by the system verification community during the last twenty five years.

Also, in recent years, verification of information flow security has become an emergent field of research in computer science. It can be applied to the analysis of cryptographic protocols where numerous uniform and concise characterizations of information flow security properties (*e.g.* confidentiality, authentication, non-repudiation or anonymity) in terms of non-interference have been proposed.

Control vs. Verification. The *verification* problem for a given system S and a specification ϕ consists in checking whether S satisfies ϕ which is often written $S \models \phi$ and referred to as the *model-checking problem*. The *control problem* assumes the system is *open i.e.* we can restrict the behaviour of S: some events in S are *controllable* and the others are *uncontrollable*, and we can sometimes disable controllable actions. A *controller* C for S is a mapping which gives, for any history[1] ρ of the system S, the controllable action that can be played (a controller cannot restrict uncontrollable actions). The *supervised* system $S \times C$ (read "S supervised by C") is composed of the subset of the behaviours of S that can be generated using the action prescribed by C. The *control problem* for a system S and a specification ϕ asks the following: Is there a controller C s.t. $S \times C \models \phi$? The associated *control synthesis problem* asks to compute a witness controller C.

Controlling Non-Interference. In this paper we introduce *non-interference control problems*. In this setting we assume that high level actions are controllable and low level actions are uncontrollable. Given a system S, a type of non-interference $\phi \in \{\mathrm{SNNI}, \mathrm{BSNNI}\}$, the ϕ-non-interference control problem asks the following: Is there a controller C such that $S \times C$ has the ϕ-non-interference property? The associated synthesis problems ask to compute a witness controller.

Related Work & Our Contribution. [1] considers the complexity of many non-interference verification problems but control is not considered in this paper. [2] presents a tranformation that removes timing leaks from programs to make them non-interferent w.r.t. a bisimulation condition. The transformation is decidable but this problem is still different from that of control. Indeed, the transformation removes timing leaks from programs without untimed leaks, by padding with dummy instructions where needed. There is also a large body of work on the use of static analysis techniques to enforce information flow policies. A general overview may be found in [3]. Much of this work is behaviour based; systems are deemed to be interference-free if their trace sets, sequences of actions labelled "high" or "low", satisfy certain properties. Here we use a more extensional approach, saying that a system is interference-free if low level observers are unable to discern the presence or absence of high level components. However a formal comparison between notions of non-interference on programming languages and similar notions on event-based systems is not straightforward and at the best of our knowledge, has never been investigated. The non-interference control problem was first considered in [4] for dense timed systems given by timed automata. The timed non interference properties are expressed in terms

[1] If the system is a finite automaton, the history is the complete run with states and labels of the transitions.

of states equivalence and co-simulation relations. These control problems are proved decidable and the associated synthesis problems are computable but lead to a non-interferent controlled system which is not necessarily the most permissive.

In this paper, we precisely define the non-interference control problems for two types of non-interference properties: a trace-based property, SNNI and a bisimulation-based property, BSNNI. We show that both problems are decidable. Moreover, given a system S, we prove that a *most permissive* non-interferent (sub)system of S can be computed. To our knowledge, non-interference control problems for finite state systems have not been considered so far.

Organization of the paper. Section 2 recalls the basics of finite automata, languages, bisimulation, and μ-calculus. Section 3 is devoted to the definition of non-interference and known results about control problems for finite automata. Section 4 is the core of the paper and contains the main results: control of SNNI and BSNNI. Section 5 gives the directions of future work.

2 Preliminaries

Let Σ be a finite set, $\varepsilon \notin \Sigma$ and $\Sigma^\varepsilon = \Sigma \cup \{\varepsilon\}$. A *word* w over Σ is a sequence of letters $w = a_0 a_1 \cdots a_n$ s.t. $a_i \in \Sigma$ for $0 \leq i \leq n$. Σ^* is the set of words over Σ. We denote $u.v$ the *concatenation* of two words. As usual ε is also the empty word s.t. $u.\varepsilon = \varepsilon.u = u$. A *language* is a subset of Σ^*. Given two languages A and B over Σ, $A.B$ is the set of words defined $A.B = \{w \in \Sigma^* \,|\, w = u.v \textit{ with } u \in A, v \in B\}$. The set of mappings from A to B is denoted $[A \rightarrow B]$. Given a word $w = a_0 a_1 \cdots a_n$ and $L \subseteq \Sigma$ the *projection* of w over L is denoted w/L.

2.1 Labeled Transition Systems

Definition 1 (Labeled Transition System). *A* labeled transition system (LTS) *is a tuple $A = (S, s_0, \Sigma^\varepsilon, \rightarrow)$ where S is a set of states, s_0 is the initial state, Σ a finite alphabet of actions and $\rightarrow \subseteq S \times \Sigma^\varepsilon \times S$ is the transition relation. We use the notation $q \xrightarrow{a} q'$ if $(q, a, q') \in \rightarrow$. A LTS is finite is S is finite. We let $En(q)$ be the set of labels a s.t. $(q, a, q') \in \rightarrow$ for some q'.* □

A run ρ of A from s is a finite sequence of transitions $\rho = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$ s.t. $q_0 = s$ and $(q_i, a_i, q_{i+1}) \in \rightarrow$ for $0 \leq i \leq n-1$. We let $Runs(s, A)$ be the set of runs from s in A and $Runs(A) = Runs(s_0, A)$. The *trace* of ρ is $trace(\rho) = a_0 a_1 \cdots a_n$. A word $w \in \Sigma^*$ is *generated* by A if $w = trace(\rho)$ for some $\rho \in Runs(A)$. The language generated[2] by A, $\mathcal{L}(A)$, is the set of words generated by A. Two transition systems A and B are *language equivalent* denoted $A \approx_{\mathcal{L}} B$ if $\mathcal{L}(A) = \mathcal{L}(B)$ *i.e.* they generate the same set of words.

Definition 2 (Product of LTS). *Let $A_1 = (S_1, s_0^1, \Sigma_1, \rightarrow)$ and $A_2 = (S_2, s_0^2, \Sigma_2, \rightarrow)$ be two LTS. The* synchronized product *of A_1 and A_2 is the LTS $A_1 \times$*

[2] Notice that $\mathcal{L}(A)$ is *prefix closed* as we use LTS that have no accepting or final states.

$A_2 = (S, s_0, \Sigma, \rightarrow)$ defined by: $S = S_1 \times S_2$, $s_0 = (s_0^1, s_0^2)$, $\Sigma = \Sigma_1 \cup \Sigma_2$ and $(s_1, s_2) \xrightarrow{\lambda} (s_1', s_2')$ if for some $i \in \{1, 2\}$ (i) either $s_i \xrightarrow{\lambda} s_i'$ with $\lambda \in \Sigma \setminus \Sigma_{2-i+1}$ and $s_{2-i+1}' = s_{2-i+1}$ or (ii) $s_i \xrightarrow{\lambda} s_i'$ if $\lambda \in \Sigma_1 \cap \Sigma_2$. $\square$

2.2 Bisimulation, Restriction and Abstraction

Definition 3 (Bisimulation). *Let $A = (S_A, s_0^A, \Sigma^\varepsilon, \rightarrow)$, $B = (S_B, s_0^B, \Sigma^\varepsilon, \rightarrow)$ be two LTS. $\mathcal{R} \subseteq S_A \times S_B$ is a* bisimulation *if*

1. *for each $s \in S_A$ if $s \xrightarrow{\lambda} s'$, then there is a state $t \in S_B$ s.t. $s \,\mathcal{R}\, t$ and $t \xrightarrow{\lambda} t'$ and $s' \,\mathcal{R}\, t'$;*
2. *for each $t \in S_B$ if $t \xrightarrow{\lambda} t'$, then there is a state $s \in S_A$ s.t. $s \,\mathcal{R}\, t$ and $s \xrightarrow{\lambda} s'$ and $s' \,\mathcal{R}\, t'$.*

Two LTS A and B are strongly bisimilar *if there is a bisimulation $\mathcal{R}$ for A and B s.t. $(s_0^A, s_0^B) \in \mathcal{R}$. We write $A \approx_{\mathcal{S}} B$ if A and B are strongly bisimilar.* $\square$

Definition 4 (ε-Abstract LTS). *Let $A = (S, s_0, \Sigma^\varepsilon, \rightarrow)$ be a LTS. The ε-*abstract *LTS associated with A is $A^\varepsilon = (S, s_0, \Sigma, \rightarrow_\varepsilon)$ where $s \xrightarrow{u}_\varepsilon s'$ iff there is a run $s \xrightarrow{\varepsilon^*.u.\varepsilon^*} s'$ in A.* $\square$

Two LTS A and B are *weakly bisimilar*, denoted $A \approx_{\mathcal{W}} B$, if $A^\varepsilon \approx_{\mathcal{S}} B^\varepsilon$.

The *abstracted* transition system hides a set of labels $L \subseteq \Sigma$:

Definition 5 (Abstracted Transition System). *Given a LTS $A = (S, s_0, \Sigma^\varepsilon, \rightarrow)$ and $L \subseteq \Sigma$ we define the LTS $A/L = (S, s_0, (\Sigma \backslash L)^\varepsilon, \rightarrow_L)$ where $q \xrightarrow{a}_L q' \iff q \xrightarrow{a} q'$ for $a \in \Sigma \setminus L$ and $q \xrightarrow{\varepsilon}_L q' \iff q \xrightarrow{a} q'$ for $a \in L \cup \{\varepsilon\}$.* $\square$

The *restricted* transition system cuts transitions labeled by the letters in $L \subseteq \Sigma$:

Definition 6 (Restricted Transition System). *Given a LTS $A = (S, s_0, \Sigma^\varepsilon, \rightarrow)$ and $L \subseteq \Sigma$, $A \setminus L$ is the LTS $(S, s_0, (\Sigma \setminus L)^\varepsilon, \rightarrow_L)$ where $q \xrightarrow{a}_L q' \iff q \xrightarrow{a} q'$ for $a \in \Sigma^\varepsilon \setminus L$.* $\square$

2.3 The Modal μ-Calculus and Characteristic Formulæ

The modal μ-calculus was introduced by Kozen [5]. It is used to specify properties of LTS and often called the assembly language w.r.t. temporal logics formalisms. A recent survey on the subject can be found in [6]. Let $\mathcal{V}$ be a countable set of *formula variables* and $A = (S, s_0, \Sigma, \rightarrow)$ be a LTS. We consider the modal μ-calculus with formulas in positive normal form. It is defined by the following grammar:

$$\phi ::= \mathsf{tt} \mid \mathsf{ff} \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid [\sigma]\phi \mid \langle\sigma\rangle\phi \mid Z \mid \mu Z.\phi \mid \nu Z.\phi. \qquad (1)$$

where $\sigma \in \Sigma$, $Z \in \mathcal{V}$ and $\mathbb{B} = \{\mathsf{tt}, \mathsf{ff}\}$ is the set of boolean values. If ϕ is generated by the previous grammar we say that ϕ is a μ-formula or formula for short. A *closed formula* (formula for short in the sequel) of the μ-calculus is a formula s.t. every variable is under the scope of an operator ν or μ. As usual we agree on $\wedge_{\phi\in\emptyset}\phi = \mathsf{tt}$ and $\vee_{\phi\in\emptyset}\phi = \mathsf{ff}$. We denote $[\![\varphi]\!]^A_\rho \subseteq S$ the interpretation of φ w.r.t. a LTS A and a *context* $\rho \in [\mathcal{V} \longrightarrow 2^S]$. $[\![\varphi]\!]^A_\rho$ is defined inductively by the equations of Fig. 1. For the semantics definition we use the notations:

$$\begin{array}{lll@{\qquad}lll}
[\![Z]\!]^A_\rho & = & \rho(Z) & [\![\phi \wedge \psi]\!]^A_\rho & = & [\![\phi]\!]_\rho \cap [\![\psi]\!]^A_\rho \\
[\![\neg\phi]\!]^A_\rho & = & S \setminus [\![\phi]\!]^A_\rho & [\![\phi \vee \psi]\!]^A_\rho & = & [\![\phi]\!]_\rho \cup [\![\psi]\!]^A_\rho \\
[\![\langle\sigma\rangle\phi]\!]^A_\rho & = & [\![\langle\sigma\rangle]\!]^A([\![\phi]\!]^A_\rho) & [\![\nu Z.\phi]\!]^A_\rho & = & \bigcup\{\xi \subseteq S : \xi \subseteq [\![\phi]\!]^A_{\rho[Z\mapsto\xi]}\} \\
[\![[\sigma]\phi]\!]^A_\rho & = & [\![[\sigma]]\!]^A([\![\phi]\!]^A_\rho) & [\![\mu Z.\phi]\!]^A_\rho & = & \bigcap\{\xi \subseteq S : [\![\phi]\!]^A_{\rho[Z\mapsto\xi]} \subseteq \xi\}
\end{array}$$

Fig. 1. Semantics of the modal μ-calculus

- let $Z \in \mathcal{V}$ and ρ be a context, $\xi \in 2^S$, $\rho[Z \mapsto \xi]$ is the context defined by $\rho[Z \mapsto \xi](X) = \rho(X)$ if $X \neq Z$ and $\rho[Z \mapsto \xi](Z) = \xi$;
- given $\sigma \in \Sigma$, $[\![[\sigma]]\!]^A$ and $[\![\langle\sigma\rangle]\!]^A \in [2^S \longrightarrow 2^S]$ are the predicate transformers defined as follows:

$$[\![[\sigma]]\!]^A(X) = \{s \in S : \forall s' \text{ s.t. } s \xrightarrow{\sigma} s' \text{ then } s' \in X\}$$
$$[\![\langle\sigma\rangle]\!]^A(X) = \{s \in S : \exists s' \text{ s.t. } s \xrightarrow{\sigma} s' \text{ and } s' \in X\}.$$

For a closed formula ϕ, one has that $[\![\phi]\!]^A_\rho = [\![\phi]\!]^A_{\rho'}$, for any contexts ρ, ρ'. In this case, we simply use $[\![\phi]\!]^A$. We define the *satisfaction relation* $\models$ by: $A, s \models \phi$ if and only if $s \in [\![\phi]\!]^A$. If $A, s_0 \models \phi$, we say that A is a model of ϕ, in short $A \models \phi$.

Definition 7. *Let $A = (S, s_0, \Sigma, \rightarrow)$ be a LTS. A formula ϕ s.t. $A \models \phi$ is a* characteristic formula *of A up to strong bisimulation (or simply* characteristic formula *of A) if for any LTS B, $A \approx_{\mathcal{S}} B$ iff $B \models \phi$.* □

Let $A = (S, s_0, \Sigma, \rightarrow)$. A characteristic formula $CF(A)$ of A is given by the system of equations [7,8] for each $q \in S$:

$$X_q = \bigwedge_{a,q',\, q \xrightarrow{a} q'} \langle a\rangle X_{q'} \wedge \bigwedge_{a\in\Sigma} [a]\Big(\bigvee_{q'\in Q, q\xrightarrow{a} q'} X_{q'}\Big)$$

By definition of $CF(A)$ we have:

Lemma 1. *$A \approx_{\mathcal{S}} B \iff B \models CF(A)$.*

3 Control and Non-interference

3.1 Control Problems

Let $A = (S, s_0, \Sigma, \rightarrow)$ be a LTS s.t. the set of actions is partitioned into Σ_u (*uncontrollable actions*) and Σ_c (*controllable actions*). In this case we say A is a *Game* LTS (GLTS).

Definition 8 (Controller). *A* controller *for A is a (possibly infinite) LTS $C = (Q, q_0, \Sigma, \rightarrow)$ which is* complete *w.r.t. Σ_u: $\forall q \in Q, \forall u \in \Sigma_u$, there is some $q' \in Q$ s.t. $(q, u, q') \in\rightarrow$.* $\square$

Supervisory Control Problem [9,10]. Given a language L, the *prefix closure* of L, denoted $\overline{L}$ is the set of prefixes of words in L. L is a *closed language* if $\overline{L} = L$. Let A be a GLTS and $\emptyset \subset K \subseteq \mathcal{L}(A)$ be a closed language. K is *controllable* w.r.t. $(\mathcal{L}(A), \Sigma_u)$ if $\overline{K}.\Sigma_u \cap \mathcal{L}(A) \subseteq \overline{K}$.

The *supervisory control problem (SCP)* asks the following: given A and K, is there a controller C s.t. $\mathcal{L}(C \times A) = K$?

One result in [9,10] is that such a controller exists iff K is controllable. In case K is not controllable, one can compute the largest language included in K that is controllable: it is called the *supremal controllable sub-language* of K. This problem is referred to as the *supremal controllable sub-language problem*:

Compute the supremal controllable sub-language of K. (SCSLP)

Let $\sup(A, K)$ be the supremal controllable sub-language. By definition any controllable language for A is a subset of $\sup(A, K)$.

Assume K is a language given by a *deterministic* finite automaton A_K. Computing $\sup(A, K)$ can be done in polynomial time in the size of A and A_K. In case $\sup(A, K) \neq \emptyset$ we can even obtain the *most liberal controller* C s.t. $\mathcal{L}(C \times A) = \sup(A, K)$ in polynomial time. C can be represented by a finite state automaton: one can prove (*e.g* [9,10,11]) that C is a memoryless[3] controller for the GLTS $A \times \bar{A}_K$ where $\bar{A}_K$ generates the complement language of K. Hence the most liberal controller C is a finite memory[4] controller that has size at most $|A| \cdot |\bar{A}_K|$.

μ-Calculus Control Problem [12,13]. Given a LTS A, a closed μ-formula φ, the *μ-control problem* (μ-CP) is the following:

Is there a (non trivial) controller C such that $C \times A \models \varphi$? (μ-CP)

The actual control problem we want to solve is to compute a non trivial controller *i.e.* one that does not disable every action. This problem has been proved to be solvable in [12,13]. An algorithm to solve μ-CP and compute the most permissive controller is given [12]: 1) first construct a *modal-loop formula*[5] $\varphi(A)$ which is a *quotient automaton* of φ by A s.t. $P \models \varphi(A)$ iff $P \times A \models \varphi$; 2) transform $\varphi(A)$ into a non *deterministic loop automaton* and synthesize a controller for this automaton. This transformation into an automaton may cause an exponential blow-up and the complexity of the μ-CP depends on many parameters. The exact complexity of μ-CP is not yet known nevertheless a finite memory most permissive controller can be synthesized if A is controllable w.r.t. φ.

[3] C is a memoryless controller for A if C can be described using the states of A: the controllable events that are allowed after a particular sequence of events depend only on the state that is reached in A after reading this sequence of events.

[4] C may need more states than the one of A and in this sense it has finite memory.

[5] Modal-loop formulas are μ-formulas extended with a modality to check loops.

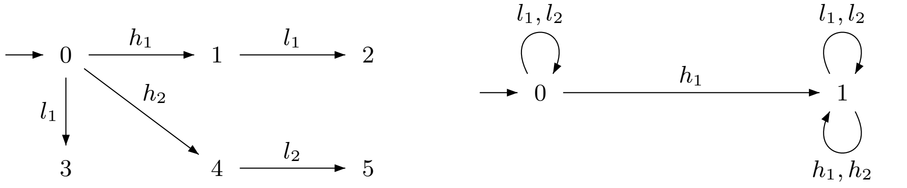
3.2 Non-interference Problems

The *strong non-deterministic non-interference* (SNNI) property has been first proposed by Focardi [14] as a *trace-based* generalization of non-interference for concurrent systems. Let $A = (S, s_0, \Sigma, \rightarrow)$ be a GLTS, s.t $\Sigma = \Sigma_u \cup \Sigma_c$: Σ_u (resp. Σ_c) is the set of *public* (resp. *private*) actions.

Definition 9 (SNNI). *A has the* strong non-deterministic non-interference *(SNNI) property if and only if $A/\Sigma_c \approx_{\mathcal{L}} A\backslash\Sigma_c$.* □

The SNNI *verification problem* (SNNI-VP) is the following: given a GLTS A with $\Sigma = \Sigma_u \cup \Sigma_c$, decide whether A has the SNNI property. The problem of establishing language equivalence of non-deterministic finite automata is reducible in polynomial time to the problem of checking trace equivalence. Such a problem is known to be PSPACE-complete. Also, SNNI-VP is in PSPACE as well [14].

Example 1 (SNNI). Fig. 2(a) gives an example of a system that has not the SNNI property. The high level (controllable) actions are $\Sigma_c = \{h_1, h_2\}$ and the low level (uncontrollable) actions are $\Sigma_u = \{l_1, l_2\}$. l_2 is a trace of $\mathcal{A}/\Sigma_c$ but not of $\mathcal{A}\backslash\Sigma_c$ and $\mathcal{A}$ does not have the SNNI property. If we consider $\mathcal{A}$ without the action h_2, l_2 (no states 4 and 5) then $\mathcal{A}$ satisfies the SNNI property. ■



(a) The automaton $\mathcal{A}$ (b) The Most Liberal Controller for $\mathcal{A}$

Fig. 2. Automaton and Controller

We now give the bisimulation-based definition of strong non-deterministic non-interference proposed in [14]. Actually, any bisimulation-based information flow property presented in [14] could be recast in a similar manner.

Definition 10 (Bisimulation-based SNNI). *A has the* bisimulation-based strong non-deterministic non-interference *(BSNNI) property if and only if $A/\Sigma_c \approx_{\mathcal{W}} A\backslash\Sigma_c$.* □

The BSNNI *verification problem* (BSNNI-VP) is the following: given a GLTS A with $\Sigma = \Sigma_u \cup \Sigma_c$, decide whether A has the BSNNI property. The problem is known to be polynomial time [14] in the size of the ϵ-abstract automaton A^ϵ.

4 Control of Non-interference

The previous non interference problems (SSNI-VP, BSNNI-VP) consist in *checking* whether a GLTS has the non-interference property. In case the answer is "no" one has to investigate why the non-interference property does not hold, modify A and check again the property again. In contrast to the verification problem, the control problem indicates whether there is a way of restricting the behaviour of the high level user to ensure a given property. It is interesting because we can start with a very permissive high level user and then check whether its behaviour can be restricted by a controller to ensure a non-interference property. The SNNI-*Control Problem* (SNNI-CP) is the following: given a GLTS A with $\Sigma_c \cup \Sigma_u$,

$$\textit{Is there a controller } C \textit{ s.t. } (C \times A)/\Sigma_c \approx_{\mathcal{L}} (C \times A)\backslash\Sigma_c\textit{?} \qquad \text{(SNNI-CP)}$$

As stated previously, we are interested in non trivial controllers *i.e.* those that do not disable every action (which is always a solution). In the sequel we show that we can compute the most liberal controller: if the most liberal controller disables every action then there is no non trivial controller and otherwise we obtain all the possible non trivial controllers. The SNNI-*Control Synthesis Problem* (SNNI-CSP) asks to compute a witness when the answer to the SNNI-CP is "yes". The BSNNI-CP is defined in the obvious manner: given a GLTS A with $\Sigma_c \cup \Sigma_u$,

$$\textit{Is there a controller } C \textit{ s.t. } (C \times A)/\Sigma_c \approx_{\mathcal{W}} (C \times A)\backslash\Sigma_c\textit{?} \qquad \text{(BSNNI-CP)}$$

Again the synthesis problem associated with the BSNNI-CP asks to compute a witness when the answer to the BSNNI-CP is "yes". In the sequel we show how to solve Problems SNNI-CP and BSNNI-CP.

4.1 SNNI Control Problem

We reduce the SNNI-CP to the supervisory control problem. Let $\mathcal{U}$ be an automaton that accepts Σ_c^*. We first prove the following lemma:

Lemma 2

$$(C \times A)/\Sigma_c \approx_{\mathcal{L}} (C \times A)\backslash\Sigma_c \iff \mathcal{L}(C \times A) \subseteq \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U}) \cap \mathcal{L}(A).$$

Proof. By definition, $(C\times A)/\Sigma_c \approx_{\mathcal{L}} (C\times A)\backslash\Sigma_c \iff \mathcal{L}((C\times A)/\Sigma_c) = \mathcal{L}((C\times A)\backslash\Sigma_c)$. First notice that $\mathcal{L}((C\times A)\backslash\Sigma_c) = \mathcal{L}(A\backslash\Sigma_c)$. Indeed, any controller C for A cannot prevent uncontrollable actions from occurring. Moreover, a controller can only restrict the set of controllable actions from any state s and thus $(C \times A)\backslash\Sigma_c$ is just the LTS A where all the branches following a Σ_c action have been pruned. Notice also that $\mathcal{L}((C \times A)\backslash\Sigma_c) \subseteq \mathcal{L}((C \times A)/\Sigma_c)$. Using the previous two equations, we obtain:

$$(C \times A)/\Sigma_c \approx_{\mathcal{L}} (C \times A)\backslash\Sigma_c \iff \mathcal{L}((C \times A)/\Sigma_c) \subseteq \mathcal{L}(A\backslash\Sigma_c)$$

Moreover we can also prove that for any controller C for A:

$$(i)\, \mathcal{L}((C \times A)/\Sigma_c) \subseteq \mathcal{L}(A\backslash\Sigma_c) \iff (ii)\, \mathcal{L}(C \times A) \subseteq \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U}) \cap \mathcal{L}(A)$$

Assume (i) holds. Let $w \in \mathcal{L}(C \times A)$. Then $w/\Sigma_c \in \mathcal{L}(C \times A)/\Sigma_c = \mathcal{L}((C \times A)/\Sigma_c)$. By (i), $w/\Sigma_c \in \mathcal{L}(A\backslash\Sigma_c)$. Then w must be equal to w/Σ_c in which some Σ_c actions are inserted, which is exactly the definition of $\mathcal{L}(A\backslash\Sigma_c \times \mathcal{U})$. As $\mathcal{L}(C \times A) \subseteq \mathcal{L}(A)$, this entails $\mathcal{L}(C \times A) \subseteq \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U}) \cap \mathcal{L}(A)$. Assume (*ii*) holds. Let $w \in \mathcal{L}((C \times A)/\Sigma_c)$. By definition there is some $w' \in \mathcal{L}(C \times A)$ s.t. $w = w'/\Sigma_c$. By (*ii*) $w' \in \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U})$. This entails $w \in \mathcal{L}(A\backslash\Sigma_c)$. □

This enables us to reduce the SNNI-CP to SCSLP:

Theorem 1. *The SNNI-CSP is in EXPTIME.*

Proof. The SNNI-CP asks whether there exists a controller C s.t. $(C\times A)/\Sigma_c \approx_{\mathcal{L}} (C\times A)\backslash\Sigma_c$. Using Lemma 2, this problem amounts to finding a controller C s.t. $\mathcal{L}(C \times A) \subseteq \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U}) \cap \mathcal{L}(A)$. Thus we just need to solve an instance of the SCSLP with $K = \mathcal{L}(A\backslash\Sigma_c \times \mathcal{U}) \cap \mathcal{L}(A)$. The right-hand side of this equation is a fixed (closed) language K that can be generated by a deterministic automaton A_K. Notice that as A may be non deterministic, this automaton has size at most exponential in the size of A. As stated in section 3.1, we can compute the most liberal controller s.t. C s.t. $(C\times A)/\Sigma_c \approx_{\mathcal{L}} (C\times A)\backslash\Sigma_c$ and moreover C is a finite memory controller of exponential size in the size of A. Indeed, C is a memoryless controller for a game $A \times \bar{A}_K$ where $\bar{A}_K$ generates the complement language of K. Notice that if A is deterministic the algorithm becomes polynomial because no determinization step is needed. □

A consequence of Theorem 1 is that SNNI-VP can be solved by our algorithm: to solve SNNI-VP, we compute the most liberal controller C and then check that for each state (s, p) of $A \times \bar{A}_K$ we have $En(s, p) = En(s)$ *i.e.* the most liberal controller does not prevent any of the high level actions.

Example 2 (Control of SNNI). We use the automaton of Fig. 2(a) (example 1) to show how to synthesize a controller that ensures SNNI. We recall that $\Sigma_c = \{h_1, h_2\}$ and $\Sigma_u = \{l_1, l_2\}$. First we have $\mathcal{L}(\mathcal{A}) = \{h_1.l_1, l_1, h_2.l_2\}$, $\mathcal{L}(\mathcal{A}\backslash\Sigma_c \times \mathcal{U}) = \Sigma_c^*.l_1.\Sigma_c^*$, and thus $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}\backslash\Sigma_c \times \mathcal{U}) = \{h.l_1, l_1\}$.

We can use standard controller synthesis procedures[6] (see e.g. [11]) to compute the most liberal controller. This is a controller that avoids states $\{4, 5\}$ in $\mathcal{A}$ and thus prevents h_2. The subsystem of $\mathcal{A}$ obtained by removing h_2 is the maximal subsystem that has the SNNI property and the most liberal controller C is given on Fig. 2(b). ■

4.2 BSNNI Control Problem

We can define a satisfaction relation $\models_\varepsilon$ that considers the ε action as the invisible action and extend our interpretation of μ-calculus formulas over LTS containing ε by: $A \models_\epsilon \phi \iff A^\epsilon \models \phi$. If $L \subseteq \Sigma^\varepsilon$, we define the following "macro" operators for the μ-calculus:

$$[L]\varphi \overset{\text{def}}{=} \wedge_{b\in L}[b]\varphi \qquad \langle L\rangle\varphi \overset{\text{def}}{=} \vee_{b\in L}\langle b\rangle\varphi$$

$$\langle L^*\rangle\varphi \overset{\text{def}}{=} \mu X.(\varphi \vee \langle L\rangle X) \qquad [L^*]\varphi \overset{\text{def}}{=} \nu X.(\varphi \wedge [L]X)$$

[6] This generates *state* based memoryless controllers which generates the supremal controllable sub-language of $\mathcal{A}$ w.r.t. $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}\backslash\Sigma_c \times \mathcal{U})$.

The translation $\kappa_L(\varphi)$ of a μ-formula φ w.r.t. $L \subseteq \Sigma^\varepsilon$ is inductively given by:

$$\begin{array}{ll} \kappa_L(Z) = Z & \kappa_L(\neg\varphi) = \neg\kappa_L(\varphi) \\ \kappa_L(\phi \wedge \psi) = \kappa_L(\phi) \wedge \kappa_L(\psi) & \kappa_L(\phi \vee \psi) = \kappa_L(\phi) \vee \kappa_L(\psi) \\ \kappa_L([\sigma]\varphi) = [L^*][\sigma][L^*]\kappa_L(\varphi) & \kappa_L(\langle\sigma\rangle\varphi) = \langle L^*\rangle\langle\sigma\rangle\langle L^*\rangle\kappa_L(\varphi) \\ \kappa_L(\nu Z.\phi) = \nu Z.\kappa_L(\varphi) & \kappa_L(\mu Z.\phi) = \mu Z.\kappa_L(\varphi) \end{array}$$

Lemma 3. *Let $L \subseteq \Sigma$. Then $A/L \models_\varepsilon \varphi \iff A \models \kappa_L(\varphi)$.*

Proof. It follows directly from Definitions (5) and (4) and $\models_\varepsilon$. □

Example 3. Let A to be the automaton of Fig. 2(a) (example 1). We recall that $\Sigma_c = \{h_1, h_2\}$ and $\Sigma_u = \{l_1, l_2\}$. $CF(A\backslash\Sigma_c)$ is defined by the following equation system:

$$\begin{aligned} X_0 &= \langle l_1\rangle X_3 \wedge [l_1]X_3 \wedge [l_2]\mathsf{ff} \\ X_3 &= [l_1]\mathsf{ff} \wedge [l_2]\mathsf{ff} \end{aligned}$$

and $\kappa_{\Sigma_c}(CF(A))$ is defined by the following equation system:

$$\begin{aligned} X_0 &= \langle\Sigma_c^*\rangle\langle l_1\rangle\langle\Sigma_c^*\rangle X_3 \wedge [\Sigma_c^*][l_1][\Sigma_c^*]X_3 \wedge [\Sigma_c^*][l_2][\Sigma_c^*]\mathsf{ff} \\ X_3 &= [\Sigma_c^*][l_1][\Sigma_c^*]\mathsf{ff} \wedge [\Sigma_c^*][l_2][\Sigma_c^*]\mathsf{ff} \end{aligned}$$ ■

Moreover we can relate *weak bisimulation*, *hiding* and satisfiability of a characteristic formula:

Lemma 4. *Assume B is an automaton with no ε transitions and $L \subseteq \Sigma$. Then $A/L \approx_{\mathcal{W}} B \iff A \models \kappa_L(CF(B))$.*

Proof.

$$\begin{aligned} A/L \approx_{\mathcal{W}} B &\iff (A/L)^\varepsilon \approx_{\mathcal{S}} B \quad \text{[by definition of } \approx_{\mathcal{W}}\text{]} \\ &\iff (A/L)^\varepsilon \models CF(B) \quad \text{[by Lemma 1]} \\ &\iff A/L \models_\varepsilon CF(B) \quad \text{[by definition of } \models_\varepsilon\text{]} \\ &\iff A \models \kappa_L(CF(B)) \quad \text{[by Lemma 3]} \end{aligned}$$ □

Using the previous lemmas and the characteristic formula $CF(A)$ of a LTS, we can reduce the BSNNI-CP to a μ-CP:

Theorem 2. *BSNNI-CP is decidable and if the answer to the BSNNI-CP is "yes" a most permissive controller can be effecively synthesized* i.e. *BSNNI-CSP is computable.*

Proof. The BSNNI-CP is the following (Cf. equation (BSNNI-CP)):

Is there any controller C for A s.t. $(C \times A)/\Sigma_c \approx_{\mathcal{W}} (C \times A)\backslash\Sigma_c$?

As C does not restrict the uncontrollable actions, we have again that $(C \times A)\backslash\Sigma_c = A\backslash\Sigma_c$. We can then compute the characteristic formula for $A\backslash\Sigma_c$ and obtain $CF(A\backslash\Sigma_c)$. Applying Lemma 4 we obtain:

$$\begin{aligned} (C \times A)/\Sigma_c \approx_{\mathcal{W}} (C \times A)\backslash\Sigma_c &\iff (C \times A)/\Sigma_c \approx_{\mathcal{W}} A\backslash\Sigma_c \\ &\iff (C \times A) \models \kappa_{\Sigma_c}(CF(A\backslash\Sigma_c)) \end{aligned}$$

This enables us to reduce the BSNNI-CP to the following μ-CP:

$$\exists C \text{ s.t. } C \times A \models \kappa_{\Sigma_c}(CF(A \backslash \Sigma_c))$$

We have now a μ-CP to solve of the form $\exists C$ s.t. $C \times A \models \varphi$ with φ a μ-calculus formula. This can be done as stated in [12,13]. If A is controllable w.r.t. $\kappa_{\Sigma_c}(CF(A \backslash \Sigma_c))$ we can build a finite memory most permissive controller that satisfies BSNNI. □

As stated in section 3 the complexity of μ-CP is not yet known and thus we cannot obtain complexity result with our reduction.

Example 4 (Control of BSNNI). We use the automaton of Fig. 2(a) (example 1) to show how to synthesize a controller that ensures BSNNI. There are four possibilities for the most permissive controller C: either it allows $\{h_1, h_2\}$ or $\{h_1\}$ or $\{h_2\}$ or nothing. If it allows h_2, the initial state of $A \times C$ does not satisfy $[\Sigma_c^*][l_2][\Sigma_c^*]\mathsf{ff}$ in X_0 because after h_2 an action l_2 is enabled which is forbidden by $\kappa_{\Sigma_c}(CF(\mathcal{A} \backslash \Sigma_c))$. If it allows l_1, $\mathcal{A} \times C$ satisfies $\kappa_{\Sigma_c}(CF(\mathcal{A} \backslash \Sigma_c))$ and thus the most permissive controller is the one given by Fig. 2(b) again. ■

5 Conclusion

In this paper we have defined the control problems SNNI-CP and BSNNI-CP for the two types of non-interference properties SNNI and BSNNI. We have proved that SNNI-CP and BSNNI-CP are decidable and we solved both associated control synthesis problems *i.e.* given a system S, compute the *most permissive* non-interferent (sub)system of S. Our future work will consist in:

- extending our result to other types of non-interference (*e.g* TSNNI, PBNDC or BNDC ...);
- finding the exact complexity of both SNNI-CP and BSNNI-CP;
- implementing our framework. For SNNI there is BDD based tool [15] to solve the supervisory control problem. Concerning BSNNI and the μ-CP, the tool Synthesis [16] implements the theoretical setting of [12]. This would enable us to apply our results to reasonable size problems.
- extending our results to timed systems and timed non-interference to generalize and refine the results of [4].

Acknowledgments. Work supported by the French Government under the projects DOTS (`http://www.lsv.ens-cachan.fr/anr-dots/`) and CORTOS (`http://www.lsv.ens-cachan.fr/aci-cortos/`).

References

1. van der Meyden, R., Zhang, C.: Algorithmic verification of noninterference properties. In: Proceedings of the Second International Workshop on Views on Designing Complex Architectures (VODCA 2006). Electronic Notes in Theoretical Computer Science, vol. 168, pp. 61–75. Elsevier, Amsterdam (2006)

2. Agat, J.: Transforming out timing leaks. In: Proc. of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages(POPL'00), Boston, MA, USA, pp. 40–53. ACM Press, New York (2000)
3. Sabelfeld, A., Myers, A.: Language-based information-flow security. IEEE Journal on Selected Areas in Communications 21(1) (2003)
4. Gardey, G., Mullins, J., Roux, O.H.: Non-interference control synthesis for security timed automata. In: 3rd International Workshop on Security Issues in Concurrency (SecCo'05), San Francisco, USA. Electronic Notes in Theoretical Computer Science, Elsevier, Amsterdam (2005)
5. Kozen, D.: Results on the propositional -calculus. In: Nielsen, M., Schmidt, E.M. (eds.) Automata, Languages, and Programming. LNCS, vol. 140, pp. 348–359. Springer, Heidelberg (1982)
6. Arnold, A., Niwiński, D.: Rudiments of μ-calculus. In: Studies in Logic and the Foundations of Mathematics, vol. 146, North-Holland, Amsterdam (2001)
7. Aceto, L., Ingólfsdóttir, A.: Characteristic formulæ: From automata to logic. Technical Report RS-07-2, BRICS, Aarhus, Denmark (2007)
8. Müller-Olm, M.: Derivation of characteristic formulæ. In: Workshop on Mathematical Foundations of Computer Science, Brno, Slovakia. Electronic Notes in Theoretical Computer Science, vol. 18, Elsevier, Amsterdam (1998)
9. Ramadge, P., Wonham, W.: Supervisory control of a class of discrete event processes. SIAM J. Control Optim. 25(1) (1987)
10. Ramadge, P., Wonham, W.: The control of discrete event systems. In: Proceedings of the IEEE, IEEE Computer Society Press, Los Alamitos (1989)
11. Thomas, W.: On the synthesis of strategies in infinite games. In: Mayr, E.W., Puech, C. (eds.) STACS 95. LNCS, vol. 900, pp. 1–13. Springer, Heidelberg (1995)
12. Arnold, A., Vincent, A., Walukiewicz, I.: Games for synthesis of controllers with partial observation. Theor. Comput. Sci. 1(303), 7–34 (2003)
13. Riedweg, S., Pinchinat, S.: Quantified mu-calculus for control synthesis. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 642–651. Springer, Heidelberg (2003)
14. Focardi, R., Gorrieri, R.: Classification of security properties (part I: Information flow). In: Focardi, R., Gorrieri, R. (eds.) Foundations of Security Analysis and Design. LNCS, vol. 2171, pp. 331–396. Springer, Heidelberg (2001)
15. Hoffmann, G., Wong-Toi, H.: Symbolic synthesis of supervisory controllers. In: Proc. of the American Control Conference, Chicago, IL, pp. 2789–2793 (1992)
16. Point, G.: The synthesis toolbox - from modal automata to controller synthesis. Technical Report, 1342-05, LaBRI (2005) available at `http://www.labri.fr/publications/mvtsi/2005/Poi05`

Privacy-Preserving Credential Verification for Non-monotonic Trust Management Systems

Changyu Dong, Giovanni Russello, and Naranker Dulay

Department of Computing
Imperial College London
180 Queen's Gate, London, SW7 2AZ, UK
{changyu.dong,g.russello,n.dulay}@imperial.ac.uk

Abstract. Trust management systems provide a flexible way for performing decentralized security management. However, most trust management systems only support monotonic policies. Compared with non-monotonic policies, monotonic ones are less flexible and cannot express policies such as "Chinese wall policies" and "separation of duties". To support non-monotonic policies, trust management systems must be able to correctly identify the credentials which a subject has that are required by the policies. Previous efforts address the problem by letting the system query the issuers directly to verify the possession status of the credentials. But this approach can violate the subject's privacy. The main contribution of this paper is a cryptographic credential verification scheme for non-monotonic trust management systems that can correctly identify the credentials that a subject has while also protecting the subject's privacy. We also analyze the security of the scheme and prove that with correct construction and certain cryptographic assumptions, the scheme is secure.

Keywords: Trust Management, Non-monotonic Policy, Privacy, Cryptography.

1 Introduction

In the past ten years, we have seen the emergence of trust management systems for access control and privacy protection. Trust management systems have advantages in flexibility, scalability and extensibility over traditional security mechanisms and support decentralized security management for contemporary distributed computing environments.

Trust management was first proposed by Blaze *et al.* [1]. It aims to provide a unified approach to specify and interpret security policies, credentials and relationships that allow direct authorization of security-critical actions. The basic problem that they address is: "Does the set of credentials C prove that the request R complies with the local security policy P?"

Most trust management systems, such as [2,3,4,5,6], assume monotonicity: additional credentials can only result in the increasing of privilege. There are several reasons why monotonicity is a desirable property in trust management [7,4,8]. Firstly, monotonicity simplifies the design of trust management systems. The systems do not need to evaluate all potential policies and credentials, but are still provably correct and analyzable. Monotonicity also avoids policy conflicts [9,10] which are often caused by non-monotonicity. Furthermore, in some cases, non-monotonic policies can be converted into monotonic policies. For example, instead of defining a negative policy that requires credential C, one can define a positive policy to require a credential of "not have C".

The monotonic assumption oversimplifies the real world by cutting off the negative part, thus it cannot handle many important scenarios. For example, with monotonic semantics, it is hard to express explicit negative policies such as a consultant cannot serve company A and B at the same time because there is a conflict of interest (the Chinese Wall policy); a bank teller should not be an auditor of the same bank (Separation of Duties). Explicit negation is particularly important for authorization in distributed system scenarios, where the number of potential requesters is high. Without negations, we cannot express policies such as "allow all except some' elegantly.

Non-monotonicity allows more flexible and expressive security policies [11,12,10]. The difficulty with non-monotonic trust management systems is that the systems must have the exact set of credentials from an entity to make a sound decision. It is hard because of information asymmetry. If a subject knows or can predict that a certain credential will result in the decrease of its privileges, it may prefer not to reveal it. A trust management system cannot distinguish whether the absence of certain credentials is caused by "not having" or "not disclosing". To solve this problem, previous studies on non-monotonic trust management [13,14,15,16] suggest that the system should be able to collect credentials directly from the credentials issuers rather than only from the subjects. Although this approach seems to be able to solve the problem, it causes new problems. One problem is privacy: the issuer could disclose information about the subject, i.e. the credential, to anyone who wants the credential. It also requires issuers to be always online, which may not be practical.

To handle non-monotonicity in trust management systems, we present a cryptographic credential verification scheme which guarantees that the trust management system can identify all the required credentials possessed by the subject while also providing protecting the subject's privacy.

2 Problem Definition

Let's consider a trust management system controlling access to a resource. Let $\mathcal{V}$ be the set of atomic privileges, $\mathcal{C}_p$ be the set of all credentials relevant to the trust decision, the trust policies can be formalized as $pol : \mathcal{P}(\mathcal{C}_p) \rightarrow \mathcal{P}(\mathcal{V})$, where $\mathcal{P}(\mathcal{C}_p)$ and $\mathcal{P}(\mathcal{V})$ are the power sets of $\mathcal{C}_p$ and $\mathcal{V}$ respectively. Loosely speaking, this means that given a set of relevant credentials, the trust management system can decide a set of privileges based on its local trust policies.

If the policies are monotonic, then we have $\mathcal{C}_1 \subseteq \mathcal{C}_2 \rightarrow pol(\mathcal{C}_1) \subseteq pol(\mathcal{C}_2)$ for all $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{P}(\mathcal{C}_p)$. In contrast, if the policies are non-monotonic, then there exists $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{P}(\mathcal{C}_p)$ such that $\mathcal{C}_1 \subseteq \mathcal{C}_2 \wedge pol(\mathcal{C}_1) \nsubseteq pol(\mathcal{C}_2)$.

One required security property of trust management systems is that the subject should not receive excessive privileges. In other words, for a subject who has a set of credentials $\mathcal{C}_s$, the privileges it can get should be bound by $pol(\mathcal{C}_r)$ where $\mathcal{C}_r = \mathcal{C}_s \cap \mathcal{C}_p$ are the credentials possessed by the subject and required by the policies (see Fig. 1). It seems trivial since given $\mathcal{C}_p$, for each set of credentials $\mathcal{C}_s$, there is exactly one $\mathcal{C}_r$. However, in most cases, the system only knows $\mathcal{C}_{s'}$ which is a set of credentials collected by it. The policy evaluation is therefore based on $\mathcal{C}_{r'} = \mathcal{C}_{s'} \cap \mathcal{C}_p$ rather than $\mathcal{C}_r$. The credentials are digital assertions signed by the credential issuers and are unforgeable, i.e. $\mathcal{C}_{s'} \subseteq \mathcal{C}_s$. In consequence, it is clear that $\mathcal{C}_{r'} \subseteq \mathcal{C}_r$.

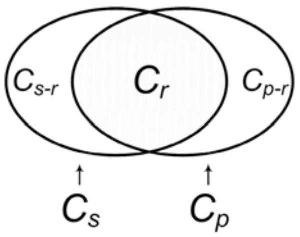


Fig. 1. Credential Sets

In monotonic trust management systems, the required property is preserved in all situations. Since $C_{r'} \subseteq C_r$, by monotonicity, $pol(\mathcal{C}_{r'}) \subseteq pol(\mathcal{C}_r)$ is always true. But in non-monotonic trust management systems, the potential privileges the subject can get is bound by $\bigcup_{\mathcal{C}_{ri'} \in \mathcal{P}(\mathcal{C}_r)} pol(\mathcal{C}_{ri'})$. This means that if the system cannot identify $\mathcal{C}_r$ correctly, the subject may get more privileges than it should. As a result, credential collection and verification is crucial to non-monotonic trust management systems.

In monotonic trust management systems, credentials are usually submitted by the subjects. This is obviously not appropriate in non-monotonic trust management systems. A scheme that most non-monotonic trust management systems use is to actively collect and verify the credentials. For each credential $c_i \in \mathcal{C}_p$, the system sends a query to the credential issuer. The issuer returns a positive reply if it has issued c_i to the subject, a negative reply otherwise. If the issuer's reply is positive, the system can infer that $c_i \in \mathcal{C}_s$ and in consequence, $c_i \in \mathcal{C}_r$. If the reply is negative, then $c_i \in \mathcal{C}_{p-r}$, where $\mathcal{C}_{p-r}$ is the set of credentials that were required by the policies but not possessed by the subject. After receiving definite replies for all the credentials in $\mathcal{C}_p$, the system identifies the correct set $\mathcal{C}_r$.

The scheme is problematic in the sense that it considers little about the subject's privacy. Credentials may contain sensitive information about the subject, but there is no way to prevent a malicious system from probing the credentials the subject has, e.g. the system can query about a credential in $\mathcal{C}_{s-r}$, which is not relevant to the trust decision at all. It can be even worse since the query is open to everyone. Another noticeable defect is that the query may not always get a definite reply. A query can go unanswered if the issuer is offline. In such situations, the system cannot verify the possession status of the credential.

In the following sections, we will describe a new credential verification scheme designed for non-monotonic trust management systems. The scheme allows the system to identify $\mathcal{C}_r$ efficiently and correctly. The scheme also protects the subject's privacy. The verification must first be permitted by the subject, and after the verification, the system knows nothing about the credentials in $\mathcal{C}_{s-r}$.

3 Credential Verification Scheme

Our credential verification scheme tries to keep a balance between avoiding unnecessary security breaches caused by lack of information and respecting the users right of controlling their information. In sections 6 and 6 we will prove that the scheme is:

- Correct. The scheme can correctly identify all the credentials that the subject has that are required by the target. And also,
- Privacy-preserving. The verification is controlled by the subject. Without the permission of the subject, the target cannot learn anything about the credentials possessed by the subject.

3.1 Overview of the Scheme

There are three roles in our scheme:

- Subjects: The subjects are entities who send access requests and need to be authorized.
- Targets: The targets are entities who provide resources and make the trust decisions.
- Credential issuers: Issuers create the credentials, and also credential profiles (see section 3.2) to allow the targets to identify the credentials possessed by the subjects.

As described earlier, the aim of a credential verification scheme for non-monotonic trust management systems is to identify the correct $\mathcal{C}_r$. The traditional scheme achieves the goal by finding two mutually exclusive credential sets $\mathcal{C}_r$ and $\mathcal{C}_{p-r}$ such that $\mathcal{C}_p = \mathcal{C}_r \cup \mathcal{C}_{p-r}$. This approach relies totally on the target to verify the credentials. Our approach is different. In our scheme, we let the subject provide a credential set $\mathcal{C}'_r$ such that $\mathcal{C}'_r \subseteq \mathcal{C}_p$ and $\mathcal{C}'_r \subseteq \mathcal{C}_s$. Then the subject must convince the target that $\mathcal{C}'_r = \mathcal{C}_r$ by proving $\mathcal{C}_{p-r'} \cap \mathcal{C}_{s-r'} = \emptyset$, where $\mathcal{C}_{p-r'} = \mathcal{C}_p - \mathcal{C}'_r$, $\mathcal{C}_{s-r'} = \mathcal{C}_s - \mathcal{C}'_r$.

To see that this is correct, first let's assume that when $\mathcal{C}_{p-r'} \cap \mathcal{C}_{s-r'} = \emptyset$, $\mathcal{C}'_r \neq \mathcal{C}_r$. Because $\mathcal{C}'_r \subseteq \mathcal{C}_p$ and $\mathcal{C}'_r \subseteq \mathcal{C}_s$ and $\mathcal{C}_r = \mathcal{C}_p \cap \mathcal{C}_s$, the only possibility of $\mathcal{C}'_r \neq \mathcal{C}_r$ is $\mathcal{C}'_r \subset \mathcal{C}_r$, therefore we can find a non-empty credential set $\mathcal{C}''_r$ such that $\mathcal{C}''_r \cap \mathcal{C}'_r = \emptyset$ and $\mathcal{C}''_r \cup \mathcal{C}'_r = \mathcal{C}_r$. Then it follows that $\mathcal{C}_{p-r'} \cap \mathcal{C}_{s-r'} = \mathcal{C}''_r$, which contradicts the assumption. So $\mathcal{C}'_r = \mathcal{C}_r$ must be true.

The difficulty with our scheme is how to preserve the privacy of the subject, namely how to effectively prove $\mathcal{C}_{p-r'} \cap \mathcal{C}_{s-r'} = \emptyset$ without letting the target know any credentials in $\mathcal{C}_{s-r}$. We address the problem by constructing a cryptographic bijection mapping $\rho : \mathcal{C}_s \rightarrow \mathcal{E}_s$. $\mathcal{E}_s$ is publicly available to any entity through a highly available P2P directory service. ρ is constructed using well-defined cryptographic primitives, so $\mathcal{E}_s$ discloses no information about $\mathcal{C}_s$ to the targets. The subject must prove that for any credential $c_i \in \mathcal{C}_{p-r'}$, $c_i \notin \rho^{-1}(\mathcal{E}_{s-r'})$. Because ρ is a bijection, so $\rho^{-1}(\mathcal{E}_{s-r'}) = \rho^{-1}(\rho(\mathcal{C}_{s-r'})) = \mathcal{C}_{s-r'}$. Therefore the above proof is equivalent to proving $\mathcal{C}_{p-r'} \cap \mathcal{C}_{s-r'} = \emptyset$. The proof is zero-knowledge, thus at the end, the target can be convinced about the statement but knows nothing more.

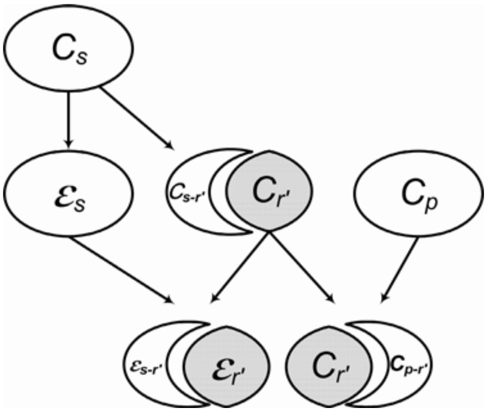


Fig. 2. Overview of our approach

3.2 Architecture

In our scheme, $\mathcal{E}_s$ is implemented as a *credential profile* (or profile for simplicity) which allows targets to verify which credentials the subjects has. Each credential profile has four basic components. The components are:

- ID Hash: The hash value of the subject's identity. It can be used to search all the profiles associated with the identity.
- Profile Entries: Each entry is *linked* to a credential held by the subject and contains some encrypted information. The target can verify that the subject has the linked credential by performing a computation on the entry. Profile entries are discussed in more detail later.
- Timestamp: The time when this profile was created. It allows entities to determine which profile is the latest.
- Signature: The digital signature of the issuer for this profile. This signature is used to ensure that no one can modify the profile after it has been created.

The profiles are distributed independently of the credentials through a P2P directory service. The P2P directory service is maintained by the credential issuers and can be used by any entity. The advantage of the P2P approach is that each profile can be duplicated and stored in multiple places over a wide area. Therefore it provides higher availability of the profiles than storing them in one place.

To ensure that all credential information is preserved in the profile, we use an "onion" structure for the profile. If the subject has n credentials, its profile has $n + 1$ layers. The innermost layer of the profile, layer 0, is the ID hash of the subject. Each layer i, consists of a profile entry, a timestamp and is wrapped by an overall signature on the content of layer 0 to layer i. The onion structure is built up along with the updating process of the subject's credential set. As shown in Fig. 3, each time the subject needs a credential, it contacts the credential issuer (1). The credential issuer generates a credential for the subject, at the same time it must also create a new profile for the subject. To do so, the credential issuer first needs to obtain the latest version of the subject's profile. This can be done by searching the P2P directory service using the hash value of the subject's identity (2). After getting the latest profile (3), the issuer generates a new entry for the credential it is issuing to the subject and also a timestamp, then appends them to the old profile. The issuer then signs the new profile and releases it to the subject (4) with the credential and also to the P2P directory service (5). As we can see, by using the onion structure, we ensure that the next time that a credential issuer creates a profile, it cannot modify or remove any content from the old profile. Suppose it modifies the content in layer k, it would then need to forge all the signatures from layer k to layer n. We also require the peers in the P2P directory service to check the contents of a newer versions of a profile with their local version, and reject them if the checking fails.

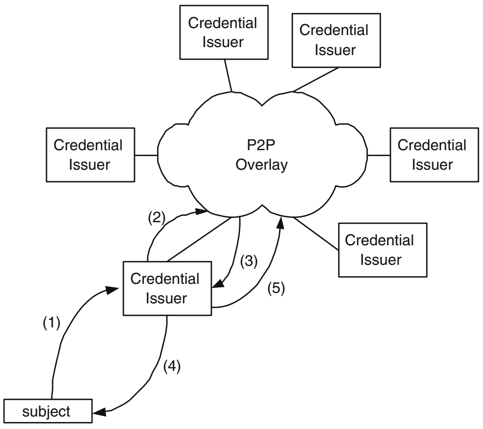


Fig. 3. Issuing a credential using the P2P directory service

3.3 Cryptographic Building Blocks

Our scheme is realised by using cryptographic techniques. In this section, we briefly outline the cryptographic primitives used. In section 4, we will show how the security of our scheme follows the security properties of the primitives. For more information about these cryptographic primitives and their formal security definition, please refer to [17,18,19]. The cryptographic primitives used are:

1. *A commitment scheme*, $Commit : \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^l$. It is a two-phase protocol between two parties, the committer and the receiver. In the first phase (the commitment phase), the committer commits to $r \in \{0,1\}^n$ by choosing a Secret $s \in \{0,1\}^k$ to generate a commitment $Commit_s(r)$ through a polynomial time algorithm which binds r to $Commit_s(r)$, i.e. it's infeasible for the receiver to find r' and s' which produce the same commitment $Commit_{s'}(r') = Commit_s(r)$ (this is the *binding property*). The committer sends $Commit_s(r)$ to the receiver. Given only the commitment, it's infeasible for the receiver to compute the committed string r (this is the *hiding property*). In the second phase (the open phase), the committer reveals r and s to the receiver. Now the receiver checks whether they are valid against the commitment, if the receiver can compute $Commit_s(r)$ from r and s, then it is convinced that r was indeed committed by the committer in the first phase, otherwise it rejects r.
2. *Zero-Knowledge Proof Protocols.* Let P, V be two Interactive Turing Machines, L be a language over $\{0,1\}^*$, the goal of a zero-knowledge proof protocol (P, V) is to allow the prover P to prove to the verifier V that a given x belongs to language L, without disclosing any other information.
   In the following sections, we will use the notion introduced in [20] for the zero-knowledge proofs. The convention is that the elements listed in the round brackets before ":" denote the knowledge to be proved to the verifier and all other parameters are known to the verifier. For example: $PK\{(a, b) : y = g^a h^b\}$ means a zero-knowledge proof of integers a, b such that $y = g^a h^b$ holds and y, g, h are known to the verifier.

3.4 Profile Entry

Profile entries (or entries for simplicity) can be used by the target to verify that the subject possesses the corresponding credential. Entries are generated by using cryptographic techniques so that one cannot learn anything about the credentials from the entries, unless following the credential verification protocol.

We assume that there is a common vocabulary for specifying credentials which is used by all the entities in the system. We also assume that each credential has a credential name, e.g. student, member etc.. An entry is linked to a credential whose name is c and generated by the credential issuer when it issues the credential. To generate an entry, the creator (the credential issuer) first creates a commitment for the credential name $Commit_s(c)$. The secret s for opening the commitment will be sent to the subject through a secure channel. The entry is a tuple $(Commit_s(c), Sig(cred), exp\_time)$. $Sig(cred)$ is the signature of the linked credential and is used to associate the entry with the real credential. exp_time is the expiration time of the credential. An entry can be revoked implicitly or explicitly. Each entry contains the expiration time of the linked credential. The entry becomes invalid when the credential expires. When a credential is revoked before expiring, the credential issuer puts the signature of the credential into a revocation list, and publishes it into the P2P network.

We use a modified Pedersen Commitment Scheme which is slightly different from the standard one.

Setup. The issuer chooses two large prime numbers p and q such that q divides $p-1$. Let g be a generator of G_q, the unique order-q subgroup of $\mathbb{Z}_p^*$. The issuer chooses x uniformly randomly from $\mathbb{Z}_q$ and computes $h = (g^x \bmod p)$. The issuer keeps the value x secret and makes the values p, q, g, h public.

Commit. The issuer chooses s uniformly randomly from $\mathbb{Z}_q$ and computes the commitment $Commit_s(c) = g^c h^s$.

There are three parties involved: a committer (the issuer), a prover (the subject) and a receiver(the target). The issuer generates the commitment to c and lets the subject know the secret for opening the commitment. This is because the binding property can only guarantee that after generating the commitment, the committer cannot change what it committed to; however, it provides no guarantee on what can be committed to. If we let the subject generate the commitment, it could commit to another credential name c' rather than c and it could take advantage from this, i.e. to hide the credential in order to gain excessive privileges. So we let a trusted third party (the issuer) generate the commitment to ensure that the commitment is indeed a commitment to c. Note that the subject does not need to open the commitment. What the subject needs to do is to use the commitment and the secret s to convince the target by a zero-knowledge proof protocol that the linked credential is not required.

3.5 Credential Verification Protocol

The protocol is described in the following and shown in a message sequence chart in Fig. 4. Note: any party can terminate the process if a malicious behavior is detected during the protocol. If the protocol terminates prematurely, it will output "$\perp$".

1. The target receives a request from the subject.
2. The target decides the credentials that need to be checked according to its local policies, i.e. $\mathcal{C}_p$, and lets the subject know $\mathcal{C}_p$.
3. The subject decides $\mathcal{C}_r = \mathcal{C}_p \cap \mathcal{C}_s$. If there is any credential in $\mathcal{C}_r$ which is sensitive and the subject does not want the target to know, it can choose to refuse and terminate the process. Otherwise, the subject proceeds. Note: as we have mentioned before, $\mathcal{C}_p$ is the set of all the credentials the target needs to check according to its local policies, not the the credentials the subject must have, so even if the subject does not have all the credentials requested, it can still choose to proceed.

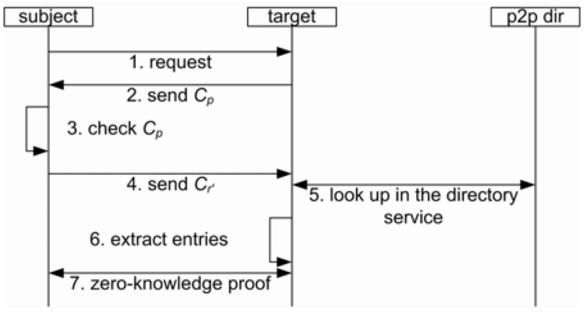


Fig. 4. Message sequence chart of the protocol for credential verification

4. If the subject chooses to proceed, it sends $\mathcal{C}'_r = \mathcal{C}_r$ to the target. The target checks the validity of the credentials in $\mathcal{C}'_r$. If $\mathcal{C}'_r$ contains all the credentials that appear in $\mathcal{C}_p$, i.e. $\mathcal{C}'_r = \mathcal{C}_p$, the protocol terminates and outputs $\mathcal{C}'_r$, otherwise the target creates the set $\mathcal{C}_{p-r'} = \mathcal{C}_p - \mathcal{C}'_r$ and the protocol continues.
5. The target obtains the subject's latest credential profile P from the P2P directory service.
6. The target creates a set containing all the valid entries extracted from P. Valid means not expired or revoked. The set is effectively equal to $\mathcal{E}_s$. For each valid entry $(Commit_{s_j}(c_j), Sig(cred_j), exp\_time_j)$, if the target can find a credential in $\mathcal{C}'_r$ whose signature is $Sig(cred_j)$, then this entry is removed from $\mathcal{E}_s$. At the end, the target will have the set $\mathcal{E}_{s-r'}$.
7. For each entry $(Commit_{s_k}(c_k)$, $Sig(cred_k)$, $exp\_time_k)$ in $\mathcal{E}_{s-r'}$, the subject must run a zero-knowledge proof protocol as described in section 3.6 to convince the target that there is no credential in $\mathcal{C}_{p-r'}$ whose name is c_k. The target will then know that the credential is not required by its policies, but nothing more than that.
8. The credential verification protocol completes by outputting $\mathcal{C}'_r$.

3.6 Zero-Knowledge Proof Protocol

The zero-knowledge proof protocol that we use in step 7 above is adapted from [21]. The original protocol is a two-party secure computation protocol used to compare two integers. We simplify the settings because there is only one secure input, which is the name of the credential held by the subject.

The aim of this protocol is: given public security parameters p, q, g, h, a commitment $g^c h^s$ as described in 3.4, and a different credential name $c' \in \mathbb{Z}_q$, the subject must prove $c' \neq c$ to the target.

We use two well-defined sub-protocols in the proof. The first one is Schnorr's protocol [22] $PK\{(x) : y = g^x\}$ which proves knowledge of a discrete logarithm. The other is Okamoto's protocol [23] $PK\{(a,b) : y = g_1^a g_2^b\}$ which proves knowledge of how to open a commitment. It can be easily extended to $PK\{(a,b) : (x = g_1^a) \wedge (y = g_2^a g_3^b)\}$. The zero-knowledge proof protocol is:

1. The subject uniformly randomly chooses $x \in \mathbb{Z}_q$, x is kept secret. It computes $h_2 = h^x$ and sends it to the target. Then it proves to the target that it knows x by $PK\{(x) : h_2 = h^x\}$.
2. The subject computes $P_s = h_2^s$ where s is the secret to open the commitment $Commit_s(c) = g^c h^s$ and sends it to the target. Then it proves to the target that it knows c, s and that it used the same s in computing h_2^s as in computing $g^c h^s$ by $PK\{(c, s) : (P_s = h_2^s) \wedge (Commit_s(c) = g^c h^s)\}$.
3. The target also selects a random element $s' \in \mathbb{Z}_q$ and computes $P_{s'} = h_2^{s'}$ and $Commit_{s'}(c') = g^{c'} h^{s'}$. The target sends $Commit_{s'}(c')$ to the subject as a challenge.
4. The subject computes $Q = (\frac{Commit_s(c)}{Commit_{s'}(c')})^x$, sends the result to the target, and proves $PK\{(x) : h_2 = h^x \wedge Q = (\frac{Commit_s(c)}{Commit_{s'}(c')})^x\}$.
5. Finally, the target checks whether $Q \neq \frac{P_s}{P_{s'}}$ holds. If so, then $c' \neq c$

We have implemented a proof-of-concept prototype in Java 1.5 and done a preliminary performance evaluation. The evaluation was done on a Pentium IV 3.2 GHz (dual core) desktop with 1 GB memory. The execution of the zero-knowledge protocol takes about 130 milliseconds. The result was obtained by averaging the time for 1000 executions. The performance can be further improved by optimizing the code.

4 Security Analysis

In our credential verification protocol, the security requirements of the parties are different. For the subject, the requirement is for the scheme to prevent a malicious target from learning excessive information about its credentials. For the target, the requirement is for the scheme to correctly identify the subject's credentials and prevent a malicious subject from cheating about the credentials it has.

To reflect the requirements of both parties, we define the security of our protocol as follows:

1. Correctness. Given the target follows the protocol, if the subject sends $\mathcal{C}'_r = \mathcal{C}_r$ to the target, then at the end, the protocol should output $\mathcal{C}_r$ with overwhelming probability; otherwise, the protocol should output $\perp$ with overwhelming probability.
2. Privacy-preserving. Given the subject follows the protocol, the target should learn either nothing or the set of credentials $\mathcal{C}_r$. In the latter case, it should be computationally infeasible for a malicious target to learn any credential in $\mathcal{C}_{s-r}$.

The proofs can be found in the appendix.

5 Related Work

There are few non-monotonic trust management systems. REFEREE [13] is a trust management system for web applications. It uses PICS labels [24] as credentials and assumes they can be obtained from authorities' websites. The system is responsible for collecting all the credentials, therefore it is possible to gather complete information. TPL [15] allows negative credentials which are interpreted as suggestions of "not to trust". In TPL, positive credentials are submitted by the subject, and the negative credentials are collected by the system. [25] discusses non-monotonic access policies in trust negotiation and argues that to avoid relying on outside information, the system should only have non-monotonic policies according to its local information and the credentials submitted to the system should be monotonic. A recent study [16] adds a restricted form of negation to the standard RT trust management language. But as we have mentioned in section 1, none of these systems address privacy or availability issues.

6 Conclusion and Future Work

In this paper we discussed the benefits and problems of non-monotonic trust management systems. To handle non-monotonicity, we developed a credential verification scheme which guarantees that the system can identify all the required credentials possessed by the subject while also protecting the subject's privacy. The scheme is implemented by using several cryptographic primitives. We also analyzed our scheme and proved that with correct construction and certain cryptographic assumptions, our scheme is secure.

One aspect of our future work is to allow more expressive trust policies. Currently, our scheme does not support wildcard credential names in policies, for example, the policy "a subject can access the patient's medical record if it has a doctor credential signed by any hospital". At present, such policies cannot be handled directly. The example has to be translated into verifying the subject has at least one credential in the list of all doctor credentials signed by a hospital. This approach is static and also increases the computation time because in the worst case, all the credential names that appear in the list need to be verified.

We will also investigate using the cryptographic credential verification scheme in automated trust negotiation [26]. Automated trust negotiation is a promising approach to build trust management systems in a privacy-preserving way. Disclosure policies are established to regulate the disclosure of sensitive credentials and policies. Traditional trust negotiation systems disclose the credentials incrementally, therefore must be monotonic. They are also subject to policy cycles where a negotiator A has a disclosure policy that requires credential c_1 from another negotiator B before disclosing credential c_2 while B has a disclosure policy that requires credential c_2 from A before disclosing credential c_1. When there is a policy cycle, the negotiation fails. In [27], the authors propose the *Reverse Eager (RE) trust negotiation strategy* in which two negotiators start from the maximum credentials sets and in each iteration prune the credentials sets by removing the unusable credentials according to their own policies. The RE strategy is cycle-tolerant which means even with policy cycles, the negotiation can still succeed. But the RE strategy does not support non-monotonic policies and the trust negotiation protocol requires intensive computation. We are looking at developing a more efficient protocol using our credential verification scheme and the RE strategy for non-monotonic and cycle-tolerant trust negotiation.

Acknowledgments. This research was supported by the UK's EPSRC research grant EP/C537181/1 and forms part of CareGrid, a collaborative project with the University of Cambridge. The authors would like to thank the members of the Policy Research Group at Imperial College for their support.

References

1. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized trust management. In: SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 164–173. IEEE Computer Society Press, Los Alamitos (1996)
2. Blaze, M., Feigenbaum, J., Ioannidis, J., Keromytis, A.: Rfc2704: The keynote trust-management system version 2 (1999)
3. Jim, T.: Sd3: A trust management system with certified evaluation. In: SP '01: Proceedings of the 2001 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 106–115. IEEE Computer Society Press, Los Alamitos (2001)
4. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust-management framework. In: SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy, Washington, DC, USA, pp. 114–130. IEEE Computer Society Press, Los Alamitos (2002)
5. Hess, A., Seamons, K.E.: An access control model for dynamic client-side content. In: SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies, pp. 207–216. ACM Press, New York (2003)
6. Carbone, M., Nielsen, M., Sassone, V.: A formal model for trust in dynamic networks. In: SEFM, pp. 54–61. IEEE Computer Society Press, Los Alamitos (2003)
7. Blaze, M., Feigenbaum, J., Strauss, M.: Compliance checking in the policymaker trust management system. In: Proceedings of the Second International Conference on Financial Cryptography, London, UK, pp. 254–274. Springer, Heidelberg (1998)
8. Seamons, K., Winslett, M., Yu, T., Smith, B., Child, E., Jacobson, J., Mills, H., Yu, L.: Requirements for policy languages for trust negotiation. In: POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), Washington, DC, USA, pp. 68–79. IEEE Computer Society Press, Los Alamitos (2002)
9. Lupu, E.C., Sloman, M.: Conflicts in policy-based distributed systems management. IEEE Trans. Softw. Eng. 25(6), 852–869 (1999)
10. Jajodia, S., Samarati, P., Subrahmanian, V.S., Bertino, E.: A unified framework for enforcing multiple access control policies. In: SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data, pp. 474–485. ACM Press, New York (1997)
11. Clark, D.D., Wilson, D.R.: A comparison of commercial and military computer security policies. In: IEEE Symposium on Security and Privacy, pp. 184–195. IEEE Computer Society Press, Los Alamitos (1987)
12. Brewer, D.F.C., Nash, M.J.: The chinese wall security policy. In: IEEE Symposium on Security and Privacy, pp. 206–214. IEEE Computer Society Press, Los Alamitos (1989)
13. Chu, Y.H., Feigenbaum, J., LaMacchia, B., Resnick, P., Strauss, M.: Referee: trust management for web applications. Comput. Netw. ISDN Syst. 29(8-13), 953–964 (1997) 283252

14. Li, N., Feigenbaum, J., Grosof, B.N.: A logic-based knowledge representation for authorization with delegation (extended abstract). In: Proceedings of the 1999 IEEE Computer Security Foundations Workshop, pp. 162–174. IEEE Computer Society Press, Los Alamitos (1999)
15. Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., Ravid, Y.: Access control meets public key infrastructure, or: assigning roles to strangers. In: the 2000 IEEE Symposium on Security and Privacy, Berkeley, CA, pp. 2–14. IEEE Computer Society Press, Los Alamitos (2000)
16. Czenko, M., Tran, H., Doumen, J., Etalle, S., Hartel, P., den Hartog, J.: Nonmonotonic trust management for p2p applications. Electronic Notes in Theoretical Computer Science 157(3), 113–130 (2006)
17. Goldreich, O.: Foundations of Cryptography. vol. I. Basic Tools, Cambridge University Press, Cambridge (2001)
18. Goldwasser, S., Bellare, M.: Lecture notes on cryptography `http://www-cse.ucsd.edu/users/mihir/papers/gb.pdf`
19. Goldreich, O.: Foundations of Cryptography. vol. II. Basic Applications Cambridge University Press, Cambridge (2004)
20. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups (extended abstract). In: Jr., B.S.K. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
21. Boudot, F., Schoenmakers, B., Traoré, J.: A fair and efficient solution to the socialist millionaires' problem. Discrete Applied Mathematics 111(1-2), 23–36 (2001)
22. Schnorr, C.P.: Efficient signature generation by smart cards. J. Cryptology 4(3), 161–174 (1991)
23. Okamoto, T.: Provably secure and practical identification schemes and corresponding signature schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
24. Resnick, P., Miller, J.: Pics: Internet access controls without censorship. Commun. ACM 39(10), 87–93 (1996)
25. Dung, P.M., Thang, P.M.: Trust negotiation with nonmonotonic access policies. In: Aagesen, F.A., Anutariya, C., Wuwongse, V. (eds.) INTELLCOMM 2004. LNCS, vol. 3283, pp. 70–84. Springer, Heidelberg (2004)
26. Winsborough, W.H., Seamons, K.E., Jones, V.E.: Automated trust negotiation. In: DARPA Information Survivability Conference and Exposition, 2000, pp. 88–102. IEEE Computer Society Press, Los Alamitos (2000)
27. Frikken, K.B., Li, J., Atallah, M.J.: Trust negotiation with hidden credentials, hidden policies, and policy cycles. In: NDSS, The Internet Society (2006)
28. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)

Appendix

Security Proof of the Zero-Knowledge Proof Protocol

Lemma 1. *The protocol in section 3.6 is complete: if $c' \neq c$, then $Pr[(P,V)(c' \neq c) = 1] = 1$.*

Proof. The zero-knowledge proof convinces the verifier by comparing $Q = (\frac{Commit_s(c)}{Commit_{s'}(c')})^x = (\frac{g^c h^s}{g^{c'} h^{s'}})^x = g^{(c-c')x} h^{(s-s')x}$ and $\frac{P_s}{P_{s'}} = \frac{h^{sx}}{h^{s'x}} = h^{(s-s')x}$ where c

is a credential name and c' is another credential name. $Q \neq \frac{P_s}{P_{s'}}$ holds only when $c' \neq c$, therefore the protocol is complete.

Lemma 2. *The protocol in section 3.6 is sound: if* $c' = c$, *then* $\forall P'\ Pr[(P', V)(c' \neq c) = 1] \leq \delta$, *where* δ *is a negligible probability.*

Proof. If a malicious prover can manipulate $Q, P_s, P_{s'}$, then it can control the result of the zero-knowledge proof protocol. For example, if the prover can construct $P_{s'} = h^{s''x}$ using $s'' \neq s'$, then $Q \neq \frac{P_s}{P_{s'}}$ even $c = c'$. But a cheating prover can succeed with only a negligible probability. Firstly, h^x is revealed to the verifier and proved to be constructed correctly in step 1 using Schnorr's protocol. In step 2, the prover must prove it uses the same s in computing $P_s = (h^x)^s$ as in computing $g^c h^s$ using the extended Okamoto protocol. $Commit_s(c)$ is known by the target and $Commit_{s'}(c')$ is computed by the target, and in step 5, the prover must prove that it uses the same x in computing h^x and $Q = (\frac{Commit_s(c)}{Commit_{s'}(c')})^x$ using Schnorr's protocol. To manipulate h^x, P_s and Q, a malicious prover must break Schnorr's protocol or the extended Okamoto protocol. But in the two subprotocols, the challenges are chosen randomly from $[1, 2^t]$, so the probability of successful cheating is at most 2^{-t}. When t is sufficiently large, the probability is negligible. Therefore h^x, Q and P_s must be constructed correctly with a overwhelming probability. The prover cannot manipulate $P_{s'} = (h^x)^{s'}$ because s' is selected by the verifier. So the protocol is sound.

Lemma 3. *Under the Discrete Logarithm Assumption and the Decisional Diffie-Hellman Assumption, the protocol in section 3.6 is zero-knowledge.*

Proof. The execution of the protocol produces a view in the form $\{h_2 = h^x, P_s = h_2^s, s', Commit_{s'}(c'), Q = (\frac{Commit_s(c)}{Commit_{s'}(c')})^x\}$. Following the definition of zero-knowledge, we need to show that there exists a probabilistic polynomial time simulator M^* which can produce a simulation of a view. Note that because the simulator of the main protocol can call the simulators of the sub-protocols, and because the sub-protocols have been proven to be zero-knowledge, we omit views of the sub-protocols here.

We can construct M^* as follows:

1. The public input is $p, q, g, h, g^c h^s, c'$.
2. M^* randomly chooses $x^* \in \mathbb{Z}_q$, and compute $h_2^* = h^{x^*}$.
3. M^* randomly chooses $s'^* \in \mathbb{Z}_q$, and computes $P_{s'}^* = (h_2^*)^{s'^*}$ and $Commit_{s'}(c')^* = g^{c'} h^{s'^*}$.
4. M^* computes $Q^* = (\frac{Commit_s(c)}{Commit_{s'}(c')^*})^{x^*}$.
5. M^* chooses s^* such that $(h_2^*)^{s^*} \neq Q^* P_{s'}^*$, then let $P_s^* = (h_2^*)^{s^*}$.
6. M^* outputs $\{h_2^*, P_s^*, s'^*, Commit_{s'}(c')^*, Q^*\}$

It is easy to see that under the Discrete Logarithm Assumption and the Decisional Diffie-Hellman Assumption, the simulation is computationally indistinguishable from a view produced in the protocol. So our protocol is zero-knowledge.

Proof of Correctness of Credential Verification Protocol

Theorem 1. *The credential verification protocol is correct.*

Proof. Firstly we need to prove that in the protocol, if the subject sends $\mathcal{C}'_r = \mathcal{C}_r$ to the target, then at the end, the protocol should output $\mathcal{C}_r$ with overwhelming probability. It is clear that in step 4, if an honest subject sends $\mathcal{C}'_r = \mathcal{C}_r$ to the target, then there are two cases to consider:

Case 1: $\mathcal{C}_r = \mathcal{C}_p$. In this case, the target will detect $\mathcal{C}_{r'} = \mathcal{C}_p$, and will terminate the protocol and output $\mathcal{C}_{r'}$, which equals $\mathcal{C}_r$.

Case 2 $\mathcal{C}_r \subset \mathcal{C}_p$. In this case, the target has $\mathcal{C}_{p-r'} = \mathcal{C}_p - \mathcal{C}_{r'} = \mathcal{C}_p - \mathcal{C}_r = \mathcal{C}_{p-r}$ in step 4. In step 6 the target will have $\mathcal{E}_{s-r'} = \rho(\mathcal{C}_s - \mathcal{C}_{r'}) = \rho(\mathcal{C}_s - \mathcal{C}_r) = \rho(\mathcal{C}_{s-r})$. We have proven that the zero-knowledge proof protocol is complete in Lemma 1, therefore in step 7, the subject can prove that for each entry $(Commit_{s_k}(c_k),\ Sig(cred_k),\ exp\_time_k)$ in $\mathcal{E}_{s-r'}$, that there is no credential in $\mathcal{C}_{p-r'}$ whose name is c_k. Then in step 8, the protocol outputs $\mathcal{C}_{r'}$, which equals $\mathcal{C}_r$.

Next we will prove that if the subject sends $\mathcal{C}'_r \neq \mathcal{C}_r$ to the target, then at the end, the protocol should output $\perp$ with overwhelming probability.

In step 4, if a malicious subject sends $\mathcal{C}'_r \neq \mathcal{C}_r$ to the target, there are also two cases to consider:

Case 1: $\mathcal{C}_{r'} \nsubseteq \mathcal{C}_r$. In this case, there exists at least one credential c such that $c \in \mathcal{C}_{r'}$ and $c \in \mathcal{C}_{s-r}$. If $\mathcal{C}_{r'} = \mathcal{C}_r$, then all the credentials in $\mathcal{C}_{r'}$ must also be in $\mathcal{C}_p$, but now c is not in $\mathcal{C}_p$ because it is in $\mathcal{C}_{s-r}$. Therefore the target can detect $\mathcal{C}'_r \neq \mathcal{C}_r$ easily. The target will then terminate the protocol and output $\perp$.

Case 2: $\mathcal{C}_{r'} \subset \mathcal{C}_r$. In this case, there exists a non-empty credential set $\mathcal{C}''_r$ such that $\mathcal{C}''_r \cap \mathcal{C}'_r = \emptyset$ and $\mathcal{C}''_r \cup \mathcal{C}'_r = \mathcal{C}_r$. In step 4, the target has $\mathcal{C}_{p-r'} = \mathcal{C}_{p-r} \cup \mathcal{C}''_r$ and in step 6, the resulting set $\mathcal{E}_{s-r'} = \rho(\mathcal{C}_{s-r} \cup \mathcal{C}''_r)$. Because $\rho^{-1}(\mathcal{E}_{s-r'}) \cap \mathcal{C}_{p-r'} = \mathcal{C}''_r$, for a credential in $\mathcal{C}''_r$ whose name is c_k, its entry $(Commit_{s_k}(c_k),\ Sig(cred_k),\ exp\_time_k)$ can be found in $\mathcal{E}_{s-r'}$. We have proven that the the zero-knowledge proof protocol is sound in Lemma 2, therefore in step 7, the subject cannot convince the target. The target will terminate the protocol and output $\perp$ with overwhelming probability.

Proof of Privacy-Preservation of Credential Verification Protocol

Now we prove that the subject's privacy is preserved by the protocol. We first need to show that the credential profile is privacy-preserving.

Lemma 4. *Given only the credential profile, the target learns nothing about the credential possessed by the subject.*

Proof. A credential profile contains profile entries. Let's look at the structure of a profile entry. Each entry is a tuple $(Commit_s(c),\ Sig(cred),\ exp\_time)$.

$Sig(cred)$ discloses no information because the signature is generated from the hash value of the credential content. As the hash function is not invertible, no one can learn anything about the credential by looking at the signature. The commitment scheme we use is unconditionally hiding, which means even with unbounded computational power, the possibility of an adversary finding the committed value c is still negligible because for any $c \in \mathbb{Z}_q$ and uniformly randomly chosen $s \in \mathbb{Z}_q$, $Commit_s(c)$ is uniformly distributed in G_q [28]. So by looking at the commitment, one can learn nothing about c. It is impossible to infer the credential from exp_time. Overall, the profile entries leaks no information about the credential.

For the other parts in the profile: an adversary cannot learn any information about the credentials from the signatures, and ID hash and timestamps contain no information about the credentials.

Therefore, given only the credential profile, the target learns nothing about the credentials possessed by the subject.

Theorem 2. *The credential verification protocol is privacy-preserving.*

Proof. The credential profile is publically available in the P2P directory service before entering the protocol, so is available to the target. But by Lemma 4, the target learns nothing about the subject's credentials by looking at the profile.

In steps 1-3, the subject discloses no additional information about its credentials, so the target still knows nothing. If the subject terminates in step 3, then the target knows nothing about the credentials in $\mathcal{C}_s$.

If the subject decides to proceed, then in step 4, the subject discloses $\mathcal{C}_r$ to the target. The target knows all the credentials in $\mathcal{C}_r$, but nothing about the credentials in $\mathcal{C}_{s-r}$. In step 7, the subject needs to run the protocol described in section 3.6 with the target. We have shown that under the Discrete Logarithm Assumption and the Decisional Diffie-Hellman Assumption, that the protocol is zero-knowledge in Lemma 3, so for any credential $c \in \mathcal{C}_{s-r}$, it is computationally infeasible for the target to learn any information other than the fact that $c \notin \mathcal{C}_r$.

Covert Channel Invisibility Theorem

Alexander Grusho[1], Nick Grebnev[1], and Elena Timonina[2]

[1] Moscow State University, GSP-2, Leninskie Gory, Moscow, 119992, Russian Federation
grusho@yandex.ru
[2] Russian State University for the Humanities, 25 Kirovogradskaya, Moscow, 113534, Russian Federation
eltimon@yandex.ru

Abstract. We consider[1] a sequence of finite products of a finite set. A statistical test problem is defined on every product. Consistent sequences of probability measures on these products of the set generate probability measures on the set of infinite sequences. Sufficient conditions of nonexistence for consistent test sequences are proved. These results may be interpreted from the point of view of covert channel secrecy.

Keywords: covert channel, consistent test sequence, secrecy of data hiding.

1 Introduction

There are many definitions of covert channels [3,9]. In this work we chose the following definition. Covert channel allows to hide information transmission from a warden with a given set of resources. This definition deals with only one secrecy parameter which is required for the data hiding. We say that a data hiding scheme is good if

– information transmission can not be detected by the warden with a given set of resources;
– bandwidth of the covert channel can not be done less than the given bound with the help of transformations from the given class. In particular, covert information transmission can not be destroyed with the help of any method from the given class.

Every covert channel is based on an interaction between the sender and the receiver of the covert data.

Unlike the classification of covert channels given in TCSEC [4] and its interpretations we consider three types of covert channels:

– storage covert channels;
– timing covert channels;
– statistical covert channels.

[1] This work was supported by the Russian Foundation for Basic Research, grant 07-01-00484, grant 07-07-00236.

For information transmission in statistical covert channels a modulation of parameters of probability distributions of some random variables or some random processes are used. Probabilistic methods for secrecy analysis of storage channels or timing channels can also be used. For example the Shannon's methodology [13] for secrecy analysis can be applied.

In this work we investigate the concept of a covert channel invisibility for the warden. In earlier works [5,6,8] we used the Shannon's approach to prove an absolute secrecy of some data hiding methods. In that works we discussed the indistinguishability between a legal information transmission and a covert transmission. These and later works have allowed to formulate some common principles for estimation of secrecy (invisibility) of covert channels. There are two principles.

– The first principle maintains that a number of places in which we can hide information should be large. The warden should not be able to analyse all possible places for information hiding.
– The second principle maintains that the place with hidden information cannot be distinguished from the same place when it isn't used for the hidden information transmission.

These principles are difficult to realize. Traditionally, [11] a data hiding method is associated only with one method of embedding of covert information into a container. The simultaneous usage of many data hiding methods is difficult in this case. The first and second principles are tightly bound. If the method of data hiding is known, it is very difficult or almost impossible to eliminate all signs of hidden information. This elimination becomes more difficult when the receiver should be able to reconstruct the hidden information.

On the basis of these principles we can calculate covert channel secrecy (invisibility) as an estimation of labor that is needed for the warden to detect traces of the hidden information transmission. When these principles are used the warden has to analyse traces of hidden information in a lot of places and a lot of false alarms follow from the well known effect in mathematical statistics. The large deviations produce a large number of false places which are detected as places with hidden information. Similar effect appears in the problem of intrusion detection when a number of false detected attacks is so large that true attacks cannot be identified [1,2,7].

In statistical covert channels the unique method of covert channel detection is a statistical method. A set of places for information hiding defines a set of alternatives which the warden should test versus the main hypothesis that there is no hidden information transmission. At the same time the information receiver has an advantage over the warden because he exactly knows the alternative or the set of alternatives used by the sender of message for information transmission.

The main problem for the estimation of secrecy (invisibility) under statistical method usage is uncertainty of probability models used for statistical decision. In contrast to other statistical methods implementations the estimation of secrecy essentially depends on accuracy of definition of random processes which are modulated by hidden information. Inadequacy of such description can result in the simple detectability of the covert channel if the warden has more exact model of the process without embedded hidden information. So the proof of invisibility of a statistical covert channel should be carried out in a wide class of probability models of sender and receiver interaction. Therefore we should make assertions of the impossibility of a detection of hidden transmission in a class of all distributions which satisfy the given conditions.

The moment of the warden's decision about the presence of a hidden information transmission cannot be determined a priori. Such a decision may depend upon various organizational factors. For example, we need a quick decision because the cryptographical key of low length can be transmitted through the covert channel. Or the warden may allow the hidden transmission to last very long. In the case of any covert channel detection the system can be modified to prevent the further interaction between sender and the receiver of the hidden information. Then for the estimation of covert channel secrecy we should consider a sequences of all first parts (from a certain moment) of sender and receiver interaction.

The proof of impossibility of a statistical covert channel detection is a serious problem. First of all it relates to the fact that all statistical procedures give the answer "may be" and almost never give answers "yes" or "no". The second reason consists of the fact that there is no the best (in all senses) statistical decision if we consider composite alternatives.

That is why we come to consider sequences of decision procedures in the conditions of weak determination of probabilistic models of sender and receiver interaction and hidden information transmission. As a result of our analysis we should get answers about "secrecy" or "unsecrecy" of the covert channel. The answer "secrecy" means that the warden has no statistical procedures for the detection of any covert interaction between the sender and the receiver using the observation of such interaction of an arbitrary length. So we need to use the conception of consistent test sequence [10]. The proof of the fact that there is no consistent test sequence means that the warden has no asymptotically good statistical decision rules for detection of covert channels with the arbitrary length of observation of interaction between the parties. If there is such statistical procedure, it particularly means that there is a consistent procedure sequence of decision making.

The purpose of this work is the proof of nonexistence of consistent test sequence under sufficiently general conditions in considered probability models. In section 2 we describe the model and prove sufficient conditions in considered class of models. In section 3 we give an example with the interpretation of the obtained decision for a certain model of covert channels. In conclusion we describe the future work directions in this area.

2 Mathematical Model

As in [7] we consider two computer systems KA and KB, connected by the link S. Let X, $|X| < \infty$, - be a set of all possible messages, which can be sent from KA to KB through S. KA sends to KB the sequence of messages from X as an unidirectional stream in S without feedback. We consider the discrete time and points of time are numbered by the natural numbers $\mathbf{N}$. All statistical problems, solved in view of transmission of a sequence of messages from KA to KB through the link S, are based on data handling during finite time interval. These time intervals may be wide and asymptotical properties of decision functions are analysed. Therefore, we describe the information stream from KA to KB with a random infinite sequence of messages taken from X. Denote a space of messages sequences by

$$X^\infty = \{\alpha = (x_{i_1}, x_{i_2}, ..., x_{i_n}, ...),\ x_{i_n} \in X,\ n = 1, 2, ...\}.$$

Let $(x_{i_1}, x_{i_2}, ..., x_{i_n}) \times X^\infty$, $x_{i_k} \in X$, $n \in N$, be an elementary cylindrical set in X^∞. Let $\mathcal{A}$ be a σ-algebra, which is generated by all cylindrical sets. Assume that probabilities on elementary cylindrical sets

$$\{P_{0,\ 1,...,n}(x_1, ..., x_n)\} \tag{1}$$

generate a consistent family of finite-dimensional distributions. Then the only probability measure P_0 on the measurable space $(X^\infty, \mathcal{A})$ is generated by (1). This measure describes the normal interaction of the system KA with the system KB. Assume that hardware/software agent KA' in computer system KA tries to send secretly a message to hardware/software agent KB' in KB through S. They need a covert channel. We characterize the transmission from KA' to KB' with a consistent family of finite-dimensional distributions generated by probabilities on elementary cylindrical sets

$$\{P_{1,\ 1,...,n}(x_1, ..., x_n)\}. \tag{2}$$

This family determines the only probability measure P_1 on the space $(X^\infty, \mathcal{A})$. To detect a signal from KA' the agent KB' should test a hypothesis $H_0^{(n)} : P_{0,\ n}$, where $P_{0,\ n} = P_{0,\ 1,...,n}$ versus an alternative $H_1^{(n)} : P_{1,\ n}$, where $P_{1,\ n} = P_{1,\ 1,...,n}$. For every n there is the best test (in accordance with lemma Neyman-Pearson).

Assume that a sequence of messages from KA to KB is observed by the warden U. He should solve the problem if the sequence of messages corresponds to the normal behavior of the system or there is a signal from KA' to KB' in this sequence. The warden U knows the distributions $P_{0,\ n}$, but he cannot know distributions $P_{1,\ n}$. In order to describe the wardens's problem let us define a set of the alternatives $H_{11}^{(n)}$ instead of simple alternative $H_1^{(n)}$. The set of alternative is defined by the set of consistent distributions:

$$\{P_{1,\ \theta,\ 1,...,n}(x_1, ..., x_n),\ \theta \in \Theta\}. \tag{3}$$

For all $\theta \in \Theta$ there is the only probability measure $P_{1,\ \theta}$ on the space $(X^\infty, \mathcal{A})$. Then for all $\theta \in \Theta$, $P_{1,\ \theta,\ 1,...,n} = P_{1,\ \theta,\ n}$ and $P_{1,\ \theta} = P_1$ for some $\theta \in \Theta$. The warden's capability to reveal a transmission of a signal from KA' to KB' will be investigated in the terms of asymptotic properties of test sequences for testing $H_0^{(n)}$ versus alternatives $H_{11}^{(n)}$, $n = 1, 2, ...$

Let us define these properties for the test sequences [10]. Denote by $S_n,\ n = 1, 2, ...,$ a sequence of critical sets for tests $T_n,\ n = 1, 2, ...,$ testing $H_0^{(n)}$ versus alternatives $H_{11}^{(n)}$. Corresponding significance levels are denoted $\alpha_n,\ n = 1, 2, ...,$ and $W_n(\theta)$ are power functions of these tests.

Definition 1. Test sequence $T_n,\ n = 1, 2, ...,$ is called consistent iff $\alpha_n \to 0,\ n \to \infty$, for every $\theta \in \Theta\ W_n(\theta) \to 1,\ n \to \infty$,.

Consider a special case for alternatives. Let $\Theta = X^\infty$ and $P_{1,\,x}(\{x\}) = 1,\ x \in X^\infty$. That means that any chosen sequence may be a covert information transmission to KB'.

Theorem. In the considered class of alternatives there is no consistent test sequence for testing $H_0^{(n)}$ versus alternatives $H_{11}^{(n)},\ n = 1, 2, ...,$ for all measures P_0.

Proof. First of all, consider the case when for every $x \in X^\infty :\ P_0(\{x\}) = 0$. Suppose that there exists a consistent test sequence for testing $H_0^{(n)}$ versus alternatives $H_{11}^{(n)}$. Then $S_n,\ n = 1, 2, ...,$ – be the sequence of corresponded critical sets. By the definition of consistency

$$P_{0,\,n}(S_n) \to 0,\ n \to \infty,$$

$$P_{1,\,\theta,\,n}(S_n) \to 1,\ n \to \infty,\ \forall\,\theta \in X^\infty. \tag{4}$$

From the definitions P_0 and $P_{1,\,\theta}$ it follows that

$$P_0(S_n \times X^\infty) \to 0,\ n \to \infty,$$

$$P_{1,\,\theta}(S_n \times X^\infty) \to 1,\ n \to \infty,\ \forall\,\theta \in X^\infty. \tag{5}$$

Lemma 1. For every $x \in X^\infty$ there exists N such, that $\forall n \geq N$:

$$x \in S_n \times X^\infty.$$

Proof. Let there exists $x \in X^\infty$ such, that for every N there exists $n \geq N$ that $x \overline{\in}\ S_n \times X^\infty$. Then we can define a subsequence S_{n_k} such that for all k:

$$x \overline{\in} S_{n_k} \times X^\infty.$$

But then

$$P_{1,\,x}(S_{n_k} \times X^\infty) \to 0,\ k \to \infty,$$

that contradicts (5). Lemma is proved.

Lemma 2

$$\bigcup_{n=1}^{\infty}\bigcap_{k=n}^{\infty} S_k \times X^\infty = \bigcap_{n=1}^{\infty}\bigcup_{k=n}^{\infty} S_k \times X^\infty = X^\infty.$$

Proof. Let $x \in X^\infty$, then from lemma 1 it follows that there exists N such, that $\forall k \geq N$:

$$x \in\ S_k \times X^\infty.$$

Therefore

$$x \in \bigcap_{k=N}^{\infty} S_k \times X^{\infty}$$

and

$$x \in \bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} S_k \times X^{\infty}.$$

Hence

$$\bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} S_k \times X^{\infty} \supseteq X^{\infty}.$$

From lemma 1 it follows that for every N there exists $k \geq N$ such that

$$x \in \ S_k \times X^{\infty}.$$

From here for all n

$$x \in \bigcup_{k=n}^{\infty} S_k \times X^{\infty}.$$

So

$$x \in \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} S_k \times X^{\infty}.$$

Then

$$\bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} S_k \times X^{\infty} \supseteq X^{\infty}.$$

Then it follows that

$$\bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} S_k \times X^{\infty} = \bigcap_{n=1}^{\infty} \bigcup_{k=n}^{\infty} S_k \times X^{\infty} = X^{\infty}.$$

Lemma is proved.

Let's consider the probability

$$P_0(\bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} S_k \times X^{\infty}).$$

Using lemma 2 we get

$$P_0(\bigcup_{n=1}^{\infty} \bigcap_{k=n}^{\infty} S_k \times X^{\infty}) = \lim_{n \to \infty} P_0(\bigcap_{k=n}^{\infty} S_k \times X^{\infty}) = P_0(X^{\infty}) = 1. \tag{6}$$

Since for all n

$$\bigcap_{k=n}^{\infty} S_k \times X^{\infty} \subset S_n \times X^{\infty},$$

then for all n

$$P_0(\bigcap_{k=n}^{\infty} S_k \times X^{\infty}) \leq P_0(S_n \times X^{\infty}).$$

Passing on to limit by n, we carry out

$$\lim_{n\to\infty} P_0(\bigcap_{k=n}^{\infty} S_k \times X^{\infty}) \leq \lim_{n\to\infty} P_0(S_n \times X^{\infty}).$$

But then from (6) it follows that

$$\lim_{n\to\infty} P_0(S_n \times X^{\infty}) \geq 1.$$

This contradicts to (5).

Let's consider the case when there exists $x \in X^{\infty}$ that

$$P_0(\{x\}) > 0.$$

Then for all consistent test sequence with the critical sets $S_1, S_2, ..., S_n, ...$, there exists N, that for all $n \geq N$ it follows that $x \overline{\in} S_n$. If we assume the existence of sequence $n_1, n_2, ..., n_k, ...$, such that $x \in S_{n_k}$ for all k, then

$$P_0(S_{n_k} \times X^{\infty}) \geq P_0(\{x\}) > 0,$$

that contradicts to the condition

$$\lim_{n\to\infty} P_0(S_n) = 0.$$

If $x \overline{\in} S_n$ for all $n \geq N$, then

$$P_{1,\,x}(S_n \times X^{\infty}) = 0,$$

that contradicts to the condition of consistency. Theorem is proved.

Corollary 1. Let the family of probability measures in H_{11}^n includes the family $\{P_{1,\,\theta},\ \theta \in X^{\infty}\}$ such that $\forall \theta \in X^{\infty}$ it follows that $P_{1,\,\theta}(\{\theta\}) = 1$. Then there is no consistent test sequence for testing H_0^n versus H_{11}^n, $n = 1, 2, ...$

Corollary 2. Let for $\forall x \in X^{\infty}$ the probability $P_0(\{x\}) = 0$, and the hypothesis H_{11}^n consists of all measures which are perpendicular to P_0 [12]. Then there is no consistent test sequence for testing hypothesis H_0^n versus H_{11}^n, $n = 1, 2, ...$

3 The Example

Let a sequence of messages from KA to KB be formed outside of agent KA' attainability. Let intervals between messages can be 0 or 1. Suppose that lengths of intervals are independent random equiprobable variables. Their values are distributed with the probabilities $\frac{1}{2}$. Let us add intervals 0 and 1 to the set of possible messages X. The new set of messages denote Y. Then for all measures on the sequences of messages of X the probability of each sequence of messages from the new set Y equals to zero. In this case we consider probability measures only on the sequences of intervals between the messages from X. Furthermore binary sequence of the intervals between messages may be considered as a decomposition of a number from [0; 1] into the binary fraction. In this case the sequence of intervals corresponds a random number between 0 and 1. Then induced distribution on the set of numbers from the segment [0; 1] is a uniform distribution. If the agent KA' can choose the lengthes of the intervals between the messages of X then every chosen sequence of intervals may be considered as a covert message of the agent KA' transmitted to the agent KB'. But it is clear that if the agent KA' has chosen a sequence of only zeros then the warden using the test based on the deviation of relative frequencies of units from $\frac{1}{2}$ gets a consistent test sequence for $H_0^n : P_{0,\,n}$, where $P_{0,\,n}$ is a uniform distribution of intervals between the messages from X, versus alternative $H_1^n : P_{1,\,\overline{0},\,n}$, where $\overline{0}$ is a null sequence of such intervals.

While considering this example an illusion of contradiction with the proved theorem can appear. However this is not the case. The theorem asserts that there is no consistent test sequence versus all alternatives which are described by a singular distribution on every element of considered space. This doesn't mean that there is no consistent test sequence versus the part of such alternatives, for example, versus the given fixed alternative.

Then it follows that not all alternatives influence equally on impossibility of consistent test sequence construction. Therefore the arbitrary sequence which is chosen as a covert signal is not always convenient even if there is no consistent test sequence in the given class of alternatives. The absence of consistent test sequence means that for all methods of testing there exists a sequence of messages produced in accordance with the measure $P_{0,\,n}$, but the warden has no assurance that he is not swindled.

Then there is a question about the difference between the two situations. The first situation produces the warden's uncertainty in the case of existence of consistent test sequence without knowledge of n when the power of test is sufficiently closed to 1. The second situation is the case when there is some alternative for which the power of test does not tend to 1 at all. These two cases have the principal difference. In the first case if the consistent test sequence exists then there exists the constructive procedure to choose the sequence of such critical sets S_n for which the power of test will sooner or later become sufficiently close to 1 for every alternative. In the second case such sequence of critical sets doesn't exist in principle. It means that in the first case the warden should try to find consistent test sequence for covert channel detection but in the second case he has no reason to do this. At the same time in the example for any fixed alternative under the theorem condition such consistent test sequence can be constructed.

Let's stand in the position of an adversary and try to use these results for construction of a covert channel. For this purpose the adversary must use the same reasonings as in the warden's determination of a covert transmission.

In our example there exists a lot of sequences that can not be separated statistically from equiprobable sequences. All such sequences are acceptable to be chosen as a signal transmitted through the covert channel.

One of the methods of constructing such an invisible sequence is the selection of the same intervals between the messages as were used for the legal message transmitting from KA to KB. Then the warden who does no remember such sequences can not detect this signal because all known to him differences don't work in this case.

4 Conclusion

Provable secrecy of a statistical covert channel follows from nonexistence of consistent test sequences. That is why we investigate sufficient conditions for nonexistence of consistent test sequences. The result should not depend on certain probability distributions because the warden can know the model of normal behavior better than an adversary. The better knowledge of the model follows the better statistical decision.

In the space of infinite sequences we have found such sufficient conditions. These conditions can be checked on practice because very often it is possible to prove that the probability of every possible sequence equals to zero.

If we suppose that every sequence can be chosen as a signal then the warden has to possess a sequence of tests for detection of any covert signal. An adversary can make a mistake if he chooses a covert signal randomly, for example, as a point distributed uniformly in the segment $[0; 1]$.

The previous example shows that in the case of absence of consistent test sequence we should not use all possible alternatives as covert signals. There should exist a certain infinite set of alternatives that every infinite subset of this set cannot be separated by a consistent test sequence from $P_{0,\,n}$, $n = 1, 2, \ldots$. A signal chosen from this set cannot compromise the covert channel. The problem is to define such a set.

There are more problems that are to be solved during the work on this theory. As we defined before the secrecy of covert channel can be estimated with as a number of places for information hiding. It is very interesting to research a connection between the increasing the number of places for data hiding and the properties of probability measures P_0 and $P_{1,\,\theta}$, $\theta \in \Theta$. We suppose, that the high increase of the number of places for hiding follows that a lot of sequences have probability equals to zero.

It is interesting to find sufficient conditions of nonexistence of consistent test sequences in terms of finite dimensional distributions.

The warden can use many tests for analysis of the sequence of messages. The problem is to prove that he cannot increase his ability to detect a covert channel in the case when he uses many tests and there is no consistent test sequence.

We have considered the problem of covert channel invisibility when its bandwidth is minimal. But the conditions of invisibility can be different if the bandwidth of the covert channel should be more then a predefined limit.

References

1. Axelson, S.: The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. In: Proc. of the 6th Conference on Computer and Communications Security, November 1999 (1999)
2. Bainov, D., Grusho, A., Timonina, E., Volkovich, V.: On a Pobabilistic Model of Intrusion Detection. Int. J. of Pure and Appl. Math. 39(1) (2007)
3. Covert channels through the looking glass (October 2005), `http://www.gray-world.net`
4. Department of Defense Trusted Computer System Evaluation Criteria. DoD (1985)
5. Grusho, A.: Subliminal channels and information security in computer systems. Descrete Mathematics and Applications 8(2), 127–134 (1998)
6. Grusho, A.: On existence of subliminal channels. Descrete Mathematics and Applications. 9(1), 1–8 (1999)
7. Grusho, A., Kniazev, A., Timonina, E.: Detection of Illegal Information Flow. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 235–244. Springer, Heidelberg (2005)
8. Grusho, A.A., Timonina, E.E.: Some relations between discrete statistical problems and properties of probability measures on topological spaces. Descrete Mathematics and Applications. 16(6), 547–554 (2006)
9. A Guide to Understanding Covert Channel Analysis of Trusted Systems. National Computer Security Center. – NCSC-TG-030. Ver. 1 (1993)
10. Lehmann, E.L.: Testing Statistical Hypotheses (Springer Texts in Statistics), 2nd edn. p. 600. Springer, Heidelberg (1997)
11. Wu, M., Liu, B.: Multimedia data hiding, p. 219. Springer, New York (2003)
12. Prokhorov, U.V, Rozanov, U.A.: Theory of probabilities. Moscow. Science (in Russian) (1973)
13. Shannon, C.: Works on Theory of Informatics and Cybernetics. Moscow. Foreign Literature, 830 (in Russian) (1963)

Policy-Based Proactive Monitoring of Security Policy Performance

Vitaly Bogdanov and Igor Kotenko

Computer Security Research Group, St. Petersburg Institute for Informatics and Automation
39, 14 Liniya, St.-Petersburg, 199178, Russia
{bogdanov,kotenko}@comsec.spb.ru

Abstract. One of topical tasks of policy-based security management is checking that the security policy stated in organization corresponds to its implementation in the computer network. The paper considers the suggested approach to proactive monitoring of security policy performance and security mechanisms functioning. This approach is based on the different strategies of automatic imitation of possible users' actions in the computer network, including exhaustive search, express-analysis and generating the optimized test sequences. It is applicable to different security policies (authentication, authorization, filtering, communication channel protection, etc.). The paper describes stages, generalized algorithms and main peculiarities of the suggested approach and formal methods used to fulfill the test sequence optimization. We consider the generalized architecture of the proactive monitoring system "Proactive security scanner" (PSC) developed, its implementation and an example of policy testing.

Keywords: Security policy, monitoring, test sequence optimization.

1 Introduction

On the computer network exploitation stage the security administrators need to check the correspondence of the security policy formulated on the design stage to its implementation in the computer network. This task is equivalent to the monitoring of correct performance of security tools and services deployed.

To monitor the functioning of security tools and services on the exploitation stage *a passive approach* is being used as a rule. According to this approach the current system configuration is being periodically compared with the configuration that was installed (by analyzing the settings of operating systems and applications). The differences found say that security policy does not hold true. However such approach can not guarantee correct security policy fulfillment. The complex distributed network system requires a huge quantity of such settings. Not all of them can be under control. The malefactors can bypass given settings or change them dynamically by injecting and using particular applications. As a result the settings of software and hardware may differ from the security policy accepted.

The paper proposes *a common approach to the proactive monitoring of security mechanisms*. The approach suggested consists in modeling and imitating user actions in the network and evaluating the results of these actions. When using this approach the administrator acts mainly as observer: he takes a part in the system administration only when the serious violation of the security policy is revealed or its participation is needed to make an important decision. The suggested approach to proactive monitoring is similar to active vulnerability analysis or penetration testing. Their difference is that during proactive monitoring we do not apply exploits in the tested network. All user actions are usual actions and their fulfillment is controlled by the security policy.

The paper is structured as follows. *Section* 2 analyzes relevant works and describes the essence of the suggested approach. The common monitoring technique and algorithms applied for different security policies are represented in *section* 3. *Section 4* considers the approach features and restrictions as well as formal methods used for test sequence optimization. *Section* 5 describes the architecture and implementation of the proactive monitoring system developed, as well as an example of policy testing. *Conclusion* looks round the paper results and further research directions.

2 Related Work and the Approach Suggested

Existing works on security policy, in particular on security policy monitoring, form the necessary basis for the paper. Russel and Gangemi [21] affirm that one of the necessary conditions of system security is users' actions monitoring and checking the correspondence of actions to the security policy. Marriot and Sloman [17] consider the issues of monitoring the network events and creating the policies to react on events. In [3, 6, 15, 20, 22, 25] the necessity of security policy monitoring is also noted. Carney and Loe [7] suggest using monitoring to select a stronger policy when the current is violated. Chosh et al. [12] consider active tries to violate security policy as possibility to form a security policy for vulnerable application. Gama and Ferreira [11] suggest the methods of policy detailed elaboration to monitor events in information systems. Beigi et al. [4] use network configuration checking to confirm a policy correspondence. Agrawal et al. [2] consider a security policy monitoring system that checks the correctness of configuration changes. Strembeck [24] describes connection between policy rules, user behavior scenarios and user aims.

The proactive monitoring works of Sailer et al. [23], El-Atawy et al. [9] and Wheeler [26] are the closest to our paper.

Sailer et al. [16] use the proactive monitoring approach for testing the IPSec protocol. Authors consider a simple IPSec based VPN realization and all its possible violations. To check that network hosts are configured properly the authors suggest to place on the one host of VPN-connection a network packet generator and on the second - the tool to capture and check the incoming packets. The generator forms ICMP packets and sends them to the host on the opposite connection point. The packet capturing and checking tool captures the packets before they are processed by the network protocol stack and checks if the packets were formed (secured) correctly. Such scheme allows to quickly reveal the incorrect IPSec configurations.

El-Atawy et al. [9] analyze two methods of the proactive firewall testing. The naive approach to testing is based on sending the random packets to the firewall. The second approach is more effective from the point of view of network resource usage. The specification of each rule contains the description of packets sets that this rule affects.

One can determine all possible intersections of such sets. It is suggested to analyze the result of sending one packet for each such intersection.

Wheeler [26] describes a distributed proactive system for firewall testing. He suggests a formal language for specifying the filtering rules and the architecture of the testing system. This system contains three types of modules (Prober, Manager and Wizard) that are connected in a tree-like structure. Probers are tree leafs, Wizard is the tree root and managers are between the root and leafs. Tester is the simplest module. It listens to some network port or sends packets to another tester. Manager controls testers and managers in its sub tree. It sends commands to subordinate modules and transmits answers upward on the tree-like structure. Wizard interprets the policy description, creates a testing plan, sends commands to subordinate managers, receives and analyses testing results and provides the user interface. Two testers are used to test one connection; they try to create connections using TCP or UDP protocols.

A lot of different *commercial systems* are being used to manage security policies nowadays. Although all these tools include the components of audit and monitoring of security mechanisms, they mainly targeted on the integration and correlation of security events and an integrated reporting about defended system state.

The approach to security policy monitoring suggested in the paper is based on active imitation of users' actions (as permitted as well as prohibited by the security policy) and on determining the differences between actual system reactions and reactions that are corresponds to the security policy. In existent works this approach was not researched enough. In difference from most relevant papers [9, 23, 26], where monitoring is used to check IPSec and filtration rules, in this paper we suggest the common approach to proactive monitoring of security policy which can be used for different policies (authentication, authorization, filtering, channel protection, etc.). According to the approach suggested the tested system behavior that is initiated by the generated actions is being compared with the behavior of *standard system* [5]. As a standard a formal model of the security policy is used. This model is built on basis of the security policy rules and network configuration described on special specification languages.

3 Main Stages and Techniques

To test the security policy of the computer network completely one need to model and plan all possible sequences of users' actions, perform these actions and compare network responses with expected results. Such testing can not be performed manually; furthermore it may take too much time. To make this task be performable in practice it is needed to systematize the testing procedure, introduce some restrictions for the monitoring process and suggest mechanisms that can make checking more effective.

The suggested approach is based on the independent checking of different rule categories. The every category (or policy) - authentication, authorization, filtering, channel protection, etc. - is implemented using different security facilities. Filtering policy is implemented in the network using firewalls, but authentication, authorization and channel protection policies - as network services.

We suppose that all these facilities do not change their state after they perform a user's action, i.e. the sequence of actions does not affect their results. In this condition, it is sufficient to check only results of performing each action separately. Such task statement reduces the task complexity and testing time.

To form a list of all possible actions for each policy category, the proactive monitoring system needs information about the policy and the network configuration. This information is input for monitoring. After receiving this information the monitoring system builds the models of the security policy and the network — this is *the first stage* of the system operation. On *the second stage* the system generates the sequence of test impacts. This sequence can be formed by various ways for different policy categories. *The third stage* is optimizing the sequence of test impacts. *The fourth stage* consists in performing the test impacts in the network and receiving the results of impacts. *The fifth (the last) stage* is generalizing the monitoring results and forming the report. The results of each action should be analyzed taking into account the policy category of the action.

To check the security policy rules completely, it is needed to check success (or failure) for all user operations on all assets with all user system accounts. Such approach (based on *exhaustive search*) is a very labor-consuming, and does not always allow check up the policy for an adjusted time. There are other approaches that have no lacks of exhaustive search. They can approve with a certain probability that policy holds true. For example, the *"express-analysis) approach* checks only for one random object from the class. Such approach implies essential decrease of checked operations. When the rule uses an object of some class, we check up rule only for one random object from this class. In this case we can have a high false negative rate of deviations. *The other approach* verifies only several objects of the class. The quantity of such representatives depends on quantity of objects in the given class. Thus, *exhaustive search* is the most laborious. The next is the choice of several random representatives. And the least laborious approach is the choice of one random representative.

The generalized algorithm of authorization policy checking has the following operations: (1) Look through all users; (2) Look through all privileges; (3) Look through all scanners; (4) Check up the presence of the given privilege for the given user from the given scanner. There are four possible check outcomes: (1) If there is an operation permissive rule and the operation was executed successfully then *the policy is not broken*; (2) If there is an operation permissive rule and the operation was executed with failure then *the policy is broken*; (3) If there is no operation permissive rule and the operation was executed with failure then *the policy is not broken*; (4) If there is no operation permissive rule and the operation was executed successfully then *the policy is broken*. The fourth violation is the most serious violation. In this case the user can execute operation which is actually forbidden by the policy. Another type of violation is the second violation (user can not execute operation permitted by the policy).

The generalized algorithm of authentication policy checking includes the following operations: (1) Look up through all users; (2) Look up through all credentials; (3) Look up through all services; (4) Look up through all operations; (5) Check up the success of the given operation by the given user on the given server without authentication; (6) Check up the success of the given operation by the given user on the given server with the given authentication method.

The generalized algorithm of filtering policy checking contains the following operations: (1) The checked connections set is empty; (2) Look up through all firewall rules; (3) Assign connections set from current rule to the connection set for checking;

(4) Subtract checked connections set from the connections set for checking; (5) Get the random connection from the set of the connections for checking; (6) Look up for the scanner to send packets through firewall; (7) Look up for the scanner or service to receive packets beyond firewall; (8) Check up the possibility of connection establishing between sending scanner and receiving scanner or service; (9) Add connections for checking set to the checked connections set.

The generalized algorithm of channel protection checking has the following operations: (1) Look up through all servers and ports; (2) Look up through all channel protection techniques; (3) Check up the possibility of the connection to the given host without channel protection; (4) To check up the possibility of the connection to the given port of the given host using the given channel protection technique. Note that if a channel protection policy rule uses transport level channel protection protocols (such as IPSec), an application can not choose whether it uses channel protection or not, because the channel protection using depends on the system configuration. To check the protection of transmitted data on the transport level we need to use the following algorithm: (1) Look up through all servers; (2) Look up through all target hosts and ports; (3) Look up through all channel protection techniques; (4) Check up the possibility of connection to the given server on the given port; (5) Check up if the channel protection on transport level was used.

4 Restrictions and Optimization Approaches

There are several important *aspects of the security policy monitoring*. Let us consider three such aspects:

(1) performing the “dangerous operations”,
(2) possibility of conflicts with working users,
(3) the necessity to use an authentication database.

The *dangerous operations* are operations that can lead to information integrity violations. For example, for file system such operations are removing or changing a file. To prevent integrity violations as a result of such operations one need take care about the backing up the information before performing the changes. The other way to bypass the dangerous operations is introducing several modes of system operation. One of the modes allows complete testing with dangerous operations included. The other mode allows all operation testing except dangerous. Dangerous operations are most laborious because they require performing the additional actions.

The second important aspect of the proactive monitoring is *a possibility of conflicts with other information processing operations and users*. Such conflicts may arise as a result of almost any operation both dangerous and safe from the integrity point of view. For instance, reading access to a file can be denied because this file is locked for editing by other user. A possible solution of this problem is introducing a special time to check the network when users do not work or work rarely. The disadvantage of this solution is impossibility to check the rules that have time restrictions. The other solution is to try to determine the conflict reason and use it during checking.

The third important aspect is *a need to have a database of authentication data* for all users. This aspect is stipulated by the fact that before checking the user’s privileges for an operation on an object it is necessary to be authenticated in the system. This aspect undoubtedly reduces the applicability of the proactive monitoring.

The proactive monitoring is a very resource-intensive task due to both a large input data variety and a long time needed to receive response from the network. To decrease the testing time we need *to optimize the test impacts.*

We can optimize the test impacts by different ways: (1) remove the superfluous test impacts; (2) find the optimal impacts subsequence; (3) generate the impacts subsequences that can be performed simultaneously.

Due the limits of the paper, let us consider the procedure of *test impacts optimization on an example of filtering policy testing.*

To describe optimization methods we need to give a formal description of filtering policy. The total network filtering policy is a set of policies for all firewalls in the network. The methods of removing the superfluous test impacts and optimizing the test impacts sequence that are described below should be applied to each firewall policy.

The filtering policy of each firewall represents a set of filtering rules. Every *firewall rule* $R_i = < P_i, A_i >$ determines a set P_i of network packets, that rule affects, and an action $A_i \in \{allow, deny\}$ that has to be applied to the packets from P_i. Furthermore it is essential to take into account the ordering of rules because the packets from P_i may intersect for different rules.

Let us denote the *firewall policy* as an ordered set of the firewall rules by $FP_f = \{R_i\}_{i=1}^{N_f}$, where N_f - the quantity of rules in the policy of the firewall f .

The *test impact* for the filtering policy represents sending the network packet over the firewall.

To test the filtering policy completely we need to send all possible packets over the firewall and compare that they were passed in compliance with the policy. Such testing will take a long time. We can send one packet for every rule, but this is not a good decision because the packets P_i of the rules may intersect. In such case several rules affect the results of packet passing.

To remove superfluous impacts it is possible to use the approach suggested El-Atawy et al. [9]. The main idea of the approach named *policy segmentation* is to break the packets sets $\{P_i\}_{i=1}^{N_f}$ for all rules on their disjunctive subsets $S = \{S_j\}_{j=1}^{M}$ of so-called segments. The *segment* is such subset of the packets sets from one or several rules that all packets from the subset are related to the same rules set. Authors show that it is enough to send over firewall one packet from each segment to test all possible combinations of rules.

The sending of only one packet from each segment reduces the number of test impacts. But there is a *possibility of additional optimization by selecting the optimal sequence of test impacts.*

Let we have two rules R_1 and R_2 such that $P_{12} = P_1 \cap P_2 \neq \varnothing$ and $A_1 = allow, A_2 = deny$. We sent packet $p_{12} \in P_{12}$ first and revealed that it does not arrive at destination point. Consequently the rule R_1 does not hold and it is not needed to send the packet $p_1 \in P_1 \setminus P_2$. In other side, if we send the packet p_1 first we will need to send packet p_{12} in any case to verify the rule R_2.

The goal of the impacts sequence optimization is to create such packets sequence that will reveal all deviations from security policy by a minimal number of steps.

Let us use *binary test questionnaire optimization theory* described in [1] and define the set of events as a set of binary vectors $E = \{ \{e_i\}_{i=1}^{N_f} \}$.

Let $e_i = 1$ if a firewall rule works good and $e_i = 0$ if the rule does not work. The *test impact* t_j is one packet sending from a segment $S_j \in S$. Each test impact may be fulfilled successfully (the packet was received at the destination point) or unsuccessfully (the packet was filtered by the firewall).

Depending on the result, the test impact divides the set E of events on two classes: E_{t_j} - the events corresponding to the successful test impact t_j and $\overline{E}_{t_j}$ - the events corresponding to the unsuccessful test impact t_j.

To identify events from the set E we can use different binary test questionnaires.

The *questionnaire* is a set T of questions and the sequence of their asking. The binary questionnaire can be represented as a binary tree, where the nodes of the tree are the subsets of the set E and every node $X \subset E$ has sons X_t and $\overline{X_t}$. The leafs of the tree are nodes $X \subset E$, for which X_t or $\overline{X_t}$ is empty set.

Performing the test actions and moving at the tree we will come into list that contains the set of the events that take a place at the tested system. When we find optimal test questionnaire for the given set of the events and test actions, we find optimal sequence of the actions performing.

It is necessary to define the *optimality criteria of the test questionnaire*. Let us define such criteria as minimum of the average count of the test actions need to be performed to identify the event from the E unambiguously:

$$C = \sum_{i=1}^{N_f} p(y_i) C(y_i),$$

where $y_i \in E$ - an event, corresponding to the firewall state, $p(y_i)$ - the probability of the event y_i, $C(y_i)$ - the cost of the y_i identification.

The cost of event identification is defined as follows:

$$C(y_i) = \sum_{t_k \in \mu(y_i)} c(t_k),$$

where $\mu(y_i)$ - a path from the root of the questionnaire to the event y_i, i.e. a set that contains test actions that need to be performed to identify the event y_i; $c(t_k)$ - the cost of the question t_k.

Let the costs of every question is 1, then $C(y_i)$ will be as follows:

$$C(y_i) = |\mu(y_i)|.$$

Let also all events have equal probability then the average cost of the questionnaire will have the following form:

$$C = \frac{1}{N_f}\sum_{i=1}^{N_f} |\mu(y_i)| \cdot$$

So we can obtain binary test questionnaire where test impacts correspond to questions. If we optimize this questionnaire we will obtain optimal test impacts sequence.

Let us use *dynamic programming method for the optimization of binary test questionnaires with the incomplete set of questions.*

To describe this method we need to define the concept of situation. Let for the set E of events the set T of questions is defined. Then E_T is a set of subsets L_α on which the sequences of questions from T divide E. For each $L_\alpha \subset E_T$ there are a subset of questions T_α that have a sense relative to L_α, i.e. these questions divide L_α on two nonempty subsets. The pair (L_α, T_α) is named as a *situation*, $m_\alpha = |L_\alpha|$ is a *situation degree.*

Let for each possible situation with degree m_α the corresponding optimal sub questionnaire has been built. Then, using these sub questionnaires, we can build an optimal sub questionnaire for the situation (L_β, T_β) with the degree $m_\beta > m_\alpha$.

The Bellman's optimality equation for this case is as follows:

$$C_{opt}(L_\beta, T_\beta) = \min_{t \in T_\beta}\left\{ c(t) + \sum_{n=1}^{2} p_n C_{opt}(L_{\beta_n}, T_{\beta_n}) \right\},$$

$$p_n = \sum_{y \in L_{\beta_n}} \left(p(y) \Big/ \sum_{y' \in L_\beta} p(y') \right)$$

where $c(t)$ is the cost of the question t, $p(y)$ is the probability of the event y, p_n is the conditional probability, L_{β_n} are the subsets on which question t divides the set L_β ($n = 1, 2$). Since the costs of questions are equal, we can set them as 1. We will also consider that the probabilities of all events are equal, then the equation will be as follows:

$$C_{opt}(L_\beta, T_\beta) = \min_{t \in T_\beta}\left\{ 1 + \sum_{n=1}^{2} \frac{|L_{\beta_n}|}{|L_\beta|} C_{opt}(L_{\beta_n}, T_{\beta_n}) \right\}.$$

This equation is the base of the dynamic programming algorithm [1].

The dynamic programming algorithm for the test impacts optimization is as follows:

1. Set $m_\beta = 1$.
2. For each possible situation (L_β, T_β) with degree m_β build optimal sub questionnaire according to Bellman's equation using sub questionnaires built on the previous steps for situations with degree $m < m_\beta$.
3. If $m_\beta \leq N_f$ then increase m_β on 1 and return to step 2, in other case the optimal questionnaire has been built.

The dynamic programming method allows finding the optimal test packet subsequence for firewall testing. The disadvantage of the dynamic programming method is exponential complexity of the algorithm.

Thus we defined test impacts optimization ways to test filtering policy of the single firewall. To optimize complete network filtering policy we need to find parallel test impacts sequences. Obviously each firewall can be tested independently and different firewalls testing results do not affect each other [26].

To illustrate the presented method let us consider a simple network with a boundary host (see figure 1).

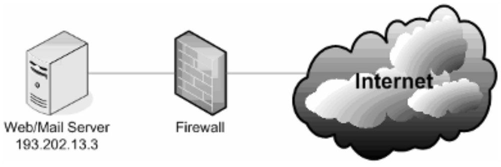


Fig. 1. The test network with a boundary host

Let us consider the filtering policy of the firewall that consists of three rules: (1) allow TCP connections with the host 193.202.13.3 at the port 80; (2) allow TCP connections with the host 193.202.13.3 at the port 110; (3) deny all other connections.

Every rule allows the particular type of packets, and the last rule denies all packets that were not referred. This rules creation method is typical for the firewall policies.

Let us consider how the test actions optimization algorithm works on the given example. We will note every state of the tested firewall by three letters, for example AbC. The capital letter means that rule works correctly in this state and small letter means that rule does not work. For example, at the state AbC the A and C rules work and B rule does not work.

The set E of the all possible states looks as follows: E = { ABC, ABc, AbC, Abc, aBC, aBc, abC, abc }. The set T of the possible test actions contains three elements: T = { Pa, Pb, Pc }, where Px corresponds to connection establishing that is processed by the rule X. For instance Pa is the establishing of TCP connection with the host 193.202.13.3 at the port 80 and Pc is the establishing of TCP connection with the host 193.202.13.3 at the port different from 80 and 110.

The set of the subsets that are results of partitioning the E by the questions from T looks as follows: E_T = { E, { ABC, ABc, AbC, Abc, aBc, abc }, { ABC, ABc, Abc, aBC, aBc, abc }, { ABC, ABc, Abc, aBc, abc }, { ABc, Abc, aBc, abc }, { ABC,

AbC, aBC, abC }, { aBC, abC }, { ABC, abC }, { ABC, AbC }, { ABC, aBC }, { abC }, { AbC }, {aBC }, { ABC } }. The test actions divide the set E into the situations of different degrees.

Let us consider all situations from the first degree situation up to the eight degree situation and create optimal test plan. *First degree situations*: ({ abC } , { }), ({ AbC } , { }), ({ aBC } , { }), ({ ABC } , { }). Because the test actions for all these situations are empty then C_{opt} for each first degree situation is equal to 0. *Second degree situations*: ({ aBC, abC }, { Pb }), ({ ABC, abC }, { Pa }), ({ ABC, AbC }, { Pb }), ({ ABC, aBC }, { Pa }). For every situation there is only one test action so for all situations $C_{opt} = c(t)$ =1. *Fourth degree situations*: ({ ABc, Abc, aBc, abc }, { }) и ({ ABC, AbC, aBC, abC }, { Pa, Pb }). For the first situation the test actions set is empty, consequently C_{opt} = 0. Second situation allows two test actions, consequently C_{opt} = min (1 + 2/4 + 2/4, 1 + 2/4 + 2/4) = 2. The optimal test action is any from two. Let us assign Pa as optimal test action. *Fifth degree situations*: ({ ABC, ABc, Abc, aBc, abc }, { Pc }). C_{opt} = 1 + 1/5 * 0 + 4/5 * 0 = 1. There are two *sixth degree situations* and each allows two test actions: ({ ABC, ABc, AbC, Abc, aBc, abc }, { Pb, Pc }) and ({ ABC, ABc, Abc, aBC, aBc, abc }, { Pa, Pc }). Let us consider C_{opt} for the first situation. C_{opt} = min (1 + 1/6 * 0 + 5/6 * 1, 1 + 2/6 * 1 + 4/6 * 0) = 1,33 and optimal test action is Pc. Similarly for the second situation C_{opt} = 1,33 and optimal test action — Pc. Let us assign the first situation as optimal and the optimal test action - as Pc. There is only one *eighth degree situation*: (E, { Pa, Pb, Pc }). C_{opt} = min (1 + 6/8 * 8/6 + 2/8 * 1, 1 + 6/8 * 8/6 + 2/8 * 1, 1 + 4/8 * 0 + 4/8 * 2) = min (2 1/4, 2 1/4, 2) = 2. The optimal test action is Pc. The optimal test tree that was created in accordance with the given approach is shown in figure 2.

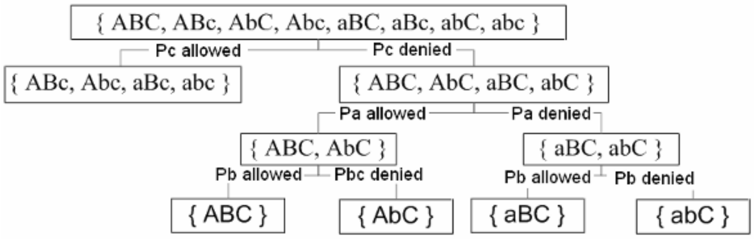


Fig. 2. Optimal test tree

The average cost of the questionnaire that corresponds to the tree is C = 1/8 * (1 * 4 + 3 * 4) = 2. Let us try to calculate *the difference between optimal and non-optimal average number of test operations*. Our policy will be typical, i.e. will contain $N_f - 1$ permissive rules and one prohibitive rule like in the example above. Then the average cost of the optimal tree can be calculated as follows:

$$C_{opt} = \frac{1}{2^{N_f}}(N_f 2^{N_f - 1} + 2^{N_f - 1})$$

and the average cost of the non-optimal tree will be:

$$C_{nopt} = \frac{1}{2^{N_f}}(N_f(2^{N_f-1}+1)+(N_f-1)(2^{N_f-1}-1)).$$

Then the difference between optimal and non-optimal average costs will be:

$$C_{nopt} - C_{opt} = \frac{1}{2^{N_f}}((N_f-2)2^{N_f-1}+1) = \frac{N_f-2}{2} + \frac{1}{2^{N_f}}.$$

When $N_f = 22$ the difference will be $C_{nopt} - C_{opt} = \frac{20}{2} + \frac{1}{2^{22}} \approx 10$ operations.

5 System Architecture, Implementation and Experiments

The generalized architecture of "Proactive security scanner" system (PSC) that implements the suggested approach is depicted in figure 3. PSC consists of configurator, scanners, correlator and management console.

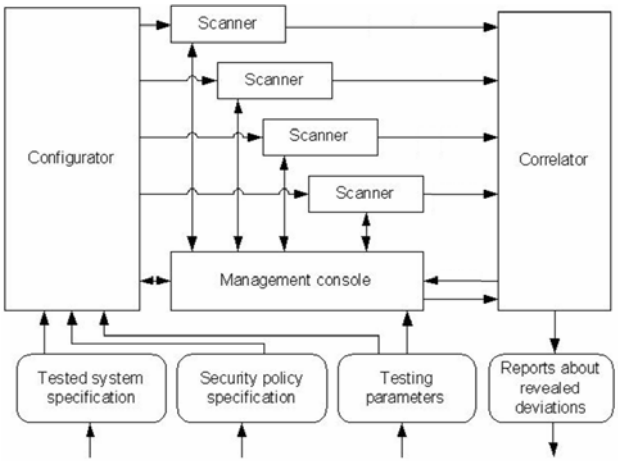

Configurator
Scanner
Scanner
Scanner
Scanner
Correlator
Management console
Tested system specification
Security policy specification
Testing parameters
Reports about revealed deviations

Fig. 3. Generalized architecture of PSC

The PSC input is the tested system specification, the tested security policy specification and testing parameters (to manage the process of testing). *The PSC output* is the report about revealed deviations from the specified security policy in the tested system and their estimation (to answer to the question how critical or crucial these deviations are). The PSC should satisfy the following *main requirements*: tested system and security policy coverage (completeness), deviations estimation fidelity (adequacy), and productivity.

PSC allows revealing the following security policy violations: Authorization policy - the impossibility of authorized operation, the possibility of forbidden operation;

Authentication policy - the possibility of operation performing by unauthenticated user, the impossibility of operation performing by authenticated user; Filtering policy - permitted connection blocking, forbidden connection possibility; Channel protection policy - unprotected channel possibility, the lack of channel protection.

Configurator (*PSC Config*) plans the test impacts sequence to monitor the security policy performance by forming the tasks for scanners (see figure 4). Its input consists of the specifications of the checked security policy in System Description Language (SDL) and the tested network in Security Policy Language (SPL) as well as the values of testing parameters. The testing parameters can be a subset of security rules to check, a part of the network to test, subjects and objects to test, information about scanners locations in the system, internal information for scanner task generation, etc.

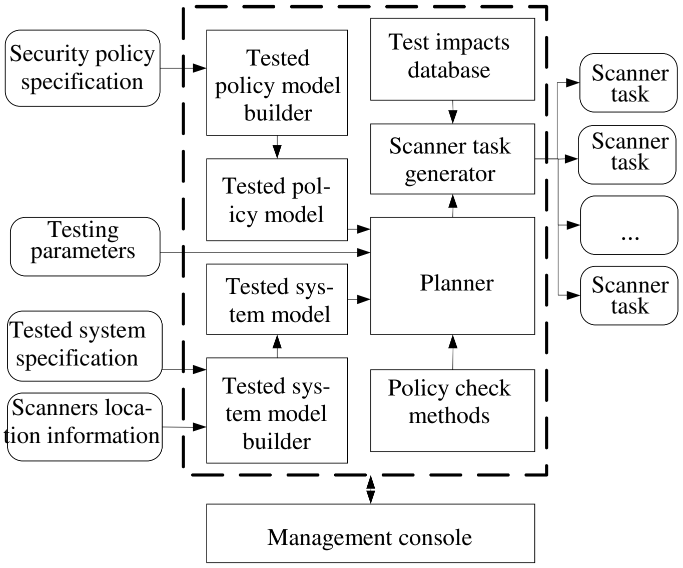


Fig. 4. Generalized architecture of configurator

Security policy description language (SPL) [19] is based on the CIM standard [8] and XML technologies. SPL allows setting the grouping, priority and classification of the policy rules and categories. It contains *five rules categories*: authentication, authorization (access control), filtering, channel protection (IPSec, SSL/TLS) and operational. Syntax of each rule category is determined by a particular scheme that is based on the XML-scheme. SPL schemes were derived from xCIM-schemes that are the representations of CIM using XML-scheme technology.

The system description language (SDL) [19] is applied to describe the network configuration and its functionality. Policy rules must use only such devices, services and functions that are contained in the SDL network description. SDL is based on XML scheme and contains the following main parts: network topology description, i.e. how network nodes are connected, connection type (wired, wireless), cable type, etc; network services description, that are available on corresponding servers and also operating systems installed; devices and interfaces description, i.e. names, addresses, ports, common purpose description, vendor, etc.

Configurator creates the tasks for scanners by processing input data. For example to check the firewall, configurator forms two tasks: (1) to send a packet from IP address before firewall and (2) to receive the packet at IP beyond firewall.

The *generalized algorithm of configurator* includes the following steps: (1) Receive the input data; (2) Receive information about scanners locations; (3) Construct the tested system model according to the system specification; (4) Designate scanners locations in the model of the tested system; (5) Construct the tested policy model according to the policy specification; (6) Look through all security policy rules; (7) Select appropriate rule checking method according to kind of the rule and testing parameters; (8) Generate the tasks for the given scanners; (9) Transfer generated tasks to scanners for processing. The details of the algorithm realization differ for each policy class check.

Scanners (PSCs) are components which check a part of the security policy in the tested sub-network. The policy part and system fragment are set by configurator. A scanner task is transferred from configurator to scanner and contains test impact instance. Scanner checks the policy by performing impact and sends check results to correlator.

The *generalized algorithm of scanner*: (1) Execute impact instance run method; (2) If there are deviations from the rule, add check results and deviations conditions at the scanner report; (3) Transfer check results to correlator.

Correlator (PSC Correlation) receives the check results, analyses them and forms summary reports about the security policy violations revealed in the network.

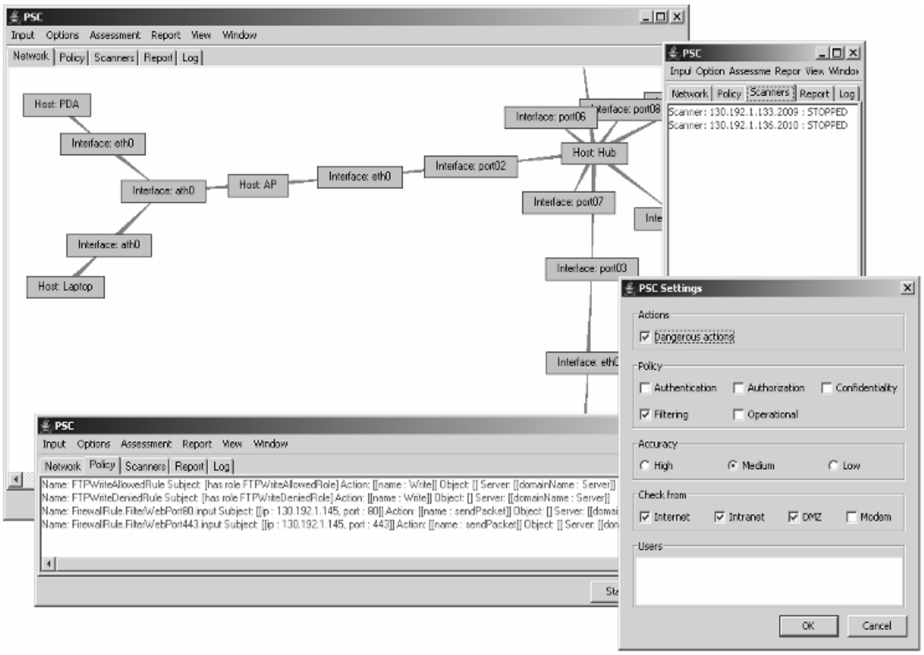
The *generalized algorithm of correlator*: (1) Receive check results from scanners; (2) Process and generalize check results to form reports about revealed deviations; (3) Generate and send reports to major system.

Management console allows the security administrator to manage all system components, set input data to configurator and look through the reports of correlator.

To simplify configuring and information transferring, configurator, correlator and management console can be incorporated in one module. To effectively monitor the network at least one scanner should be located at each network segment. In addition, for example, there should be a scanner on a workstation outside the network and a scanner on a workstation which is connected to the analyzed network over modem. If the wireless network is used we need a scanner on the wireless network host.

The main screen of the PSC management console (PSC window) is shown in figure 5. Tabs panel on the main user interface screen allows examining the information that is concerned with current network configuration (Network tab), security policy (Policy tab), scanners' states (Scanners tab), revealed deviations from the security policy (Report tab) and also system working log (Log tab).

In figure 5 (at right-bottom corner) the checking parameters window (PSC Settings window) is showed. Using these parameters one can determine the dangerous operations performed (Actions field), what policy categories will be tested (Policy field), what will be testing accuracy (Accuracy field) and locations of the scanners (Check from field).



```
Scanner: 130.192.1.133.2009 : STOPPED
Scanner: 130.192.1.136.2010 : STOPPED
```

```
Name: FTPWriteAllowedRule Subject: [has role FTPWriteAllowedRole] Action: [[name : Write]] Object: [] Server: [[domainName : Server]]
Name: FTPWriteDeniedRule Subject: [has role FTPWriteDeniedRole] Action: [[name : Write]] Object: [] Server: [[domainName : Server]]
Name: FirewalRule.FilterWebPort80.input Subject: [[ip : 130.192.1.145, port : 80]] Action: [[name : sendPacket]] Object: [] Server: [[domai
Name: FirewalRule.FilterWebPort443.input Subject: [[ip : 130.192.1.145, port : 443]] Action: [[name : sendPacket]] Object: [] Server: [[don
```

PSC Settings
- Actions: [✓ Dangerous actions]
- Policy: Authentication, Authorization, Confidentiality, [✓ Filtering], Operational
- Accuracy: High [✓ Medium] Low
- Check from: [✓ Internet] [✓ Intranet] [✓ DMZ] Modem
- Users:

Fig. 5. Test network scheme, security policy and scanners state

During the investigation of the suggested approach the *experiments* for checking PSC functioning with different network configurations, security policies and monitoring methods were performed.

Let us consider only an example of *filtering policy testing* in the test network.

During checking, planner looks through all filtering policies (each policy contains all rules for particular firewall) and passes each policy to the scanner task generator. Planner uses appropriate method to check filtering policy completely. Scanner task generator forms the set of the checked connections. Generator looks through all filtering rules from the policy. It calculates set of the connections to test by subtracting from the set of the connections the set of the already checked connections. Then generator gets a connection from the set for checking and searches scanners to test the connection. One scanner is needed to send the packets through firewall and other to receive the packets beyond firewall. If there is no receiving scanner then generator can use existing service to check connection possibility. After checking, the report tab will be generated. Log tab will contain the scanner actions log for the testing period. Performed impact steps can be seen. Scanners tab will show the scanners state changing during testing.

6 Conclusion

The paper considered the suggested approach to the proactive monitoring of the network security policy. The suggested approach is based on the imitation of different users' actions in the investigated network which can disapprove or approve the fact that security policy holds true. The particularities, advantages and disadvantages of the suggested approach are determined. *The advantage* of the proactive monitoring approach suggested is that the action results are similar to the results of actions performed by users. It enables to get the representation on actual system behavior. The system configuration analysis does not give such representation because malefactor that controls the system can violate security mechanisms and (or) replace the analysis results [10, 13]. *The disadvantages* of the approach suggested are high complexity and low speed of checking in most cases, the restrictions due performing the potentially dangerous actions and the possibility of conflicts with users. Furthermore the proactive monitoring approach will not give results if information about violation is not contained in the policy, for example, if a malefactor leaved in the system a back door which can not be revealed by imitating the regular users' actions. Nevertheless the approach suggested allows confirming that system users have the privileges that were contained in the policy specification.

The approach is implemented in the proactive monitoring system "Proactive security scanner" (PSC). The main features of implementation are as follows: distributed architecture including multitude of scanners, available in different places of the network; maintaining a centralized automatic configuration of different scanners; gathering data from different scanners located in different places and centralized analyzing the scanning results; mechanism of interpreting and transforming system and policy specifications to the scripts of user actions on evaluating conformity of the current policy and system configuration to the specified security policy and system configuration; mechanism of automatic construction and replaying scripts of user actions taking into account various intentions of users and (or) malefactors.

PSC can work in several modes which are differentiated by the accuracy and checking speed: high accuracy checking (exhaustive search), medium and low accuracy checking, as well as optimized test sequences mode. PSC allows demonstrating the approach efficiently on the examples of filtering, channel protection, authorization and authentication policies. The developed optimization methods allow speeding up of the monitoring process greatly. The suggested approach to the monitoring may be used to fulfill the set of additional testing tasks. For instance it can be applied to imitation of DDoS attacks on the network resources. Such situation imitation can be performed for the server testing purpose. The future research directions are improving the suggested solutions and comprehensive theoretical and experimental evaluation.

Acknowledgments. This research is being supported by grant of Russian Foundation of Basic Research (Project No. 07-01-00547), program of fundamental research of the Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract No 3.2/03), Russian Science Support Foundation and partly funded by the EU as part of the POSITIF project (contract No. IST-2002-002314) and RE-TRUST (contract No. 021186-2).

References

1. Argenenko, A.Y., Chugaev, B.N.: Optimal binary questionnaires. Moscow, Energoatomizdat (in Russian) (1989)
2. Agrawal, D., Giles, J., Lee, K.-W., et al.: Policy-Based Validation of SAN Configuration. In: Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2004)

3. Barman S.: Writing Information Security Policies. Sams (2001)
4. Beigi, M.S., Calo, S., Verma, D.: Policy Transformation Techniques in Policy-Based Systems Management. In: Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2004)
5. Beizer, B.: Software testing techniques. International Thomson Computer Press (1990)
6. Canavan, S.: An Information Security Policy Development Guide for Large Companies. SANS Institute (2004), http://www.sans.org/rr/whitepapers/policyissues/1331.php
7. Carney, M., Loe, B.: A Comparison of Methods for Implementing Adaptive Security Policies. In: 7th USENIX Security Symposium (1998)
8. Common Information Model (CIM) Standards (2007), http://www.dmtf.org/standards/cim
9. El-Atawy, A., Ibrahim, K., Hamed, H., Al-Shaer, E.: Policy Segmentation for Intelligent Firewall Testing. In: The 1st Workshop on Secure Network Protocols (2005)
10. Foster, J.C., Price, M., McClure, S.: Sockets, Shellcode, Porting & Coding: Reverse Engineering Exploits and Tool Coding For Security Professionals. Syngress Publishing (2005)
11. Gama, P., Ferreira, P.: Obligation Policies: An Enforcement Platform. In: Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2005)
12. Ghosh, A.K., O'Connor, T., McGraw, G.: An Automated Approach for Identifying Potential Vulnerabilities in Software. In: 1998 IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (1998)
13. Hoglund, G., McGraw, G.: Exploiting Software. Addison-Wesley, Boston (2004)
14. IODEF/IDMEF Solutions (2004), http://www.ecsirt.net/service/products.html
15. Kee, C.K.: Security Policy Roadmap - Process for Creating Security Policies. SANS Institute (2001), http://www.sans.org/rr/whitepapers/policyissues/494.php
16. Klevinsky, T.J., Laliberte, S., Gupta, A., Hack, I.T.: Security through Penetration Testing. Addison Wesley, Boston (2002)
17. Marriott, D., Sloman, M.: Management Policy Service for Distributed Systems. In: Third IEEE International Workshop on Services in Distributed and Networked Environments, IEEE Computer Society Press, Los Alamitos (1996)
18. Peltier, T.R., Peltier, J., Blackley, J.A.: Managing a Network Vulnerability Assessment. Auerbach Publications (2003)
19. Positif Project (2007), http://www.positif.org
20. Rogers, R., Miles, G., Fuller, E., et al.: Security Assessment: Case Studies for Implementing the NSA IAM. Rockland: Syngress (2004)
21. Russell, D., Gangemi, G.T.: Computer Security Basics. O'Reilly&Associates (1991)
22. Sademies, A.: Process Approach to Information Security Metrics in Finnish Industry and State Institutions. Espoo: VTT Technical Research Centre of Finland (2004)
23. Sailer, R., Acharya, A., Beigi, M., Jennings, R., Verma, D.: IPSECvalidate A Tool to Validate IPSEC Configurations. In: 15th Conference on Systems Administration (2001)
24. Strembeck, M.: Embedding Policy Rules for Software-Based Systems in a Requirements Context. In: IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2005)
25. Wack, J., Tracy, M., Souppaya, M.: Guideline on Network Security Testing. NIST Special Publication pp. 800–842. Gaithersburg (2003)
26. Wheeler, K.: Distributed Firewall Policy Validation. CSE 598Z (Distributed Systems) Final Project (2004)

Comparing Electronic Battlefields: Using Mean Time-To-Compromise as a Comparative Security Metric

David John Leversage[1] and Eric James Byres[2]

[1] British Columbia Institute of Technology, 3700 Willingdon Avenue,
Burnaby, B.C., V5G 3H2, Canada
david_leversage@bcit.ca
[2] Byres Security Inc., P.O. Box 178
Lantzville, B.C., V0R 2H0, Canada
eric@byressecurity.com

Abstract. The ability to efficiently compare differing security solutions for effectiveness is often considered lacking from a management perspective. To address this we propose a methodology for estimating the mean time-to-compromise (MTTC) of a target device or network as a comparative metric. A topological map of the target system is divided into attack zones, allowing each zone to be described with its own state-space model (SSM). We then employ a SSM based on models used in the biological sciences to predict animal behavior in the context of predator prey relationships. Markov chains identify predominant attacker strategies which are used to build the MTTC intervals which can be compared for a broad range of mitigating actions. This allows security architects and managers to intelligently select the most effective solution, based on the lowest cost/MTTC ratio that still exceeds a benchmark level.

Keywords: Network Security, SCADA Security, Time-to-Compromise, Markov Chains, Predator Model, Attack Paths, Attack Zones, Attack Trees.

1 Introduction

One of the challenges faced by any network security professional is providing a simple yet meaningful estimate of a system or network's security preparedness to management who are not security professionals. While it can be relatively easy to enumerate specific flaws in a system, seemingly simple questions like "*How much more secure will our system be if we invest in this technology*?" or "*How does our security preparedness compare to other companies in our sector*?" can prove to be a serious stumbling block to moving a security project forward.

This has been particularly true for our particular area of research, namely the security of Supervisory Control and Data Acquisition (SCADA) and Industrial Automation and Control Systems (IACS) used in critical infrastructures such as electricity generation/distribution, petroleum production/refining and water management. Companies operating these systems are being asked to invest significant resources towards improving the security of their systems, but management's understanding of the risks and benefits is often vague. Furthermore, competing interests for the limited security dollars have often left many companies making decisions based on the best sales pitch rather than a well-reasoned security program.

The companies operating in these sectors are not unsophisticated – most have had many years of experience making intelligent business decisions on a daily basis on a large variety of multifaceted issues. For example, the optimization of hundreds (or thousands) of process feedback loops in the refining and chemicals industries (typically called control loops) is both extremely complex and critical to profitable operations. Yet, models based on the concept of Key Performance Indicators (KPI) have proven to be successful in simplifying the problem to the point where upper management can make well reasoned decisions on global operations without getting mired in the details. [1]

In our discussions with these companies, it was repeatedly pointed out that similar types of performance indicators could be very useful for making corporate security decisions. What was wanted was not a proof of absolute security, but rather a measure of relative security.

To address this need, we propose the concept of a mean time-to-compromise (MTTC) interval as an estimate of the time it will take for an attacker within a specific skills level to successfully impact the target system.

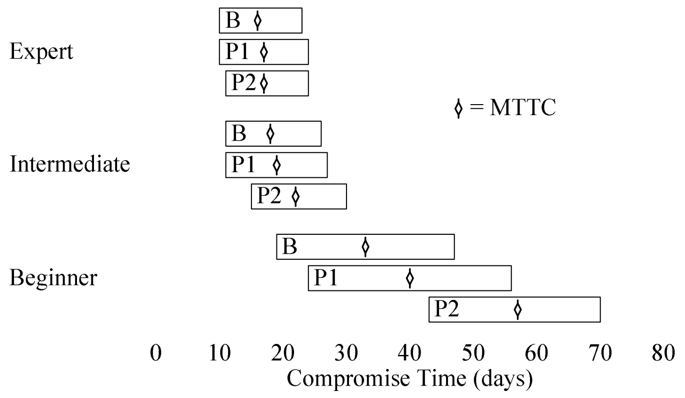


Fig. 1. Example of estimated MTTC intervals (in days) for the network shown in Fig. 2. MTTC intervals are grouped into threes for each attacker skill level and are for the case study given near the end of this paper. The top interval from each group (B) represents the baseline system, the middle interval (P1) represents more frequent patching of nodes on the primary enterprise network, and the bottom interval (P2) represents more frequent firewall rule reviews of the Internet facing firewall.

The concept of MTTC is not new – for example, Jonsson uses mean-time-to-breach to analyze attacker behaviors [2] and the Honeynet community uses MTTC as a measure of a system's ability to survive exposure to the Internet [3]. The key point with these works is that MTTC was seen as an observable variable rather than calculated indicator of relative security. McQueen et al [4] [5] moved toward the latter concept with a methodology that employed directed graphs to calculate an expected time-to-compromise for differing attacker skill levels (The second paper also offers an excellent history of related work). Other works look at probabilistic models to estimate security. However, as McQueen et al points out, many of the techniques proposed for estimating cyber security tend to require significant detail about the target system, making them unmanageable as a comparative tool for multiple systems.

To address this, our model focuses on being a comparative tool and proposes a number of averaging techniques to allow it to become a more generally applicable methodology while still allowing meaningful comparisons. We also developed our model, along with its supporting methodology, with emerging industrial security standards in mind – specifically those being developed by the International Electrotechnical Commission (IEC) [6] and by the International Society for Measurement and Control (ISA) [7] [8].

2 Lessons Learnt from Physical Security

Determining the burglary rating of a safe is a similar problem to determining the security rating of a network. Both involve a malicious threat agent attempting to compromise the system and take action resulting in loss. Safes in the United States are assigned a burglary and fire rating based on well defined Underwriters Laboratory (UL) testing methodologies such as UL Standard 687 [9]. A few selected UL safe burglary ratings are given in Table 1.

Table 1. Selected UL Safe Burglary Ratings

UL Rating	NWT (Min.)	Testing Interpretation
TL-15	15	Tool-Resistant (face only)
TL-30	30	Tool-Resistant (face only)
TRTL-15X6	15	Torch & Tool-Resistant (6 Sides)
TRTL-30X6	30	Torch & Tool-Resistant (6 Sides)
TXTL-60	60	Torch & Tool-Resistant

This rating system is based around the concept of "*Net working time" (NWT),* the UL expression for the time that is spent attempting to break into the safe by testers using specified sets of tools such as diamond grinding tools and high-speed carbide-tip drills. Thus TL-15 means that the safe has been tested for a NWT of 15 minutes using high speed drills, saws and other sophisticated penetrating equipment. The sets of tools allowed are also categorized into levels - TRTL-30 indicates that the safe has been tested for a NWT of 30 minutes, but with an extended range of tools such as torches.

Our discussions with UL testing engineers confirmed that design level knowledge about the safe is used in planning and executing the attacks. They also confirmed that although there are maybe dozens of strategies (classified as attack types) that can be used to gain access to the safe, only a few are actually tried. Finally, each surface of the safe represents an attack zone which may alter the strategies used by the attacker.

There are a few observations about this process that merit mention:

1. There is an implication that given the proper resources and enough time, any safe can eventually be broken into.
2. A safe is given a burglary rating based on its ability to withstand a focused attack by a team of knowledgeable safe crackers following a well defined set of rules and procedures for testing.
3. The rules include using well-defined sets of common resources for safe cracking.
4. The resources available to the testers are organized into well-defined levels that represent increasing cost and complexity and decreasing availably to the average attacker.
5. Even though there might be other possibilities for attack, only a limited set of strategies will be used, based on the tester's detailed knowledge of the safe.

Most important, the UL rating does not attempt to promise that the safe is secure from all possible attacks strategies – it is entirely possible that a design flaw might be uncovered that would allow an attacker to break into a given safe in seconds. However, from a statistical point of view, it is reasonable to assume that as a group, TL-30 safes are more secure than TL-15 safes. This ability to efficiently estimate a comparative security level for a given system is the core objective of our proposed methodology.

Learning from the philosophy of rating safes, our methodology for rating a target network makes the following assumptions:

1. Given the proper resources and enough time, any network can be successfully attacked by an agent skilled in the art of electronic warfare.
2. A target network or device must be capable of surviving an attack for some minimally acceptable benchmark period (the MTTC).
3. The average attacker will typically use a limited set of strategies based on their expertise and their knowledge of the target.
4. Attackers can be statistically grouped in to levels, each with a common set of resources such as access to popular attack tools or a level of technical knowledge and skill.

3 Attack Zones

Just like a safe has different sides that require their own attack strategies, we believe that networks have the same characteristic, namely that a complex network can be divided into zones that are generally homogeneous. Thus we begin by dividing a topological map of the target network into attack zones as is shown in Fig. 2. In this particular case, the target of interest is Zone 1, is a process control network (PCN) that is buried inside a corporate enterprise network (EN), which in turn is connected to the Internet[1]. Each zone represents a network or network of networks separated

[1] This is a very common architecture in SCADA systems. For example, see "NISCC Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks", http://www.cpni.gov.uk/docs/re-20050223-00157.pdf

from other zones by boundary devices. Within a zone it is assumed that there are consistent security practices in effect such as operating system deployment, patching practices and communications protocol usage. These practices could be good or bad (i.e. patching is performed randomly by users), but they are consistent within the zone.

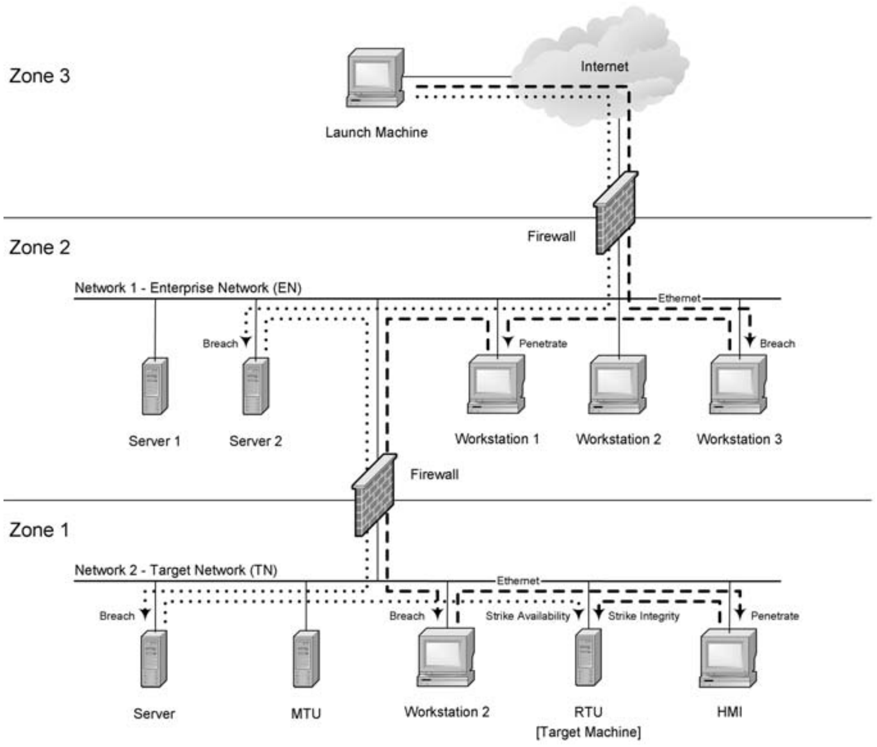


Fig. 2. An example illustrating attack zones and attacker movement through the zones to strike a target device on the target network. The dashed and dotted lines represent two different attack paths that are also represented by the same patterned lines on the attack path model of this system shown in Fig. 4. This topology is used for the case study presented near the end of this paper.

The concept of zones is important for two reasons. First, an attacker staging an attack from within the target network will likely employ a different set of strategies than he/she would from the Internet and dividing the topological map into zones allows us to represent each zone with its own SSM. Second, by assuming consistent application of practice within a zone, we can make important simplifications to the model to keep it manageable.

4 Predator Model

Papers by Sean Gorman [10] and Erland Jonsson [2] provided the motivation and insight to pursue a predator prey-based SSM. For the purposes of this paper, our proposed SSM, shown in Fig. 3, is for attacks launched from the Internet. In it we have defined three general states:

1. *Breaching* occurs when the attacker takes action to circumvent a boundary device to gain user or root access to a node on the other side of the boundary.
2. *Penetration* is when the attacker gains user or root access to a node without crossing a boundary device.
3. *Striking* is taking action to impact the confidentiality, integrity (take unauthorized control) or availability (deny authorized access) of the target system or device.

While it is possible to hypothesize many more states (and some may prove to be necessary), our experimentation indicates that having more than five states adds little to the output of the model, yet greatly increases the complexity of the calculations. For example, McQueen and others suggests *Reconnaissance* states. However, we feel that this can add a significant level of complexity to the process since virtually every state will require some reconnaissance in order to be transited. Thus reconnaissance could just be considered a sub-state and included as part of a primary state's calculations.

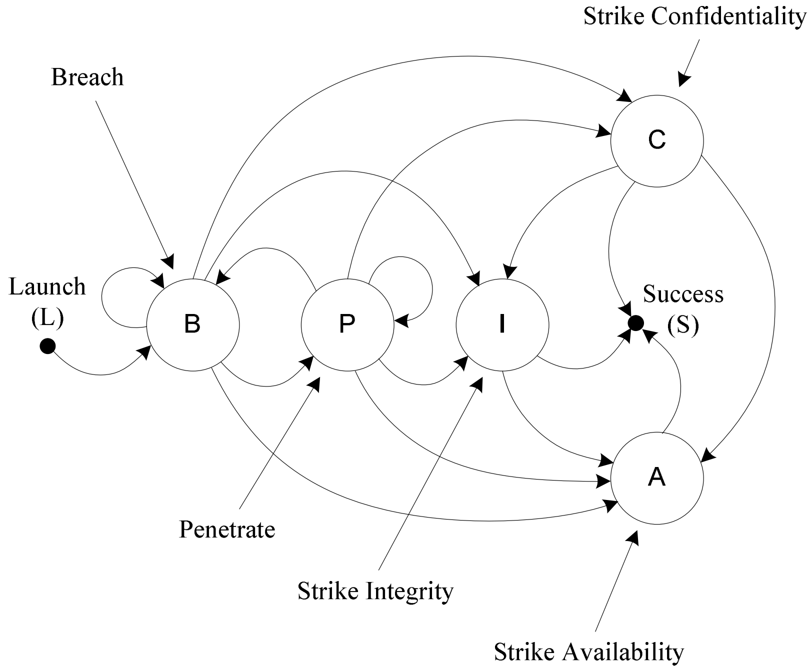


Fig. 3. SSM of attacker movement for attacks launched from the Internet

The attacker compromises one or more nodes as he/she moves towards the target network as is shown in Fig. 2. With layered network architectures, the resulting sequence of compromised nodes appears as movement towards the target and the attacker's strategy, called an attack path, is betrayed by the sequence of states — a Markov chain.

5 Attack Path Model

The state-space predator model is used to map out the attack path model, a SSM of all possible attack paths from the launch node to the target device taking network topology and security policies into consideration. Consider the network shown in Fig. 2. If we make the simplifying assumptions that: the attacker only moves forward towards the target, the firewalls cannot be compromised, and the target device cannot be compromised from a device outside of its zone, then the resulting attack path model is as shown in Fig. 4.

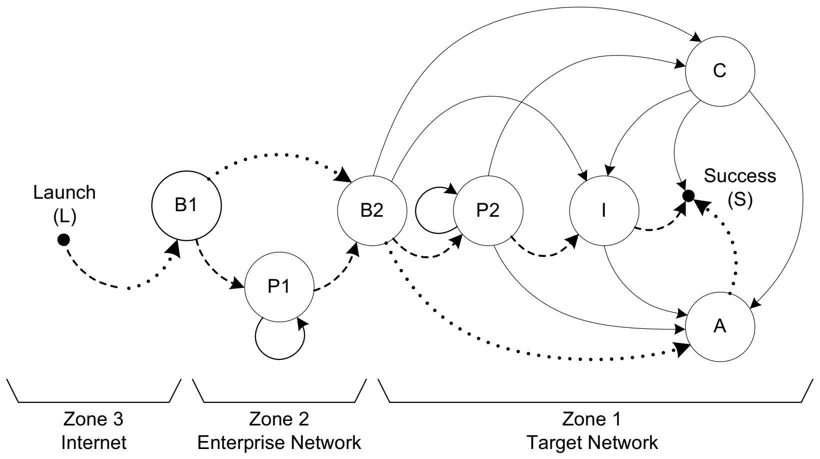


Fig. 4. Attack path model for the network shown in Fig. 2 with simplifying assumptions. The dashed and dotted paths correspond to the same patterned attack paths as are shown in Fig. 2. State times are given in tables 2, 3 and 4 for the case study given near the end of this paper.

The assumption that the attacker only moves forward towards the target reflects our philosophy that a motivated attacker will not deliberately increase their attack time with unnecessary actions. The second and third assumptions may not be typical of most networks and could be removed. However for the purpose of this paper, they simplify the attack path model to clearly illustrate its salient features.

6 Estimating State Times

The next step is to estimate state times and there are numerous methodologies that can be used for this purpose. In this paper we present two; a statistical algorithm based on a modified version of McQueen et al's Time to Compromise Model (TTCM) [5] and an attack tree-based technique. The first allows us to estimate the duration of the breach and penetration states while the second is used to obtain a general and formalized estimate for the strike states in control systems where algorithms are not yet available.

6.1 The State-Time Estimation Algorithm (STEA)[2]

The attacker's actions are divided into three statistical processes:

- Process 1 is when the attacker has identified one or more known vulnerabilities AND has one or more exploits on hand.
- Process 2 is when the attacker has identified one or more known vulnerabilities; however, he does not have an exploit on hand.
- The attacker is in process 3 when there are no known vulnerabilities and no known exploits available.

The total time of all three processes is the estimated state time (T) as is shown in (1).

$$T = t_1 P_1 + t_2 (1 - P_1)(1 - u) + t_3 u (1 - P_1) \tag{1}$$

Where:	T	=	estimated state time
	t_1	=	mean time that the attacker is in process 1
	P_1	=	probability that the attacker is in process 1
	t_2	=	mean time that the attacker is in process 2
	u	=	probability that the attacker is in process 3
	t_3	=	mean time that the attacker is in process 3

Process 1
Process 1 is hypothesized to have a mean time of 1 day as is shown in (2). We expect this time to change with experience and we defer to McQueen et al [4] for supporting arguments.

$$t_1 = 1 \text{ day} \tag{2}$$

The probability that the attacker is in process 1 is shown in (3).

$$P_1 = 1 - e^{-V \times M / K} \tag{3}$$

Where:	P_1	=	probability that the attacker is in process 1
	V	=	average number of vulnerabilities per node within a zone
	M	=	number of readily available exploits available to the attacker
	K	=	total number of non-duplicate vulnerabilities

In the absence of statistical data, we hypothesize that the distribution of attackers versus skills levels to be a Normal Distribution and we introduce a skills indicator which represents the percentile rating of the attacker and can take on any value from 0 (absolute beginner) to 1 (highly skilled attacker).

[2] To differentiate between the original TTCM of McQueen et al and our modified version we call our version the State-Time Estimation Algorithm (STEA).

M is the product of the skills multiplier and the total number of readily available exploits available to all attackers (m). McQueen chose m to be 450 based on exploit code publicly available over the Internet through sites such as Metasploit. [11] We used the same value for "m" and multiplied by the skills multiplier to get "M" for both the breach and penetration states.

K represents the number of non-duplicate software vulnerabilities in the ICAT database for both the breach and penetration states. We hypothesize that it can be extended represent other classes of vulnerabilities, such as the number of non-duplicate vulnerabilities in the protocol being used to strike the target device.

Process 2

Process 2 is hypothesized to have a mean time of 5.8 days. Again we expect this time to change with experience and we defer to McQueen et al. for supporting arguments. [4]

$$\mathrm{ET} = \frac{\mathrm{AM}}{\mathrm{V}} * \left(1 + \sum_{\mathrm{tries}=2}^{\mathrm{V}-\mathrm{AM}+1} \left[\mathrm{tries} * \prod_{\mathrm{i}=2}^{\mathrm{tries}} \left(\frac{\mathrm{NM}-\mathrm{i}+2}{\mathrm{V}-\mathrm{i}+1}\right)\right]\right) \tag{4}$$

Where: ET = expected number of tries
V = average number of vulnerabilities per node within a zone
AM = average number of the vulnerabilities for which an exploit can be found or created by the attacker given their skill level
NM = number of vulnerabilities that this skill level of attacker won't be able to use

$$t_2 = 5.8\mathrm{days} \times \mathrm{ET} \tag{5}$$

Where: t_2 = mean time that the attacker is in process 2
ET = expected number of tries

Process 3

This process hypothesizes that the rate of new vulnerabilities or exploits becomes constant over time. [12] To calculate this we need a probability variable u that indicates that process 2 is unsuccessful.

$$u = (1-s)^{V} \tag{6}$$

Where: u = probability that the attacker is in process 3
s = attacker skill level (0 to 1)
V = average number of vulnerabilities per node within a zone

$$t_3 = ((1/s) - 0.5) \times 30.42 + 5.8 \tag{7}$$

Where: t_3 = mean time that the attacker is in process 3
s = attacker skill level (0 to 1)

Equations (6) and (7) differ from the McQueen equations in that AM/V has been replaced with s (the skills factor).

The strength in the STEA model is that can be modified to include other time for sub-states (such as reconnaissance) and can also be adapted to incorporate environmental variables that effect the state times (such as patching intervals). As an example of this flexibility, the study team decided to include a rather abstract variable into the calculation– the frequency of access control list rule reviews. To do this we first assumed that boundary devices like routers and firewalls offer security by reducing the number of vulnerabilities that are visible to the attacker. In other terms, only a portion of the network's attack surface is visible to the attacker. [13] We then assume that the effectiveness of any boundary device decays if its rule sets are not reviewed regularly [14]. We then incorporated this relationship to the Equations (3) and (6) to produce equations (8) and (9).

$$P_1 = 1 - e^{-\alpha \times V \times M/K} \tag{8}$$

$$u = (1 - s)^{\alpha \times V} \tag{9}$$

Where: α = visibility ($\alpha = 1$ when estimating penetration state times)

Finally we worked with a firewall expert at the British Columbia Institute of Technology to come up with a possible correlation between visibility and update/review frequency. His estimation is: No Reviews, $\alpha = 1.00$, Semi-Annual, $\alpha = 0.30$; Quarterly, $\alpha = 0.12$; Monthly, $\alpha = 0.05$. Further research is needed to provide support for these estimations, but as a proof of concept they are sufficient.

This is one example of the opportunity to add environmental variables that may eventually prove to be important indicators of relative security performance. Other factors we have experimented with include patch intervals, operating system diversity and password policies. If industrial control loop optimization research is any indication, which indicators are truly important and how they affect the MTTC will be an area for considerable future research.

6.2 Estimating Strike State Times Using Attack Trees

In many cases analytical models are not yet available for a given state. For example, in the industrial controls world inherent vulnerabilities in the SCADA protocols themselves appear to have far more impact on the security than operating system or application vulnerabilities [15] and it is not clear if the STEA assumptions apply. To address this issue, our research activities have included exploring ways attack trees can be used to estimate state times.

We developed an attack tree methodology whereby the attacker's strategy maps to a forest of trees and yet remains bound by using a limited set of actions that can be taken at the end nodes based on Military lexicon.

Fig. 5 illustrates a partial attack tree for breaching the EN by compromising Workstation #1 through software vulnerabilities. Notice that the root of the tree represents goal of attacker and the state. The next layer of nodes represents a physical device under attack. The third layer identifies the failure mechanism (the vulnerability) and the final layer represents the exploit capabilities of the attacker.

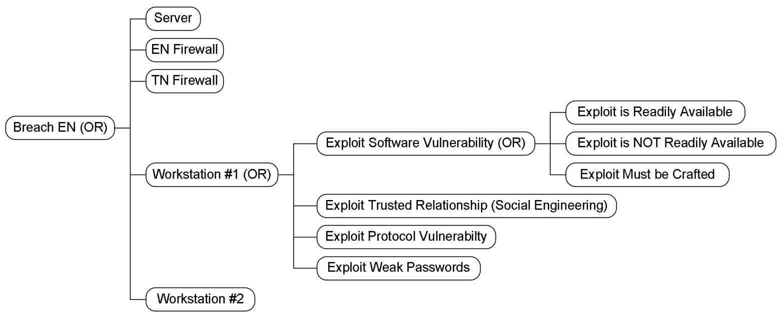

Server
EN Firewall
TN Firewall
Breach EN (OR)
Workstation #1 (OR)
Exploit Software Vulnerability (OR)
Exploit is Readily Available
Exploit is NOT Readily Available
Exploit Must be Crafted
Exploit Trusted Relationship (Social Engineering)
Exploit Protocol Vulnerabilty
Exploit Weak Passwords
Workstation #2

Fig. 5. A partial breach EN tree with software vulnerability exploits expanded

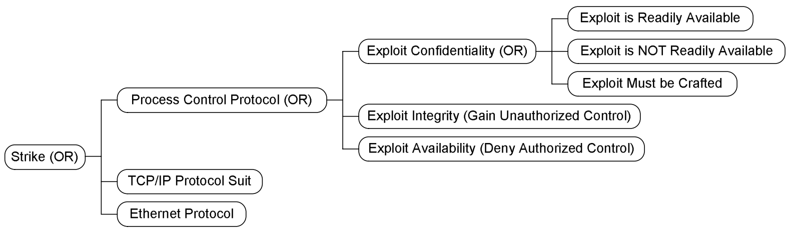

Strike (OR)
Process Control Protocol (OR)
Exploit Confidentiality (OR)
Exploit is Readily Available
Exploit is NOT Readily Available
Exploit Must be Crafted
Exploit Integrity (Gain Unauthorized Control)
Exploit Availability (Deny Authorized Control)
TCP/IP Protocol Suit
Ethernet Protocol

Fig. 6. A partial strike tree focusing on protocol vulnerability exploitation

Fig. 6 illustrates a partial strike tree that focuses on vulnerabilities in the SCADA protocols found in the target network. The root of this tree also represents goal of attacker and the state. The next layer of nodes represents the protocol (or protocols) used to attack the target. Layer three identifies the failure mechanism (the vulnerability) based on data communication security goals as they are outlined in IEC/TR 62210. [9] The final layer represents the exploit capabilities of the attacker.

Notice the overall similarity and close mapping between Fig. 5 and 6. The first layer of nodes represents the object (or objects) under attack. The second layer identifies the failure mechanisms (the vulnerabilities) and the third layer represents the exploit capabilities of the attacker.

We use attack trees to estimate the strike state's time for an attacker to: exploit confidentiality, exploit integrity or exploit availability. Child nodes are based on RFC 3552 [16] and US-CERT publications [17].

Unlike traditional capabilities based attack trees, subject matter experts estimate the time they would need to successfully craft a working exploit for attacks belonging to one or more of the strike state's categories. These times are used in calculating the strike state time when building estimated MTTC.

7 Building MTTC Intervals

Ideally, MTTC intervals should be based on predominant strategies used by attackers.

Until reliable statistical data is available, each attack path is given an equal probability and the attack path model is truncated to allow only one penetration of each network. In practice we expect this not to be true; however, this suffices to provide a metric whereby two or more systems can be compared. Results of Honeynet research would be extremely useful for this task. Each attack path time is estimated for each attacker skill level and the interval for each skill level is built from the shortest and longest attack path time. The product of each attack path probability and its mean time are summed to produce a mathematical expectation for the MTTC itself.

8 Case Study

A utility company wanted to compare mitigating solutions at one of its facilities to determine how to best focus its resources. The system had a topology similar to the system shown in Fig. 2 with an average of 6 and 10 vulnerabilities per node on the EN and PCN respectively. (Note: similar topologies are not required by the framework and are only used for illustrative purposes). Firewall reviews on both the Internet facing and Target Network facing firewalls were done on an annual basis. There is limited manpower and financial resources for security and management wants to evaluate two differing approaches. The first is to focus on patching systems on the primary enterprise network that makes up Zone 2. The second is to increase firewall rules reviews from yearly to quarterly on the Internet facing firewall. State times for the baseline system are given in table 2.

Table 2. State times (in days) for the baseline system

	B1	P1	B2	P2	C	I	A
Expert	4.6	4.6	4.0	4.0	1.0	4.0	1.0
Intermediate	5.2	5.2	4.5	4.5	1.0	4.5	1.0
Beginner	9.5	9.5	8.6	8.6	1.0	8.6	1.0

IT security determined that the number of man hours it would take to reduce the average number of vulnerabilities on the enterprise network from 6 to 3 per node is about the same man hours as it would take to do firewall reviews on the Internet facing firewall on a quarterly basis. State times for both of these approaches are given in tables 3 and 4 respectively.

Table 3. State times (in days) for increased patching frequency of the enterprise network nodes reducing the average number of vulnerab ilities per node to 3

	B1	P1	B2	P2	C	I	A
Expert	5.2	5.2	4.0	4.0	1.0	4.0	1.0
Intermediate	5.8	5.8	4.5	4.5	1.0	4.5	1.0
Beginner	13.9	13.9	8.6	8.6	1.0	8.6	1.0

Table 4. State times (in days) for qurterly firewall reviews on the Internet facing firewall

	B1	P1	B2	P2	C	I	A
Expert	5.6	4.6	4.0	4.0	1.0	4.0	1.0
Intermediate	9.1	5.2	4.5	4.5	1.0	4.5	1.0
Beginner	33.0	9.5	8.6	8.6	1.0	8.6	1.0

MTTC levels were estimated for the baseline system and both proposals and are shown in table 2.

Table 5. Estimated MTTC values (in days) for each attacker skill level

	Expert	Intermediate	Beginner
Baseline	16.3	18.3	33.2
Increased Patching	16.8	19.2	39.8
Increased Rules Reviews	16.9	22.2	56.7

IT security determined that both approaches could be implemented using existing resources (primarily human) and each was estimated to cost about $15,000. The resulting cost per day of MTTC being bought (the cost / Δ MTTC ratio) for each attacker skill level is shown in table 6.

Table 6. Cost / ΔMTTC ratios for each attacker skill level

	Expert	Intermediate	Beginner
Increased Patching	$30,000	$16,667	$2,273
Increased Rules Reviews	$25,000	$3,846	$638

Within the framework of an overall qualitative risk assessment, this information could be used to decide if increased rules reviews on the Internet facing firewall is the most effective use of company resources. Like in the case of safe testing, the real strength of this methodology is not for obtaining absolute values of security, but rather relative values for comparing differing systems and solutions.

9 Future Research

Currently the STEA methodology focuses primarily on vulnerabilities of a software nature which are exploited by attacks launched from the Internet. However, we hypothesize that it can be further modified to estimate the state times for other vulnerabilities including human related vulnerabilities (i.e. poor password selection) and protocol vulnerabilities resulting in MTTC intervals for a broad range of vulnerabilities and therefore mitigating actions.

Consider the scenarios where a threat agent breaches a plant's physical security and then logs onto an Human Machine Interface and strikes the confidentiality or integrity of the system. Or consider another attacker who takes a sledge hammer to a remotely situated target device and strikes availability. These scenarios involve four states: breach, strike confidentiality, strike integrity and strike availability - remarkably similar to the states in the SSM presented in this paper. We therefore expect that our SSM can be modified to identify attack paths for other attack classes such as social engineering or physical attacks. Similarly, we also expect that the STEA can be modified to estimate state times for other vulnerability classes such as protocol and human vulnerabilities. Identifying and describing a set of models that cover the entire attack surface of the target system is an area of considerable future research and this is where we are pursuing a Hierarchical Holographic Model which will act as the glue to unify our models.

Relevant statistical data to set the MMTC intervals confidence levels also needs to be collected and promising sources for this statistical data are the Honeynet Project [19] and the results of penetration team testing in the field. Both will help us to improve our state time estimations and to identify predominant attacker strategies. Our experience with the Industrial Security Incident Database leads us to believe that this may even help identify how an attacker's strategies are modified according to environmental conditions (network topology, defenses, etc) and attacker skill levels.

We hypothesized that the distribution of attackers with skills ranging from beginner to expert to be normal distribution. Recent research has us pursuing key risk indicators to identify the key skills and resources used for each of the three attacker levels and to relate these to the attacker's skill level through learning curve theory.

10 Conclusions

The finding of this preliminary research indicates that MTTC could be an efficient yet powerful tool for a comparative analysis of security environments and solutions.

The selection of time as the unit of measurement is paramount to the model's strength. Time intervals can be used to intelligently compare and select from a broad range of mitigating actions. Two or more entirely different mitigating solutions can be compared and chosen based on which solution has the lowest cost in dollars per day and yet meets or exceeds a benchmark MTTC.

Another important relationship that can be realized is how hard or weak a system is as seen by the attacker compared with peer systems in the same industry. MTTC industry averages (and other averages) can be calculated over time giving and can be used for making peer comparisons. Having MTTC intervals above the average should imply that an opportunistic attacker is more likely to move on to another target whereas MTTC intervals below the average should imply the opposite. However, this is also an area of considerable research.

References

1. Desborough, L., Miller, R.: Increasing Customer Value of Industrial Control Performance Monitoring – Honeywell's Experience. In: Proc. 6th Int. Conf. on Chemical Process Control (CPC VI), pp. 172–192 (2001)
2. Jonsson, E., Olovsson, T.: A Quantitative Model of the Security Intrusion Process Based on Attacker Behaviour. IEEE Transactions on Software Engineering 23(4) (1997)
3. http://archives.neohapsis.com/archives/sf/honeypots/2002-q3/0032.html

4. McQueen, M., Boyer, W., Flynn, M., Beitel, G.: Time-to-Compromise Model for Cyber Risk Reduction Estimation. In: First Workshop on Quality of Protection (2005)
5. McQueen, M., Boyer, W., Flynn, M., Beitel, G.: Quantitative Cyber Risk Reduction Estimation Methodology for a Small SCADA Control System. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS) (2006)
6. IEC TR 62210: Power System Control and Associated Communications – Data and Communication Security. International Electrotechnical Commission (2003)
7. ISA-99.00.01: Security for Industrial Automation and Control Systems Part 1: Concepts, Terminology and Models (Draft). International Society for Measurement and Control (ISA) (2006)
8. ISA-99.00.02: Security for Industrial Automation and Control Systems Part 2: Establishing an Industrial Automation and Control System Security Program (Draft). International Society for Measurement and Control (ISA) (2006)
9. UL 687: Standard for Safety Burglary-Resistant Safes. Underwriters Laboratories Inc. (2005)
10. Gorman, S., Kulkarni, R., Schintler, L., Stough, R.: A Predator Prey Approach to the Network Structure of Cyberspace. In: ACM International Conference Proceeding Series, vol. 58 (2004)
11. http://www.metasploit.com/
12. Rescorla, E.: Is Finding Security Holes a Good Idea. IEEE Security & Privacy (2005)
13. Manadhata, P., Wing, J.: Measuring A System's Attack Surface. Technical Report CMU-CS-04-102, School of Computer Science, Carnegie Mellon University (2004)
14. Wool, A.: A quantitative study of firewall configuration errors. IEEE Computer Magazine, IEEE Computer Society, 62–67 (2004)
15. Byres, E., Franz, M., Miller, D.: The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems. In: International Infrastructure Survivability Workshop (IISW), IEEE, Los Alamitos (2004)
16. RFC 3552: Security Considerations Guidelines. Internet Engineering Task Force (2003)
17. http://www.cert.org/
18. DNP3 Documentation Library, http://www.dnp.org/
19. http://www.honeynet.org/

Abstraction Based Verification of a Parameterised Policy Controlled System

Peter Ochsenschläger and Roland Rieke[⋆]

Fraunhofer Institute for Secure Information Technology SIT, Darmstadt, Germany
{ochsenschlaeger,rieke}@sit.fraunhofer.de

Abstract. Safety critical and business critical systems are usually controlled by policies with the objective to guarantee a variety of safety, liveness and security properties. Traditional model checking techniques allow a verification of the required behaviour only for systems with very few components. To be able to verify entire families of systems, independent of the exact number of replicated components, we developed an abstraction based approach to extend our current tool supported verification techniques to such families of systems that are usually parameterised by a number of replicated identical components. We demonstrate our technique by an exemplary verification of security and liveness properties of a simple parameterised collaboration scenario. Verification results for configurations with fixed numbers of components are used to choose an appropriate property preserving abstraction that provides the basis for an inductive proof that generalises the results for a family of systems with arbitrary settings of parameters.

Keywords: Formal analysis of security and liveness properties, security modelling and simulation, security policies, parameterised models.

1 Introduction

In a typical policy controlled system, a set of policy rules, posing restrictions on the system's behaviour, is used to enforce the required security objectives, such as confidentiality, integrity and availability. For safety critical systems as well as for business critical systems or parts thereof, assuring the correctness - conformance to the intended purpose - is imperative. These systems must guarantee a variety of safety, liveness and security properties.

The problem approached. Traditional model checking techniques can be used to analyse such systems and to understand and verify how they behave subject to different policy constraints. However, because of well known state explosion problems, the usage of these techniques is limited to systems with very few components. In this paper we propose an extension of these techniques to a particularly interesting class of systems called *parameterised systems*. A parameterised system describes a family of systems that are finite-state in nature but

[⋆] Part of the work presented in this paper was developed within the project SicAri being funded by the German Ministry of Education and Research.

scalable. A formal specification of a parameterised system thus covers a family of systems, each member of which has a different number of replicated components. Instances of the family can be obtained by fixing the parameters. Extensions of model checking techniques are required that support verification of properties that are valid independently of given concrete parameters.

Contributions. To be able to verify entire families of critical systems, independent of the exact number of replicated components, we developed an *abstraction based approach* to extend our current tool supported verification techniques to such parameterised systems. Abstraction is a fundamental and widely-used verification technique. It can be used to reduce the verification of a property over a concrete system, to checking a related property over a simpler abstract system [1]. In this paper however we need an inductive proof on the construction of the behaviour of the parameterised system to show that it results in identical abstract system behaviour for any given parameter configuration. This allows the verification of parameterised systems by constructing abstract systems that can be model checked.

In the case of our abstraction based approach, the key problem is the choice of an appropriate abstraction that, (1) is *property preserving*, (2) results in identical abstract system behaviour for any given parameter configuration, and, (3) is sufficiently precise to express the required properties at the chosen abstraction level. To solve this problem, we

– compute the system behaviour and verify the required properties for some configurations with fixed numbers of components;
– we then use the results to choose an appropriate property preserving abstraction that results in identical abstract system behaviour for any given parameter configuration;
– based on this abstraction, we provide an inductive proof (by hand) that generalises the results for a family of systems with arbitrary settings of parameters.

In this paper we demonstrate our technique by an exemplary verification of security and liveness properties of a simple parameterised collaboration scenario.

The subsequent paper is structured as follows. In Sect. 2 we review some related work. Section 3 introduces a collaboration scenario that we will use throughout this paper to illustrate the usage of the proposed method for analysis of parameterised models. Section 4 describes the formal modelling technique, the abstraction based verification concept and the verification tool while Sect. 5 presents an exemplary verification of the collaboration scenario. Finally, the paper ends with conclusions and an outlook in Sect. 6.

2 Related Work

Analysis of security policies. The research in the field of security policies has gained increasing attention in the past few years. Many research papers appeared that investigated security policies on its own and abstracted from the systems needed to enforce these policies. These activities concentrated on the examination of specific properties of policies like consistency, freedom of conflicts, information flow implications and effects to system safety. This allows shifting the attention from specifics of computer system towards the analysis of properties that are inherent to the policy itself.

In the information flow analysis approach presented in [2] for the *SELinux* system, a labelled transition system (LTS) is generated from the policy specifications that models the information flow policy. Temporal logic formulas are used to specify the security goals. The *NuSMV* (`http://nusmv.irst.itc.it/`) model-checker verifies the security goals on this LTS.

A method to enforce rigorous automated network security management using a network access control policy is presented in [3]. This method is illustrated using examples based on enforcement strategy by distributed packet filtering and confidentiality/authenticity goals enforced by IPsec mechanisms.

In [4] a model-based approach focussing on the validation of network security policies and the interplay of threats and vulnerabilities and system's behaviour is proposed. This approach is based on Asynchronous Product Automata (APA) [5]. APA are also used as a basis of the work presented in this paper.

Verification approaches for parameterised systems. An extension to the *Murφ* verifier to verify systems with replicated identical components through a new data type called RepetitiveID (with restricted usage) is presented in [6]. The verification is performed by explicit state enumeration in an abstract state space where states do not record the exact numbers of components. Murφ automatically checks the soundness of this abstraction and translates the system description to an abstract state graph for a system of a fixed size. During the verification of this system, Murφ uses a run-time check to determine if the result can be generalised for a family of systems. The soundness of the abstraction algorithm is guaranteed by the restrictions on the use of repetitiveIDs. These restrictions allow Murφ to decide which components are abstractable using the repetition constructors, enforce symmetry in the system, which enables the automatic construction of abstract states, and, enforce the repetitive property in the system, which enables the automatic construction of the abstract successors. A typical application area of this tool are cache coherence protocols. Many cache coherence protocols satisfy the above restrictions.

The aim of [7] is an abstraction method through symmetry which works also when using variables holding references to other processes which is not possible in Murφ. An implementation of this approach for the *SPIN* model-checker (`http://spinroot.com/`) is described.

In [8] a methodology for constructing abstractions and refining them by analysing counter-examples is presented. The method combines abstraction, model-checking and deductive verification and in particular, allows to use the set of reachable states of the abstract system in a deductive proof even when the abstract model does not satisfy the specification and when it simulates the concrete system with respect to a weaker simulation notion than Milner's. The tool *InVeSt* supports this approach and makes use of *PVS* (`http://pvs.csl.sri.com/`) and

SMV (`http://www.cs.cmu.edu/ modelcheck/smv.html`). This approach however does not consider liveness properties.

In [9] a technique for automatic verification of parameterised systems based on process algebra *CCS* [10] and the logic modal *mu-calculus* [11] is presented. This technique views processes as property transformers and is based on computing the limit of a sequence of mu-calculus formula generated by these transformers.

The above-mentioned approaches demonstrate, that finite state methods combined with deductive methods can be applied to analyse parameterised systems. The approaches differ in varying amounts of user intervention and their range of application. A survey of a number of approaches to combine model checking and theorem proving methods is given in [12].

Characteristic of our approach is the flexibility of abstractions defined by language homomorphisms and the consideration of liveness properties.

3 Collaboration Scenario

There are manifold uses and aspects of the terms *policy* in general and *security policy* specifically. In the context of this paper we use the concepts of the eXtensible Access Control Markup Language (XACML [13]) to express a security policy, but for readability we use a much simpler syntax.

We consider three *roles* (classes of collaboration partners with a uniform security policy) in this scenario namely *trustworthy clients* (TC), *observers* (OB) and a *manager* (M) representing the collaboration infrastructure. There is only one role player for the manager but an unspecified number of role players for the two types of clients. The set of *subjects* is defined by $subject = \{trustworthy\ client, observer, manager\}$. For our collaboration scenario we now assume that a group of trustworthy clients hold a session. The session can be in state *public* (*pub*) or *confidential* (*conf*). The set of possible session states is thus defined by $s\_state = \{pub, conf\}$. The initial session state is *pub*. We furthermore assume that the set of possible *actions* is defined by $action = \{join, leave, close, open\}$ and that the following policy rules govern the session.

$rule_1$ When the session is in state *pub*, then observers are permitted to *join*.
$rule_2$ Observers are permitted to *leave* at any time.
$rule_3$ When no observers participate in the session, then the manager can *close* the session (change state to *conf*).
$rule_4$ The manager can *open* the session (change state to *pub*) at any time.

To be able to decide whether observers are currently participating in a session, we furthermore use a counter $o\_count \in \mathbb{N}_0$ for the current count of observers in the session. The initial value of *o_count* is 0. We don't consider any actions of the trustworthy clients in the model because we consider this irrelevant for the security goals.

In XACML a policy is given by a set of rules and a rule-combining algorithm. Each rule is composed of a condition, an effect, and a target. The conditions (predicates on attributes of subject, resource, action) associated with a policy rule specify when the policy rule is applicable. If the condition returns *False*, the rule returns *NotApplicable*. If the condition returns *True*, the value of the effect element (*Permit* or *Deny*) is returned.

For better readability we use an abbreviated syntax in this paper and define the rules from our example now by

$$rule_x : subject \times s\_state \times action \times o\_count \rightarrow \{permit, deny, not\_applicable\}.$$

$$rule_1(s,a,z,c) = \begin{cases} permit \mid & s = observer \ \wedge \ a = join \ \wedge \ z = pub \\ not\_applicable \mid & else \end{cases}$$

$$rule_2(s,a,z,c) = \begin{cases} permit \mid & s = observer \ \wedge \ a = leave \\ not\_applicable \mid & else \end{cases}$$

$$rule_3(s,a,z,c) = \begin{cases} permit \mid & s = manager \ \wedge \ a = close \ \wedge \ c = 0 \\ not\_applicable \mid & else \end{cases}$$

$$rule_4(s,a,z,c) = \begin{cases} permit \mid & s = manager \ \wedge \ a = open \ \wedge \ z = conf \\ not\_applicable \mid & else \end{cases}$$

The rule-combining algorithm we use to derive the policy result from the given rules is the permit-overrides algorithm, if a single permit result is encountered, then the combined result is permit. So we define the policy for our example now by

$$policy : subject \times action \times s\_state \times o\_count \rightarrow \{permit, deny\}.$$

$$policy(s,a,z,c) = \begin{cases} permit \mid & rule_1(s,a,z,c) = permit \ \vee \\ & rule_2(s,a,z,c) = permit \ \vee \\ & rule_3(s,a,z,c) = permit \ \vee \\ & rule_4(s,a,z,c) = permit \\ deny \mid & else \end{cases}$$

Generally, security policies have to guarantee certain security properties of a system and moreover they must not prevent the system from working.

In our example we define the following security properties:

– the collaboration is in state *conf* only if no observer is present (security), and
– always eventually state changes between *pub* and *conf* are possible (liveness).

These properties are formally verified in Sect. 5.

4 Verification of System Properties

Our operational finite state model of the behaviour of the given collaboration scenario is based on *Asynchronous Product Automata (APA)*, a flexible operational specification concept for cooperating systems [5]. An APA consists of a family of so called *elementary automata* communicating by common components of their state (shared memory).

4.1 Formal Modelling Technique

We now introduce the formal modelling techniques used, and illustrate the usage by our collaboration example.

Definition 1. *An* Asynchronous Product Automaton *consists of*

- *a family of* state sets $Z_s, s \in \mathbb{S}$,
- *a family of* elementary automata $(\Phi_e, \Delta_e), e \in \mathbb{E}$ *and*
- *a* neighbourhood relation $N : \mathbb{E} \to \mathcal{P}(\mathbb{S})$

$\mathbb{S}$ *and* $\mathbb{E}$ *are index sets with the names of state components and of elementary automata and* $\mathcal{P}(\mathbb{S})$ *is the power set of* $\mathbb{S}$.

For each elementary automaton (Φ_e, Δ_e) *with* Alphabet Φ_e*, its* state transition relation *is* $\Delta_e \subseteq \bigtimes_{s\in N(e)}(Z_s) \times \Phi_e \times \bigtimes_{s\in N(e)}(Z_s)$*. For each element of* Φ_e *the state transition relation* Δ_e *defines state transitions that change only the state components in* $N(e)$*. An APA's (global)* states *are elements of* $\bigtimes_{s\in\mathbb{S}}(Z_s)$*. To avoid pathological cases it is generally assumed that* $\mathbb{S} = \bigcup_{e\in\mathbb{E}}(N(e))$ *and* $N(e) \neq \emptyset$ *for all* $e \in \mathbb{E}$*. Each APA has one* initial state $q_0 = (q_{0s})_{s\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$*. In total, an APA* $\mathbb{A}$ *is defined by*

$$\mathbb{A} = ((Z_s)_{s\in\mathbb{S}}, (\Phi_e, \Delta_e)_{e\in\mathbb{E}}, N, s_0)$$

Finite state model of the collaboration scenario. The collaboration model described in Sect. 3 is specified for the proposed analysis method using the following *APA state components*:

$\mathbb{S} = \{s\_state, o\_count\}$ with $Z_{s\_state} = \{pub, conf\}$ and $Z_{o\_count} = \mathbb{N}_0$, $q_0 = (q_{0s\_state}, q_{0o\_count}) = (pub, 0)$.

The set of *elementary automata* $\mathbb{E} = \{OB\_join, OB\_leave, M\_conf, M\_pub\}$ represents the possible actions that the subjects (manager and observers) can take. These specifications are represented in the data structures and initial configuration of the state components in the APA model. The lines in Fig. 1 between state components and elementary automata represent the neighbourhood relation.

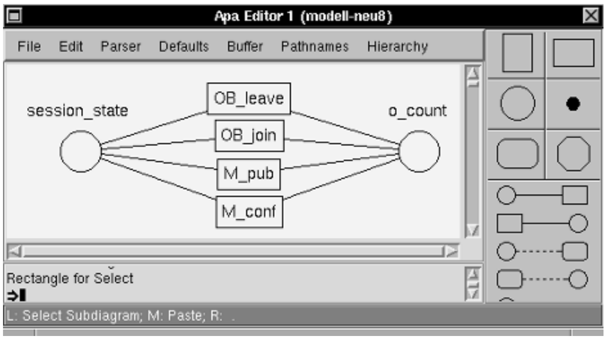
From Fig. 1 we conclude that $N(e) = \mathbb{S}$ for each $e \in \mathbb{E}$.
For each $e \in \mathbb{E}$ we choose $\Phi_e = \{\#\}$. Therefore we can omit the middle component of the state transition relation Δ_e.

Using the abbreviation $state = \{pub, conf\}$, it holds $\Delta_e \subset (state \times \mathbb{N}_0) \times (state \times \mathbb{N}_0)$ for each $e \in \mathbb{E}$.

In detail:

$$\begin{aligned}
\Delta_{OB_{leave}} &= \{((x,y),(x,y-1)) \in (state \times \mathbb{N}_0) \times (state \times \mathbb{N}_0) \mid \\
&\quad y > 0 \wedge policy(observer, leave, x, y) = permit\} \\
\Delta_{OB_{join}} &= \{((x,y),(x,y+1)) \in (state \times \mathbb{N}_0) \times (state \times \mathbb{N}_0) \mid \\
&\quad y > maxOB \wedge policy(observer, join, x, y) = permit\} \\
\Delta_{M_{conf}} &= \{((x,y),(conf,y)) \in (state \times \mathbb{N}_0) \times (state \times \mathbb{N}_0) \mid \\
&\quad policy(manager, close, x, y) = permit\} \\
\Delta_{M_{pub}} &= \{((x,y),(pub,y)) \in (state \times \mathbb{N}_0) \times (state \times \mathbb{N}_0) \mid \\
&\quad policy(manager, open, x, y) = permit\}
\end{aligned}$$

Note that this APA is parameterised by $maxOB \in \mathbb{N}_0$.



OB_join - observer *join* collaboration,
OB_leave - observer *leave* collaboration,
M_pub - manager changes state to *pub*,
M_conf - manager changes state to *conf*

Fig. 1. Collaboration model

Definition 2. *An elementary automaton (Φ_e, Δ_e) is* activated *in a state $q = (q_s)_{s\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$ as to an* interpretation *$i \in \Phi_e$, if there are $(p_s)_{s\in N(e)} \in \bigtimes_{s\in N(e)}(Z_s)$ with $((q_s)_{s\in N(e)}, i, (p_s)_{s\in N(e)}) \in \Delta_e$. An activated elementary automaton (Φ_e, Δ_e) can execute a state transition and produce a* successor state *$p = (p_s)_{s\in\mathbb{S}} \in \bigtimes_{s\in\mathbb{S}}(Z_s)$, if $q_r = p_r$ for $r \in \mathbb{S} \setminus N(e)$ and $(q_s)_{s\in N(e)}, i, (p_s)_{s\in N(e)} \in \Delta_e$. The corresponding state transition is $(q, (e, i), p)$.*

For example $((conf, 0), (M\_pub, \#), (pub, 0))$ is a state transition of our example. As mentioned above, we omit # in the sequel.

Definition 3. *The behaviour of an APA is represented by all possible coherent sequences of state transitions starting with initial state q_0. The sequence $(q_0, (e_1, i_1), q_1)\ (q_1, (e_2, i_2), q_2)\ (q_2, (e_3, i_3), q_3) \ldots\ (q_{n-1}, (e_n, i_n), q_n)$ with $i_k \in \Phi_{e_k}$ represents one possible sequence of actions of an APA. q_n is called the* goal *of this action sequence.*

State transitions $(p, (e, i), q)$ may be interpreted as labelled edges of a directed graph whose nodes are the states of an APA: $(p, (e, i), q)$ is the edge leading from p to q and labelled by (e, i). The subgraph reachable from the node q_0 is called the reachability graph *of an APA.*

Let $\mathbb{Q}$ denote the set of all states $q \in \bigtimes_{s\in\mathbb{S}}(Z_s)$ that are reachable from the initial state q_0 and let Ψ denote the set of all state transitions with the first component in $\mathbb{Q}$.

The set $L \subset \Psi^*$ of all action sequences with initial state q_0 including the empty sequence ϵ denotes the action language *of the corresponding APA. The action language is prefix closed. By definition q_0 is the goal of ϵ.*

The *reachability graph* of the example depends on the parameter $maxOB \in \mathbb{N}_0$; its set of nodes is given by $\mathbb{Q}_{maxOB}$ and its set of edges is Ψ_{maxOB}.

It is $\mathbb{Q}_{maxOB} \subset \{pub, conf\} \times \mathbb{N}_0$ and $\Psi_{maxOB} \subset \mathbb{Q}_{maxOB} \times \mathbb{E} \times \mathbb{Q}_{maxOB}$. The reachability graph for $maxOB = 0$ is shown in Fig. 2. The reachability graph for $maxOB = 1$ is depicted by the solid lines in Fig. 3, whereas the dashed lines in the same figure show the reachability graph for $maxOB = 2$.

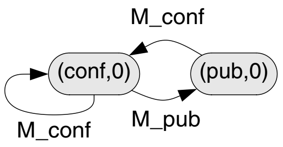


Fig. 2. Reachability graph for $maxOB = 0$

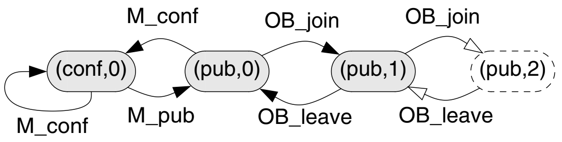


Fig. 3. Reachability graphs for $maxOB = 1$ (solid lines) and $maxOB = 2$ (dashed)

For example $((pub, 0), M\_conf, (conf, 0))((conf, 0), M\_pub, (pub, 0))$ is an element of the action language.

4.2 Abstraction Based Verification Concept

Now behaviour abstraction of an APA can be formalised by language homomorphisms, more precisely by alphabetic language homomorphisms $h : \Sigma^* \to \Sigma'^*$.

By these homomorphisms certain transitions are ignored and others are renamed, which may have the effect, that different transitions are identified with one another. A mapping $h : \Sigma^* \to \Sigma'^*$ is called a language homomorphism if $h(\epsilon) = \epsilon$ and $h(yz) = h(y)h(z)$ for each $y, z \in \Sigma^*$. It is called alphabetic, if $h(\Sigma) \subset \Sigma' \cup \{\epsilon\}$.

It is now the question, whether, by investigating an abstract behaviour, we may verify the correctness of the underlying concrete behaviour. Generally under abstraction the problem occurs, that an incorrect subbehaviour can be hidden by a correct one. We will answer this question positively, requiring a restriction to the permitted abstraction techniques [1].

As it is well known, system properties are divided into two types: safety (what happens is not wrong) and liveness properties (eventually something desired happens, e.g. availability) [14].

On account of liveness aspects system properties are formalised by ω-languages (sets of infinite long words). So to investigate satisfaction of properties "infinite system behaviour" has to be considered. This is formalised by so called Eilenberg limits of action languages (more precisely: by Eilenberg limits of modified action languages where maximal words are continued by an unbounded repetition of a dummy action) [15].

The usual concept of linear satisfaction of properties (each infinite run of the system satisfies the property) is not suitable in this context because no fairness constraints are considered. We put a very abstract notion of fairness into the satisfaction relation for properties, which considers that independent of a finitely long computation of a system certain desired events may occur eventually. To formalise such "possibility properties", which are of interest when considering what we call cooperating systems, the notion of approximate satisfaction of properties is defined in [15].

Definition 4. *A system* approximately satisfies *a property if and only if each finite behaviour can be continued to an infinite behaviour, which satisfies the property.*

For safety properties linear satisfaction and approximate satisfaction are equivalent [15]. To deduce approximately satisfied properties of a specification from properties of its abstract behaviour an additional property of abstractions called simplicity of homomorphisms on an action language [16] is required. Simplicity of homomorphisms is a very technical condition concerning the possible continuations of finite behaviours.

For regular languages simplicity is decidable. In [16] a sufficient condition based on the strongly connected components of corresponding automata is given, which easily can be checked. Especially: If the automaton or reachability graph is strongly connected, then each homomorphism is simple.

The following theorem [15] shows that approximate satisfaction of properties and simplicity of homomorphisms exactly fit together for verifying cooperating systems.

Theorem 1. *Simple homomorphisms define exactly the class of such abstractions, for which holds that each property is approximately satisfied by the abstract behaviour if and only if the "corresponding" property is approximately satisfied by the concrete behaviour of the system.*

Formally, the "corresponding" property is expressed by the inverse image of the abstract property with respect to the homomorphism.

In the example of this paper the desired security properties are safety and liveness properties. Generally there are more complex security properties. In [17] and [18] it has been shown how authenticity, provability and confidentiality are also treated in terms of prefix closed languages and property preserving language homomorphisms.

4.3 Verification Tool

The *Simple Homomorphism (SH) verification tool* [5] is used to analyse the collaboration model for different concrete values of *maxOB*. It has been developed at the *Fraunhofer-Institute for Secure Information Technology.* The SH verification tool provides components for the complete cycle from formal specification to exhaustive validation as well as visualisation and inspection of computed reachability graphs and minimal automata. The applied specification method based on *Asynchronous Product Automata (APA)* is supported by this tool. The tool manages the components of the model, allows to select alternative parts of the specification and automatically *glues* together the selected components to generate a combined model of the APA specification. After an initial state is selected, the reachability graph is automatically computed by the SH verification tool.

The tool provides an editor to define homomorphisms on action languages, it computes corresponding minimal automata [19] for the homomorphic images and checks simplicity of the homomorphisms.

Model checking. If it is required to inspect some or all paths of the graph to check for the violation of a security property, as it is usually the case for liveness properties, then the tool's temporal logic component can be used. Temporal logic formulae can also be checked on the abstract behaviour (under a simple homomorphism). The method for checking approximate satisfaction of properties fits exactly to the built-in simple homomorphism check [5].

The SH verification tool successfully has been applied in several security projects such as Valikrypt (`http://www.bsi.bund.de/fachthem/valikrypt/`) and CASENET[1].

5 Verification of the Collaboration Scenario

An outline of our verification concept for parameterised models, exemplary realised for the collaboration scenario, is given in Fig. 4.

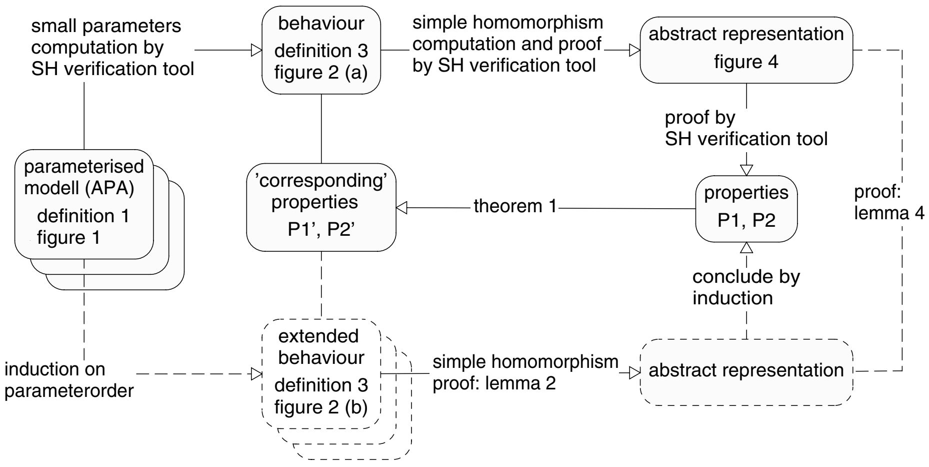

small parameters computation by SH verification tool
behaviour definition 3 figure 2 (a)
simple homomorphism computation and proof by SH verification tool
abstract representation figure 4
proof by SH verification tool
parameterised modell (APA) definition 1 figure 1
'corresponding' properties P1', P2'
theorem 1
properties P1, P2
proof: lemma 4
conclude by induction
extended behaviour definition 3 figure 2 (b)
induction on parameterorder
simple homomorphism proof: lemma 2
abstract representation

Fig. 4. Verification concept for parameterised APA

The abstraction based verification concept introduced in Sect. 4.2 and the tool support described in Sect. 4.3 cover the part marked by solid lines in Fig. 4 whereas we now prove the components marked by dashed lines.

[1] The EU project CASENET (`http://www.casenet-eu.org/`) has provided a tool-supported framework for the systematic specification, design and analysis of e-commerce and e-government transactions to produce protocols with proven security properties, and to assist in code generation for these protocols.

Using the graphs of Fig. 2 and Fig. 3 as induction base we will now prove Lemma 1 below by induction on $maxOB$. We use the abbreviations
T_{mjoin} for $((pub, maxOB), OB\_join, (pub, maxOB+1))$ and
T_{mleave} for $((pub, maxOB+1), OB\_leave, (pub, maxOB))$.

Lemma 1. *(a)* $\mathbb{Q}_{maxOB} = \{(pub, i) | 0 \le i \le maxOB\} \cup \{(conf, 0)\}$
(b) $\Psi_{maxOB+1} = \Psi_{maxOB} \,\dot\cup\, \{T_{mjoin}, T_{mleave}\}$

Proof. Figure 3 shows the reachability graph with $maxOB = 1$. Together with Fig. 2, Fig. 3 proves the induction base.
Induction step.
By inspection of the 4 elementary automata we get: $\Psi_{maxOB} \subset \Psi_{maxOB+1}$.
Starting from the nodes in $\mathbb{Q}_{maxOB}$ from $maxOB+1$ only the additional transitions T_{mjoin} and T_{mleave} are possible.

$\square$

It follows by induction:

Lemma 2. *For each $maxOB \in \mathbb{N}_0$ the corresponding reachability graph is finite and strongly connected.*

Let $L_{maxOB} \subset \Psi^*_{maxOB}$ denote the *action language*, then using Lemma 1(b) we can derive

Lemma 3. *(a)* $L_{maxOB} \subset L_{maxOB+1}$ *and*
(b) for each $u \in L_{maxOB+1}$*:* $h(u) \in L_{maxOB}$ *with the homomorphism*
$h : \Psi^*_{maxOB+1} \to \Psi^*_{maxOB}$
with $h(T_{mjoin}) = \varepsilon = h(T_{mleave})$ *and* $h(x) = x$ *for* $x \in \Psi_{maxOB}$
(c) The goal of u is identical to the goal of $h(u)$ or the goal of u is $(pub, maxOB+1)$ and the goal of $h(u)$ is $(pub, maxOB)$.

Proof of Lemma 3 (b) by induction on the length of u.
Induction base. Lemma 3(b) is true for $u = \varepsilon$. Note that by definition the goal of the empty transition sequence is equal to the initial state of the APA.

Induction step. Consider $ua \in L_{maxOB+1}$ with $a \in \Psi_{maxOB+1}$. From induction hypothesis there are 2 different cases:

Case 1. The goal of u is equal to the goal of $h(u)$ and therefore an element of $\mathbb{Q}_{maxOB}$.
Therefore $a \in \Psi_{maxOB} \cup \{T_{mjoin}\}$.
For $a \in \Psi_{maxOB}$ holds: $h(ua) = h(u)h(a) = h(u)a$
Therefore from induction hypothesis $h(ua) \in L_{maxOB}$ and goals of ua and $h(ua)$ are equal.
For $a = T_{mjoin}$ the goal of u and therefore also the goal of $h(u)$ is $(pub, maxOB)$.
Now it holds that $h(ua) = h(u)h(a) = h(u)$.
From induction hypothesis we get that $h(ua) \in L_{maxOB}$ and goal of ua is $(pub, maxOB+1)$ and goal of $h(ua)$ is $(pub, maxOB)$.

Case 2. The goal of u is $(pub, maxOB+1)$ and goal of $h(u)$ is $(pub, maxOB)$.

Then: $a = T_{mleave}$
And so:
$h(ua) = h(u)h(a) = h(u) \in L_{maxOB}$ and ua and also $h(ua)$ have the same goal, namely $(pub, maxOB)$. □

Now from Lemma 3 (a) we get $L_{maxOB} = h(L_{maxOB}) \subset h(L_{maxOB+1})$ and from 3 (b) we get $h(L_{maxOB+1}) \subset L_{maxOB}$ together $L_{maxOB} = h(L_{maxOB+1})$. For each homomorphism $f : \Psi^*_{maxOB+1} \to \Sigma'^*$ with $f(T_{mjoin}) = \varepsilon = f(T_{mleave})$ it holds that: $f(L_{maxOB+1}) = f(h(L_{maxOB+1})) = f(L_{maxOB})$ and so:

Lemma 4. *With the assumptions above holds:* $f(L_{maxOB+1}) = f(L_{maxOB})$

5.1 Proving Security and Liveness of the Collaboration Example

To consider our example's correctness we have to observe the state changes between pub and $conf$. So we define an appropriate homomorphism
$c : \Psi^*_{maxOB} \to \Psi^*_{maxOB}$ by
$c(((x_1, x_2), e, (y_1, y_2))) = ((x_1, x_2), e, (y_1, y_2))$ if $x_1 \neq y_1$, and
$c(((x_1, x_2), e, (y_1, y_2))) = \varepsilon$ if $x_1 = y_1$.

This homomorphism c fulfils the condition of Lemma 4 and therefore we get $c(L_{maxOB+1}) = c(L_{maxOB})$.

This implies $c(L_{maxOB}) = c(L_0)$ for each $maxOB \in \mathbb{N}_0$.

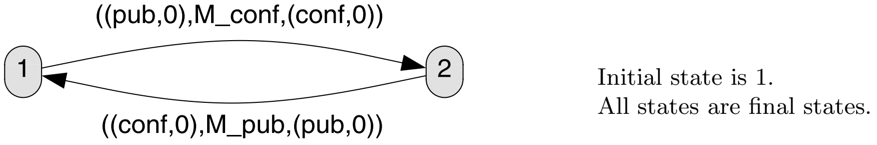


Fig. 5. Minimal automaton of $c(L_0)$

It is easy to see, that the automaton of Fig. 5 is the minimal automaton of $c(L_0)$.

This automaton shows that the collaboration is in state $conf$ only if no observer is present (P1). Moreover always state changes between pub and $conf$ are possible (P2).

By Lemma 2 c is simple on each L_{maxOB} and therefore (Theorem 1) corresponding properties P1' and P2' hold for each concrete behaviour L_{maxOB}. In content P1' is the same as P1. P2' is the property that always eventually state changes between pub and $conf$ are possible. The difference between P2 and P2' is caused by actions of the concrete behaviour which are mapped to ε by the homomorphism c. P1' and P2' are the desired properties of the collaboration as formulated in Sect. 3.

6 Conclusions and Future Work

Based on property preserving abstractions (simple homomorphisms) we combined our tool supported finite state methods with induction proofs to verify security and liveness properties of a parameterised system.

We have shown how abstractions serve as a framework for individual proofs of problem specific security properties. So our results are no contradictions to well known undecidability properties of general security models e.g. Harrison-Ruzzo-Ullman.

This paper focussed on properties which are independent of concrete parameter values. Considering parameterised abstract behaviours we will extend our method to verify parameter dependent properties. The induction proofs in this paper are "handmade". So it would be desirable to support such proofs by a theorem prover. For that purpose our system specifications based on parameterised APA have to be represented in a corresponding theorem prover.

Acknowledgements. We would like to thank Carsten Kunz, Carsten Rudolph and Björn Steinemann for cooperation on early versions of this work and many productive discussions on the subject.

References

1. Ochsenschläger, P., Repp, J., Rieke, R.: Abstraction and composition – a verification method for co-operating systems. Journal of Experimental and Theoretical Artificial Intelligence 12, 447–459 (2000) Copyright: ©2000, American Association for Artificial Intelligence `www.aaai.org` All rights reserved
2. Guttman, J.D., Herzog, A.L., Ramsdell, J.D.: Information flow in operating systems: Eager formal methods. In: IFIP WG 1.7 Workshop on Issues in the Theory of Security (2003)
3. Guttman, J.D., Herzog, A.L.: Rigorous automated network security management. International Journal of Information Security 4(1-2), 29–48 (2005)
4. Rieke, R.: Modelling and Analysing Network Security Policies in a Given Vulnerability Setting. In: Lopez, J. (ed.) CRITIS 2006. LNCS, vol. 4347, pp. 67–78. © Springer, Heidelberg (2006)
5. Ochsenschläger, P., Repp, J., Rieke, R., Nitsche, U.: The SH-Verification Tool Abstraction-Based Verification of Co-operating Systems. Formal Aspects of Computing, The International Journal of Formal Method 11, 1–24 (1999)
6. Ip, C.N., Dill, D.L.: Verifying Systems with Replicated Components in Murφ. Formal Methods in System Design 14(3), 273–310 (1999)
7. Derepas, F., Gastin, P.: Model checking systems of replicated processes with spin. In: Dwyer, M.B. (ed.) Model Checking Software. LNCS, vol. 2057, pp. 235–251. Springer, Heidelberg (2001)
8. Lakhnech, Y., Bensalem, S., Berezin, S., Owre, S.: Incremental verification by abstraction. In: Margaria, T., Yi, W. (eds.) ETAPS 2001 and TACAS 2001. LNCS, vol. 2031, pp. 98–112. Springer, Heidelberg (2001)
9. Basu, S., Ramakrishnan, C.R.: Compositional analysis for verification of parameterized systems. Theor. Comput. Sci. 354(2), 211–229 (2006)

10. Milner, R.: Communication and Concurrency. International Series in Computer Science. Prentice-Hall, Englewood Cliffs (1989)
11. Bradfield, J., Stirling, C.: Modal logics and mu-calculi: an introduction (2001)
12. Uribe, T.E.: Combinations of model checking and theorem proving. In: Kirchner, H. (ed.) Frontiers of Combining Systems. LNCS, vol. 1794, pp. 151–170. Springer, Heidelberg (2000)
13. Moses, T.: eXtensible Access Control Markup Language (XACML), Version 2.0. Technical report, OASIS Standard (2005)
14. Alpern, B., Schneider, F.B.: Defining liveness. Information Processing Letters 21(4), 181–185 (1985)
15. Nitsche, U., Ochsenschläger, P.: Approximately satisfied properties of systems and simple language homomorphisms. Information Processing Letters 60, 201–206 (1996)
16. Ochsenschläger, P.: Verification of cooperating systems by simple homomorphisms using the product net machine. In: Desel, J., Oberweis, A., Reisig, W., (eds.) Workshop: Algorithmen und Werkzeuge für Petrinetze, Humboldt Universität Berlin, pp. 48–53 (1994)
17. Gürgens, S., Ochsenschläger, P., Rudolph, C.: On a formal framework for security properties. International Computer Standards & Interface Journal (CSI), Special issue on formal methods, techniques and tools for secure and reliable applications (2004)
18. Gürgens, S., Ochsenschläger, P., Rudolph, C.: Abstractions preserving parameter confidentiality. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 418–437. Copyright: ©Springer, Heidelberg (2005)
19. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic Press, New York (1974)

Algebraic Models to Detect and Solve Policy Conflicts*

Cataldo Basile, Alberto Cappadonia, and Antonio Lioy

Dipartimento di Automatica e Informatica
Politecnico di Torino
{cataldo.basile,alberto.cappadonia,lioy}@polito.it

Abstract. The management of security for large and complex environments still represents an open problem and the policy-based systems are certainly one of the most innovative and effective solution to this problem. The policy, that at low level is expressed by sets of rules, becomes crucial for the consistency of the systems to be protected and it is necessary to check it for correctness. This paper presents a set-based model of rules that permits the static conflict detection and an axiomatic model of conflict resolution leading to semi-lattices theory to solve inconsistencies. We proved the effectiveness of the theory implementing an extensible tool supporting security officers in creating rules by providing an easy environment to identify the conflicts and to use manual as well as automatic resolution strategies.

Keywords: security policy model; policy conflicts detection; policy conflicts resolution; firewall rules analysis; policy specification.

1 Introduction

The management of security for large and complex environments still represents an open problem and the policy-based systems are certainly one of the most innovative and effective solutions to this problem. Main characteristics of policy-based systems are depicted in the document called “Terminology for policy-based management” [1] in which the term policy is defined as: “a definite goal, course or method of action to guide and determine present and future decisions. ‘Policies’ are implemented or executed within a particular context, such as the policies defined within a business unit”, and as “a set of rules to administer, manage, and control access to network resources”.

The policy becomes crucial for the consistency of the systems to be protected and it is necessary to check it for correctness. The best approach consists in assisting security officers in solving conflicts during the specification, eliminating the inconsistencies before deploying the configurations. The problem is that tools enabling the analysis of the policy are still missing or of limited usage. The reasons of this lack can be found in the absence of mathematical models that can describe in a precise manner the *anomalies* and *conflicts* that can happen between two or more rules. If some work can be found in the field of the *detection* [2], there is no available study trying to understand the resolution process and its “natural” characteristics, in order to correctly and coherently model

* This work is part of the POSITIF and DESEREC projects, funded by the European Commission under contracts IST 002314 and 026600.

it. These studies can help to derive automatic processes to solve inconsistencies even without the human intervention strongly reducing the specification time. For this, we consider our work an important progress in the field of policy conflicts analysis, providing a set-based model of rules to allow the detection of the conflict and an axiomatic model of conflict resolution leading to semi-lattices theory to solve inconsistencies. Previous work in modeling detection and resolution [9] is extended to include not only the Action-Based resolution strategy but all the techniques already described in literature, defining the criteria for an acceptable resolution method. Additionally, theoretical considerations are presented to help in developing a tool to assist the administrator in identifying the conflicts and resolving them. The tool is able to aggregate the rules using semi-lattices instead of simple ordered lists of rules. In fact, these algebraic structures permit an accurate definition of rules relationships for conflict purposes.

The paper is organized as follows: in Sec. 2 it is presented the background, some related works and the motivations; in Sec. 3 it is described the model for the conflict detection and in Sec. 4 the model for the conflict resolution; in Sec. 5 it is introduced the tool and its main characteristics; finally, in Sec. 6 we present our conclusions and we briefly sketch plans for future work.

2 Motivation and Related Work

Conflicts detection and resolution is one of the most interesting research field in the security area, the choose of what action have to be performed if a conflict arise has carried to the development of several methods and techniques dealing with policy classification and verification. deal with general management of policies. In [3] policy inconsistencies in role-based management framework are classified and techniques are presented to solve them. In [4] Dunlop combines deontic logic with temporal operators. Conflicts are classified in four category and the detection is distinct in two cases: *static conflict*, incongruence found during the initialization phase and *dynamic conflicts*, potential conflict that is quite unpredictable and is the result of a run-time action. Even if it is different from our work, this model is interesting for conflicts classification.

Others work are focused on filtering rules. Al-Shaer and Hamed [2], propose a model in which a rule corresponds to a row of a firewall's table. They model rules as tuples of conditions and identify five possible relation between two rule: *completely disjoint*, *partially disjoint*, *exactly matched*, *inclusively matched* and *correlated* in function of the reciprocal relations of conditions (using standard set operation such as subset, superset and equality). Additionally, the authors, define four possible anomalies between two rules related with the priority of the rules in the firewalls table: *Shadowing*, *Correlation*, *Generalization*, *Redundancy*. This approach is useful to detect and manually resolve policy (using trees) conflicts but is bounded with the priority of the rules. Furthermore, we identified in this approach a main incongruence in their mathematical model. The categories they identified are not completely describing all the possible cases because they only consider the relation of subset, superset, equality and different (i.e., none of previous) but, to achieve completeness it is also necessary to split the 'different' case in *none of previous but intersecting* and *none of previous and non-intersecting*. Since we model rules conditions as sets (using set operations to understand their reciprocal relations), this issue is naturally solved. Additionally, semi-lattices are more compact and effective to convey resolution information.

Bandara et al. [7] present an interesting application of augmented logic to firewall analysis able to cover at least all the cases already analyzed by Al-Shaer and Hamed. Uribe and Cheung [8] present a constrain logic based analysis of IDS and firewall in a distributed scenario. Main ideas of this paper (e.g., the `filter()` function) can be also applied to a generalization of our approach to distributed firewalls. Mayer et al. developed the second generation of firewall analysis tools, the "Firewall Analyzer" (FA) [6]. Their tool is one of our reference points to evaluate the effectiveness and validity of our model. We theoretically proved that we can answer to many of their queries (NAT is not covered yet), and we are currently implementing them. All these papers are designed and limited to firewall analysis and they are not so easily extensible to more general cases nor focus on the general instance of conflict resolution. Nevertheless we started from the firewall examples to compare our approach in a field where concrete results are already available.

3 A Model for Conflict Detection

In this section we summarize the results in [9] to model rules for detection purposes. Since a condition in a particular *selector* (e.g., a particular field such as IP addresses) defines the subset of allowed values (i.e., for which it is evaluated to true), it is possible to state that a condition in a given selector set S, is a subset c of S. To consider different selectors (e.g., IP addresses and ports) we used Cartesian product obtaining a model in which rule conditions are hyper-rectangles (*selection conditions*). Correctness is guaranteed by the important result stating that the power set (i.e., the set of all the subsets) of a given set is a boolean algebra endowed with standard operation of intersection ($\cap$) and union ($\cup$). Introducing the actions, a rule can be represented as a function that associates a selection condition to a subset of an action set $\mathcal{A}$. For this a rule can be written as a couple selection condition-action (i.e., $r = (C, a)$). A policy "that is expressed by means of a set of rules" can be defined as a piecewise function.

Given two policy rules $r_1 = (C_1, a_1)$ and $r_2 = (C_2, a_2)$ a policy conflict occurs if $C_\cap = C_1 \cap C_2 \neq \varnothing$ and $a_1 \neq a_2$. In other words, the policy conflict happens if the selection conditions of the two rules intersect (i.e., when they can be activated simultaneously), but they do not specify the same set of actions. The only region where errors originate is $C_\cap$. Indeed, in the regions out of the intersection there are no conflicts because it is clear which actions to apply. In $C_\cap$ it is impossible to define a policy function since for each element of the domain it is needed the association to only one element of the codomain. This definition is more powerful than a classification because it relies on set properties and all the possible rules relative scenarios are naturally identified. It is worth noting that there is some cases in which different actions not necessarily conflict. In these cases, it is possible to define another action set in which previous definition apply. When some actions can be enforce simultaneously we already showed that the action set can be extended using the Cartesian product in the action set $\mathcal{A}$ [9]. In other cases, when is possible to define an equivalence relation r, we work on the quotient set $\mathcal{A}/r$ where our assumption still holds.

4 A Model for Conflict Resolution

Different methods for conflict resolution are presented in literature [10]: the *prioritization* of the rules based on the order in the database, also called "First Matching Rule" (FMR), the *Deny Take Precedence* (DTP), if the actions or policy rule more restrictive are preferred and the *Most/Least Specific Take Precedence* (MSTP/LSTP), if the policy rule with the most/least specific condition is preferred. In a previous work we already addressed the modeling of the Action-Based resolution strategy, an extension of DTP. Here we present the generalization of the semi-lattice based technique [9] .

From these examples, it is possible to derive the types of information that are used in the process of resolution: the conditions (for MSTP/LSTP); the actions (for DTP) and external data (for FMR). Therefore, the model of rules used for the detection must be extended using the external data. Introducing the set $\mathcal{D}$, called the *external data space*, rules become $r = (d, C, a) \subseteq \mathcal{D} \times \mathcal{S} \times \mathcal{A} = \mathcal{R}$. We call $\mathcal{R}$ the *rule space*.

The objective of this treatment is an axiomatic definition of the resolution process. First, we start from an empirical consideration: the resolution must be the task that permit the definition of the action to be applied when two rules conflict. By modeling the resolution as an abstract operation "$\circ$" in the rule space $\mathcal{R}$, the *binary conflict resolution method* is a function such that, given two conflicting rules $r_1, r_2 \subseteq \mathcal{R}$, with $r_1 = (d_1, C_1, a_1), r_2 = (d_2, C_2, a_2)$, it returns another rules that applies where original rules conflict, i.e.,: $\circ : \mathcal{R} \times \mathcal{R} \longrightarrow \mathcal{R}$ such that $r_1 \circ r_2 \longmapsto r_{1,2} = (d_{1,2}, C_1 \cap C_2, a_{1,2})$.

It is also possible to impose some natural restrictions to this operation. The first property required from an application describing the interactions between rules is the *associativity*, i.e., $\forall r_1, r_2, r_3 \in \mathcal{R} : r_1 \circ (r_2 \circ r_3) = (r_1 \circ r_2) \circ r_3$. This characteristic reflects the fact that the composition is done from a static point of view: conflicts identified in a particular area are resolved independently from the order in which rules are composed. Another important property is the *commutativity*, i.e., $\forall r_1, r_2 \in \mathcal{R} : r_1 \circ r_2 = r_2 \circ r_1$. Even if commutativity is the less obvious property, it always holds if it the external data space is constructed correctly (e.g., including the priorities makes the First Matching Rule strategy commutative). For completeness, we impose to the "$\circ$" operation the *idempotence*, i.e., $\forall r \in \mathcal{A} : r \circ r = r$, that is, a rule is never in conflict with itself. The algebraic structure satisfying previous axioms is a *semi-lattice* [11]. The three axioms univocally identify a well-formed resolution strategy.

Given n rules $\{r_\mathrm{k}\}_{\mathrm{i} \in \mathbb{Z}_\mathrm{n}}$ and all the composition of $M = \{r_\mathrm{I}\}_{\mathrm{I} \in 2^{\mathbb{Z}_n}}$, it easy to show that M is a semi-lattice respecting associativity, commutativity and idempotence. In general, a set of rules can be completed to a semi-lattice by adding all the compositions of rules. Clearly, the semi-lattice is built in function of the actual rules.

Semi-lattices can be viewed as *partially ordered set* and for this reason they can be represented as *cover graphs* having only one terminal node. In fact, the following binary relation on $(\mathcal{R}, \circ)$, $\leq \subseteq \mathcal{R} : r_1 \leq r_2 \iff r_1 \circ r_2 = r_2$ defines a natural order from which results that $r_1 \leq r_2$ implies that $C_1 \supseteq C_2$. Additionally, if a resolution strategy uses actions (solely or in conjunction with conditions) then it is possible to deduce a semi-lattice structure for the actions too. In fact, if $\mathcal{R}$ is a semi-lattice, let $\mathrm{r_h} = (d_\mathrm{h}, C_\mathrm{h}, a_\mathrm{h})$ and $\mathrm{r_k} = (d_\mathrm{k}, C_\mathrm{k}, a_\mathrm{k})$ be two rules such that $r_\mathrm{h} \leq r_\mathrm{k}$, we have $C_\mathrm{h} = C_\mathrm{k} \Rightarrow a_\mathrm{h} \leq a_\mathrm{k}$. If the resolution method uses the actions together with external data, it is possible to deduce a semi-lattice structure only for $\mathcal{D} \times \mathcal{A}$ (but if $\mathcal{D}$ is a semi-lattice too, even the action set is a semi-lattice). All the properties above can be used as criteria to *evaluate if a resolution strategy is suitable*. The semi-lattice assumption is not too abstract, in fact, the sequence of rules ordered by means of integers form a semi-lattice (a total ordered set is a semi-lattice) and boolean algebras (i.e., the algebraic structure of conditions) is a semi-lattice too [11]. Additionally, compared with Hamed and Al-Shaer trees used for resolution, semi-lattice are more compact and expressive.

A tool implementing this model can be optimized in many ways. In fact, even though the maximum number of rules is exponential, in the practical cases it is not necessary to include all the compositions. First, it is not necessary to include the compositions having empty selection condition (they never apply). Then, two equivalent rules (originals or compositions) do not add any new information, so one of them can be dropped. Even the order relation can be used to reduce the final number or rules. In fact, if $r_{\mathrm{h}} \leq r_{\mathrm{k}}$ then $\forall i_1, \cdots, i_{\mathrm{k}} \in \mathbb{Z}_n \setminus \{h, k\} : r_{\mathrm{h}} \circ r_{\mathrm{k}} \circ r_{\mathrm{k}_1} \circ \cdots \circ r_{\mathrm{k}_{\mathrm{k}}} \equiv r_{\mathrm{k}} \circ r_{\mathrm{k}_1} \circ \cdots \circ r_{\mathrm{k}_{\mathrm{k}}}$ and one of them can be eliminated. The last case to consider is the *shadowing*. If a rule $\mathrm{r_h}$ hides a rule $\mathrm{r_k}$, the $\mathrm{r_k}$ does not add any information. For this it is possible to eliminate $\mathrm{r_k}$ and all its compositions.

Furthermore, rules enforcing the same actions are not in conflict and, for this, they can be (often) merged to form a unique rule. Using this property, we proved that the number of rules is, in worst case, polynomial of order equals to the cardinality of the action set.

5 The Tool

The tool, written using the Java programming language for POSITIF and DESEREC projects [12,13], is composed by an articulated class hierarchy that carry out with the aim to be extensible, easily modifiable and comprehensible. It relies on the class `ConflictManagingRuleDatabase`, allowing insertion and removal of rules using a recursive function on the cover graph (linear in the number of nodes). To manage conflicts, the database uses: a *detector*, that identifies the conflicts, a *resolver*, classifying the conflicts and providing the methods to resolve them (i.e., the rule to apply where two rules conflict), and an *exporter*, able to generate the set of conflict-free rules to be enforced by the target device.

The tool provides a graphical interface that assists the user in the policy specification, permitting to analyze conflicting rules. It supports and manages different resolution strategies, that solve the conflicts building an opportune semi-lattice from inserted rules. Since all the decisions taken inside the database are done using the answers given by the detector and the resolver, the way in which rules are arranged inside the semi-lattice depends only on these classes. This also guarantees the possibility of interchangeability of all the detectors and resolvers and to extend it for non-firewall scenarios. Limited tests on a HP D7700 RG582AW showed that to insert and manage conflicts (with action-based resolution strategy and 6 different actions) for 100 randomly generated rules (and for this, more correlated than real world rules) the tools needs less than 2 seconds. The average number of nodes in the graph is about 250 and after the exporter reduces the average number of rules to 63. We planned to test our tool with concrete examples from which we expect more interesting results.

6 Conclusions and Future Work

In this paper we presented a set-based model of policy rules that permits the conflict detection and a general model for resolution based on semi-lattice theory. This mathematical definition is an important and innovative aspect in the field of policy analysis. In fact, many policy-related problems are not yet addressed because of the lack of effective mathematical models. We also presented an extensible tool providing a graphical environment to assist the administrators towards the conflict management, also permitting to export sets of conflict-free rules thus reducing effort and specification mistakes.

The tool is presented for filtering rules but it can be easily extended to IPsec channels rules and web applications protected with SSL. Also the definition of new resolution strategies is one of the most interesting research field in which we believe good results can be achieved. and we will provide export facilities to vendor solutions. The model requires a very limited effort to be extended to distributed firewalls (i.e., to model what is the policy enforced by a chain of firewalls) and it can rely on the same tool by simply including topological information that we can easily get from other results of POSITIF project. Since the theory permits to identify sets of rules enforcing the same policy, we already planned to test it to optimize packet filtering performances using statistical information.

References

1. Westerinen et al.: Terminology for Policy-Based Management. RFC-3198
2. Al-Shaer, E., Hamed, H.: Firewall Policy Advisor for Anomaly Discovery and Rule Editing. In: IFIP/IEEE Eighth International Symposium on Integrated Network Management (2003)
3. Lupu, E.C., Sloman, M.S.: Conflicts in Policy-Based Distributed Systems Management. IEEE Trans. on Software Engineering 25(6) (1999)
4. Dunlop, N., Indulska, J., Raymond, K.: Methods for Conflict Resolution in Policy-Based Management Systems. In: EDOC 2003 (2003)
5. Al-Shaer, E., Hamed, H.: Modeling and management of firewall policies. IEEE Trans. Netw. Serv. Manage. 1(1) (2004)
6. Mayer, A., Wool, A., Ziskind, E.: Offline Firewall Analysis. Int. J. Inf. Secur. 5(3) (2006)
7. Bandara, A., Kakas, A., Lupu, E., Russo, A.: Using Argumentation Logic for Firewall Policy Specification and Analysis. In: 17th IFIP/IEEE Distributed Systems: Operations and Management (DSOM) (2006)
8. Uribe, T.E., Cheung, S.: Automatic Analysis of Firewall and Network Intrusion Detection System Configurations. In: Proceedings of the 2004 ACM Workshop on Formal Methods in Security Engineering, Washington, D.C, ACM Press, New York (2004)
9. Basile, C., Lioy, A.: Towards an Algebraic Approach to Solve Policy Conflicts. Workshop on Logical Foundations of an Adaptive Security Infrastructure (WOLFASI) (2004)
10. Castano, S., Fugini, M., Martella, G., Samarati, P.: Database Security. Addison Wesley, Reading (1994)
11. Szasz, G.: Théorie des treillis. Dunod Éditeur (1971)
12. POSITIF – Policy-based Security Tools and Framework. EU contract IST-2002-002314, `http://www.positif.org`
13. DESEREC – DEpendability and Security by Enhanced REConfigurability. EU contract IST-2004-026600, `http://www.deserec.eu`

Event Calculus Based Checking of Filtering Policies

Artem Tishkov, Ekaterina Sidelnikova, and Igor Kotenko

Computer Security Research Group, St. Petersburg Institute for Informatics and Automation
39, 14 Liniya, St.-Petersburg, 199178, Russia
{avt,sidelnikova,kotenko}@iias.spb.su

Abstract. The paper considers the approach to filtering policy verification. We model potential network traffic with Event Calculus and use abductive proof procedure to detect firewall filtering anomalies in dynamical way. Generally, our approach allows separating network behavior description from security inconsistency definition and thus building flexible and scalable framework for filtering policy verification.

Keywords: Security policy, policy verification, filtering, Event Calculus.

1 Introduction

One of actual problems for filtering policy management is the verification of policy specifications. It is a common case when the access control list (ACL) of a firewall contains rules which conditions are intersected. In this case, for example, a system administrator defines a denial rule (for large range of source and destination addresses, ports, protocol types) and a set of allowing rules (that give access to particular addresses, ports and protocol types). However other types of rule intersection can appear, for instance, when one rule hides another rule while the latter one is actually more important. Such *anomalies* must be opportunely detected and resolved, otherwise the network becomes unmanageable. It is especially important for large corporate networks including plenty of firewalls.

To solve this problem, effective automated policy verification systems should be created and used. There are a lot of relevant works in this research area. Al-Shaer at al. [2] present the classification of firewall anomalies, define the relations between filtering rules and determine different filtering anomalies. In [1] Firewall Policy Advisor (FPA) software is presented; it serves for detecting the filtering anomalies and editing the policy. FPA covers different inter- and intra-firewall anomalies, but not aimed to construct network model and apply it to the security policy. Bandara at al. [3] use the argumentation logic reasoning and abductive proof for detection and resolution of firewall anomalies. Application of Event Calculus to authorization and obligation policies is considered in [4]. The argumentation and abduction verification tool is described in [6]. *Our approach* to verification consists in modeling the system behavior using the axiomatics of Event Calculus (EC) [7]. Having constructed domain-dependent EC axioms and formulas that express filtering anomalies, the abductive proof procedure [5] results in a scenario (as the sequence of events) that lead the firewall to ignore rules which conditions are satisfied. If the scenario is found, participated rules are analyzed and one of resolution strategies is applied. In comparison to the approaches mentioned our security verification approach and software developed takes into account the descriptions of the network, security policies, and anomalies from different sources and thus gives more flexible and scalable solution. The paper is structured as follows. *Section* 2 considers filtering policy anomalies. Event Calculus-based filtering policy definition and anomaly detection example are represented in *section* 3. *Section 4* describes the verification software architecture, user interface and experiments. *Conclusion* presents the paper results and future research directions.

2 Filtering Policy Anomalies

The filtering policy rule allows or denies the network traffic for given source and destination addresses, ports, and protocol types. Every such rule is ordered and corresponds to a row in the firewall access control list.

According to [2] we use the following *relations for the pairs of rules*: completely disjoint, exact matching, inclusively matching, partially matching, correlated. Depending on a type of relation, the following *filtering anomalies* are defined:

(1) A rule is *shadowed* by another rule if the latter one has higher priority, matches all the packets of the former one, and performs different allow/deny action;

(2) A rule is a *generalization* of another rule if they have different allow/deny actions, the latter rule has higher priority, and the former one matches all the packets of the latter one;

(3) A *redundant* rule performs the same action on the same packets as another rule such that if the redundant rule is removed, the security policy will not be affected;

(4) Two rules are *correlated* if they have different filtering actions, and the first rule matches the packets that match the second rule and the second rule matches the packets that match the first rule.

3 Event Calculus and Axiomatization

The Event Calculus (EC) [7] formalizes the common sense principle of inertia: "normally, nothing changes". It states that *fluent* (time-varying property of the world) holds at particular *timepoints* (real or integer numbers) if it was initiated by an *event* occurrence at some earlier timepoint and was not terminated by another event occurrence in the meantime. Similarly, a fluent does not hold at a particular timepoint if it was previously terminated and was not initiated in the meantime. Fluents, events and timepoints are sorts of first-order language used for EC formal representation.

Domain-dependent axiomatics is constructed in the following way. The fluent `access` is introduced for allowing or denying the network package transferring. The fluent parameters are protocol type, source and destination addresses and ports, allow/deny action, and ACL sequential number. The following expression defines that access fluent is not hold (until the corresponding packet reaches firewall):

```
initially_false(F) :- F = access(Protocol, s_ip(X1, X2, X3, X4), P1,
d_ip(Y1, Y2, Y3, Y4), P2, Action, Rule).
```

The access fluent is initiated by request_access() event, which has analogous parameters.

Let us consider an example (see Table 1) [2].

Table 1. Example of ACL with filtering anomalies

order	protocol	source IP	source port	destination IP	destination port	action
1	tcp	140.192.37.20	any	*.*.*.*	80	deny
2	tcp	140.192.37.*	any	*.*.*.*	80	accept
3	tcp	*.*.*.*	any	161.120.33.40	80	accept
4	tcp	140.192.37.*	any	161.120.33.40	80	deny

The first rule is translated to domain-dependent EC axiomatic as:

```
initiates(E,F,T) :- E = request_access(tcp, s_ip(140, 192, 37, 20),
P1, d_ip(Y1, Y2, Y3, Y4), 80, 1), F = access(tcp, s_ip(140, 192, 37,
20), P1, d_ip(Y1, Y2, Y3, Y4), 80, deny, 1).
```

The left rules of access control list are translated in the same way. After axiomatics generation, the verification procedure runs CIFF request:

```
?- run_ciff([policies], [holds_at(access(Protocol, s_ip(X1,Y1,Z1,W1),
Port1, d_ip(X2,Y2,Z2,W2), Port2, Action1, Rule1), T),
holds_at(access(Protocol, s_ip(X1,Y1,Z1,W1), Port1,
d_ip(X2,Y2,Z2,W2), Port2, Action2, Rule2), T), Rule1 #< Rule2], A),
```

where `policies` is a file with domain-dependent axiomatics, two `access` fluents express simultaneous satisfability of rules conditions, constraint `Rule1<Rule2` makes the same sequential order as in access control list, `A` is a CIFF answer containing abducible predicates and their relations.

CIFF output contains substitutions for all variables corresponding to the intersection of conditions of first and second rules from Table 1. There are the following substitutions:

```
Rule1=1, Rule2=2, Protocol=tcp, Port1=_77366, X1=140, Y1=192, Z1=37,
W1=20, Port2=80, X2=_77316, Y2=_77326, Z2=_77336, W2=_77346,
Action1=deny, Action2=permit, T=_68723,
```

and also the substitution for the variable `A` which represents CIFF answer. It consists of two parts. The first one is ground `happens` predicates:

```
happens(request_access(tcp,s_ip(140,192,37,20),_77366,d_ip(_77316,_77
326,_77336,_77346),80,0),_59492),
happens(request_access(tcp,s_ip(140,192,37,20),_77366,d_ip(_77316,_77
326,_77336,_77346),80,1),_68699),
```

and the second one is (in)equalities in prefix form:

```
<(_68699,_68723), <(_59492,_68723), =(_59492,0),=(_68699,0).
```

This CIFF answer can be reduced by removing one of happens predicates, since they have the same argument values and, as it can be seen from inequalities, happen at the same timepoint 0. Thus, a scenario which detects an anomaly consists in TCP packet receiving with source IP 140.192.37.20, any source port, any destination address and destination port 80.

The anomaly of rules with the same action is a redundancy anomaly. In other cases the type of anomaly is detected by predicate filter_type/4. This predicate detects anomalies of shadowing, generalization, correlation, and a particular case of shadowing and generalization where rules conditions are equal. First two parameters are lists with protocols, addresses and ports. The third argument contains type of anomaly, the fourth one detects an intersection between rules conditions.

For empty lists as 1, 2 and 4th arguments, all types of anomalies are deducible:

Here is an example for generalization:

```
filter_type([x|T1],[x|T2],generalization,[x|T]) :-
filter_type(T1,T2,generalization,T).
filter_type([H1|T1],[H1|T2],generalization,[H1|T]) :-
filter_type(T1,T2,generalization,T).
filter_type([H1|T1],[x|T2],generalization,[H1|T]) :-
filter_type(T1,T2,generalization,T).
filter_type([x|T1],[H2|T2],correlation,[H2|T]) :-
filter_type(T1,T2,generalization,T).
```

The constant x corresponds to * for address and “any” for port in table 1. The first three rules in above code deduce generalization for extended lists, when one of two conditions are met: (1) if generalization for shorter lists is already deduced and the same head elements are added, (2) if * is added to the first list.

If * is added to the second list, then correlation anomaly is deduced.

Here is an example for the anomaly between the first and the second lines of table 1:

```
?-filter_type([tcp,140,192,37,20,x,x,x,x,x,80],
     [tcp,140,192,37, x,x,x,x,x,x,80], A,L).
```

The anomaly type and intersection is detected as:

```
A=generalization, L=[tcp,140,192,37,20,x,x,x,x,x,80].
```

4 Software Prototype

Filtering policy verification is a part of functionality of SEcurity Checker software (SEC). Having taken system description in System Description Language (SDL) and policy definition in Security Policy Language (SPL), “Transforming and Interpreting” component translates them to domain-dependent EC axiomatics (Figure 1). Then anomalies are formulated as CIFF queries, which result in a list of detected anomalies, if any. After that, SEC defines appropriate resolution strategy and applies it.

EC based module can be loaded and used along with other verification modules (Figure 2). Each verification modules generate logs that contain information about verification process and results. For each anomaly, the verification result form depicts: verification module name, inconsistency type, conflicting rules, possibility of inconsistency resolution and resolution strategies, if any is available.

Figure 3 presents the form on which anomaly details can be viewed.

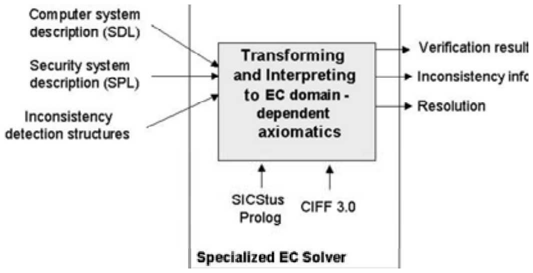
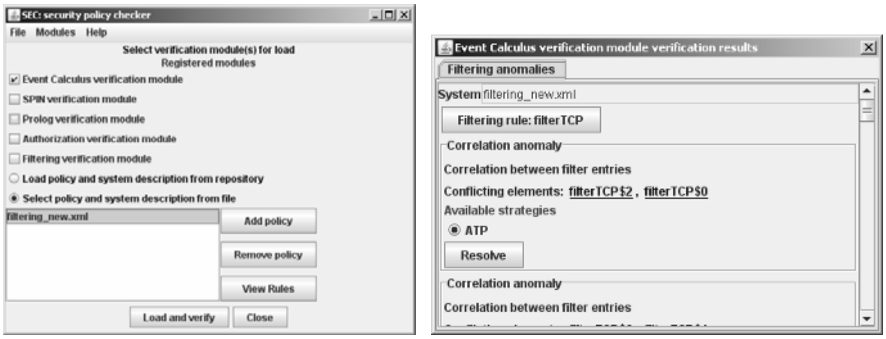

Computer system description (SDL)
Security system description (SPL)
Inconsistency detection structures
Transforming and Interpreting to EC domain - dependent axiomatics
Verification result
Inconsistency info
Resolution
SICStus Prolog
CIFF 3.0
Specialized EC Solver

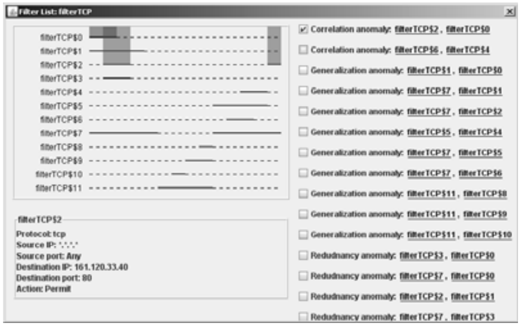
Fig. 1. Generalized representation of Event Calculus module



Select verification module(s) for load
Registered modules
[✓] Event Calculus verification module
[ ] SPIN verification module
[ ] Prolog verification module
[ ] Authorization verification module
[ ] Filtering verification module
○ Load policy and system description from repository
● Select policy and system description from file
filtering_new.xml

System filtering_new.xml
Filtering rule: filterTCP
Correlation anomaly
Correlation between filter entries
Conflicting elements: filterTCP$2, filterTCP$0
Available strategies
● ATP
Correlation anomaly
Correlation between filter entries

Fig. 2. Main form and verification result form



filterTCP$2
Protocol: tcp
Source IP: '*.*.*.*'
Source port: Any
Destination IP: 161.120.33.40
Destination port: 80
Action: Permit

[✓] Correlation anomaly: filterTCP$2, filterTCP$0
[ ] Correlation anomaly: filterTCP$6, filterTCP$4
[ ] Generalization anomaly: filterTCP$1, filterTCP$0
[ ] Generalization anomaly: filterTCP$7, filterTCP$1
[ ] Generalization anomaly: filterTCP$7, filterTCP$2
[ ] Generalization anomaly: filterTCP$5, filterTCP$4
[ ] Generalization anomaly: filterTCP$7, filterTCP$5
[ ] Generalization anomaly: filterTCP$7, filterTCP$6
[ ] Generalization anomaly: filterTCP$11, filterTCP$8
[ ] Generalization anomaly: filterTCP$11, filterTCP$9
[ ] Generalization anomaly: filterTCP$11, filterTCP$10
[ ] Redudnancy anomaly: filterTCP$3, filterTCP$0
[ ] Redudnancy anomaly: filterTCP$7, filterTCP$0
[ ] Redudnancy anomaly: filterTCP$2, filterTCP$1
[ ] Redudnancy anomaly: filterTCP$7, filterTCP$3

Fig. 3. Filtering anomaly details

One can select any anomaly and see the intersection of involved rules conditions. Allowing rules are marked with green color while denying rules are marked with red one.

Several experiments were conducted to analyze efficiency of EC and abductive search implementation of filtering policy verification. We compared such verification with exhaustive search for every pair of rules using pure Prolog.

5 Conclusions

In the paper we have described Event Calculus axiomatization and abductive search for inter-firewall anomalies. The main idea of development is to build the adequate network model including the network and policy description, and the network behavior model. Then policy anomalies definitions are added and the model is checked against possible anomalies. The verification module described was implemented in Java and uses the software library CIFF working on the base of SICStus Prolog. Experiments fulfilled have showed that our approach is gainful for real ACL examples, where the most of allowing rules are disjoint and considered as exceptions from general denial rules. Future work is to build uniform framework for verification of other policy categories, such as access control, authentication, confidentiality, etc., and to consider inter-categories policy conflicts.

Acknowledgments. This research is being supported by grant of Russian Foundation of Basic Research (Project No. 07-01-00547), program of fundamental research of the Department for Informational Technologies and Computation Systems of the Russian Academy of Sciences (contract No 3.2/03), Russian Science Support Foundation and partly funded by the EU as part of the POSITIF project (contract No. IST-2002-002314) and RE-TRUST (contract No. 021186-2).

References

1. Al-Shaer, E., Hamed, H.: Firewall Policy Advisor for Anomaly Discovery and Rule Editing. Integrated Network Management (2003)
2. Al-Shaer, E., Hamed, H., Boutaba, R., Hasan, M.: Conflict classification and analysis of distributed firewall policies. IEEE Journal on Selected Areas in Communications 23(10) (2005)
3. Bandara, A.K., Kakas, A.S., Lupu, E.C., Russo, A.: Using Argumentation Logic for Firewall Policy Specification and Analysis. In: State, R., van der Meer, S., O'Sullivan, D., Pfeifer, T. (eds.) DSOM 2006. LNCS, vol. 4269, Springer, Heidelberg (2006)
4. Bandara, A.K., Lupu, E.C., Russo, A.: Using Event Calculus to Formalise Policy Specification and Analysis. In: IEEE Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society Press, Los Alamitos (2003)
5. Endriss, U., Mancarella, P., Sadri, F., Terreni, G., Toni, F.: The CIFF Proof Procedure: Definition and Soundness Results. Technical Report 2004/2, Department of Computing, Imperial College London (2004)
6. GORGIAS. Argumentation and Abduction, http://www2.cs.ucy.ac.cy/~nkd/gorgias/
7. Kowalski, R.A., Sergot, M.J.: A Logic-Based Calculus of Events. New Generation Computing 4 (1986)

A New Approach to Security Evaluation of Operating Systems

Peter D. Zegzhda, Dmitry P. Zegzhda, and Maxim O. Kalinin

Information Security Centre of Saint-Petersburg Polytechnical University,
P.O. Box 290, K-273, Saint-Petersburg, 195273, Russia
{zeg,dmitry,max}@ssl.stu.neva.ru

Abstract. This paper addresses to the technique of security evaluation based on security attributes analysis in discretionary access control. A multi-level framework is built to calculate a set of effective user's permissions automatically. Information about the effective access rights is necessary during security verification procedure. In this paper we also propose a schema of Security Evaluation System.

Keywords: access control, effective access permissions, evaluation, multi-level framework of security attribute, security.

1 Introduction

Information security involves protecting systems against unauthorized loss of confidentiality of data and network communications; integrity of data, systems, and networks; and availability of services and data. Securing operating systems and networks is not easy. Evaluating trade-offs between functionality and security is a difficult task, but data security is too important to neglect. To get assurance in an adequate security, users can rely on the word of the manufacturer; test the system themselves; or rely on an impartial assessment by an independent body.

Inadequate default security settings provided by the manufacturer cause many security weaknesses. One of the examples is incorrect default security configuration of MS Internet Explorer described in [1]. To test the system themselves and thus raise its security level, administrator ought to look for the *Security Guides* (e.g. [2]), that provide a detailed guidance on enhancing security configuration to address threats identified in user's environment. To maintain the system and permanently monitor its security settings, they have to be the security experts.

Security evaluation is thus the only alternative to taking a security system on trust. At last decades a number of individual countries developed the security evaluation standards (e.g. [3]). To get a higher assurance (e.g., levels over EAL5 in CC), developers should specify a security model and verify its safety property using *formal* approach. For high assurance systems, the difficulties of using formal methods add further complexity to both development and evaluation procedures. In this paper we discuss our approach that represents a suitable compromise for evaluation. Our technique is based on calculus of a *set of effective access permissions* (*SEAP*). The

SEAP represents the user's real access capabilities in the operating system. Knowledge of available rights is the most valuable information for evaluation, because it makes us ensured that the given security configuration does not break data protection. The *SEAP* calculation is based on the characteristics of modern security subsystems where access control is built on the security attributes of subjects (e.g., users and groups) and objects (e.g., files and folders). We have structured the security attributes in the form of a *multi-level framework of security attributes*. The rest paper is organized as follows. In *Section 2* we observe the related works. In *Section 3* the security mechanisms used in the contemporary operating systems are analyzed. In *Section 4* we introduce a multi-level framework of security attributes that allows us to calculate the set of effective permissions. Finally, *section 5* concludes the paper.

2 The Related Works

A great many of works on computer systems security are related to security evaluation. Firstly, there are the security standards. They declare the target and purposes of evaluation, but unfortunately they do not regulate any approaches, techniques, or means of the evaluation.

Secondly, there are the theoretical methods based on theoretical security modeling, specification and resolving. For example, [4] is based on the formal logic for security model specification and conflicts searching in RBAC. The graphical approach for evaluation is proposed in [5], where security policy consists of domain (the system abstraction) and the requirements (authorization rules). A formal authorization policy model is proposed in [6], where a model establishes a connection of applying authorization policies on an administration domain with dissecting the domain into the authorized, denied, and undefined divisions. All mentioned approaches are suggested to support security engineering to evaluate the authorization rules and remove conflicts in the policies before they are integrated with other system functionalities. The user-oriented evaluation is targeted at the already assembled system, and it must guarantee the customer that she obtains a secure configuration.

The third group of works is designed for security evaluation of concrete systems: e.g., a MulVAL tool for net security analysis [7], a Prolog-based approach for Windows XP bugs scanning [8]. These techniques could reason about '*privileges escalation*' and '*misconfigurations*', but they scan the given set of the system parameters and they can not discover real access permissions available for the user. We need a smart approach that will take into consideration a mutual influence of security parameters. There are also the software vulnerabilities detectors (e.g., Enterprise Security Manager of Symantec; NetIQ Security Analyzer; Microsoft Baseline Security Analyzer). These scanners analyze a reduced set of the system user-level objects and they can not disclose the factor that leads to the access granted. Therefore, to our knowledge, the general problem of developing a practically useful approach for security evaluation in common operating systems has never been addressed.

3 A Security Fundament in Modern Operating Systems

Today, every type of the operating system supports a huge number of system resources that accessible through drivers, services, applications, and network communications. For software security has emerged as a foremost concern for modern information enterprise, each named object of any type is protected with access control mechanism. A traditional schema of access control is based on security attributes, the set of unique characteristics of every system entity. The security attributes identify the protected entity for the secure system. This approach as named as the attribute-based access control. For example, in MS Windows (as well as in other types of the systems), users and groups form the subjects. The objects (e.g. files, services, regkeys) represent data resources. Authorization decision is made based on subjects and objects attributes. The security attributes are access control lists (ACLs), owners, groups, inheritance flags, security options of application.

A particular mechanism for enforcing security in the computer system is called as *security model*. DAC security model [9] has a drawback that it does not provide a real assurance on the flow of information in a system. It is easy to bypass the access restrictions stated through the authorizations. For example, a user who is able to read data can pass it to other users not authorized to read it without the cognizance of usage of information by a user once the user got it. MAC model [10] governs access on the basis of classification. Verification of the safety of any kind of MAC policy is obvious as mathematical proof, but it becomes necessary in particular cases of real-world realizations. With RBAC [11] access decisions are based on the roles that individual users have as part of an organization. The process of defining roles should be based on a thorough analysis of how an organization operates and should include input from a wide spectrum of users in an organization.

MAC and RBAC policies are not particularly well suited to the leading world-known solutions. A great majority of the operating systems uses DAC-based security models as the basis of an access control (see Tab 1.).

Table 1. Security Models Types in the Modern Operating Systems

	UNIX-like Systems	*Windows 2000/XP/2003*	*Windows Vista*
DAC	Built-in (access bits)	Built-in (access control lists)	Built-in (access control lists)
MAC	Add-in (core patches)	No	Built-in (integrity control)
RBAC	Add-in (core patches)	No	No

As the operating systems grow from the attributeв security, the security attributes participate in the DAC-based security implemented in the system reference monitor. For instance, Microsoft Windows, Linux and other protected systems use DAC realizations in form of ACLs, access bits, messages control, etc. Configuring the security attributes one must be ensured in real security. Because of amount of security attributes involved in to access control, evaluator would never say whether access is granted or not. Such degree of system complexity forces us to design a new approach to reduce a set of processed attributes to the set of *effective (real) access permissions*. Determining the set of effective access permissions and finding the mismatches between expected and given access rights can be calculated and give a verdict of the system security *for the definite system and for its definite state*. In this paper, we use this statement to discover incorrectly granted or missed rights and, in such way, to evaluate the system security of the DAC-based systems.

4 Calculation of the Effective Permissions

To solve the problem of security evaluation in case of attribute-based access control in MS Windows which realizes the DAC security model, we have built a *multi-level framework of security attributes*. Fig. 1 demonstrates a part of our framework (levels dedicated to *Active Directory* are canceled). On this schema there are 12 levels (each level is labeled by factor that influences on the access control work):

- *lower levels* (1…3): *analysis levels* — we divide the security attribute *'ACL'* to the set of '*access control entries*' (ACE) attributes, and then *'ACEs'* to access *rights*.
- *upper levels* (4…12): *synthesis levels* — we calculate a *set of effective access permissions* (*SEAP*), i.e. the access permissions really available for subjects at the given level of the framework. Each upper level is based on the lower one.

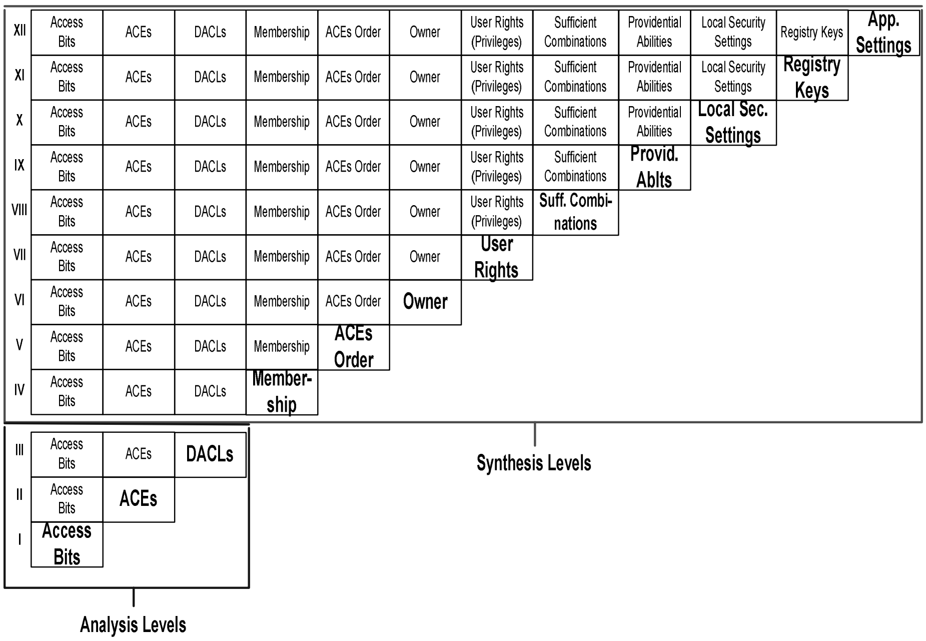


Fig. 1. The Multi-Level Framework of Security Attributes (for MS Windows)

The rights of the level are calculated as projection of security attribute (we depict this attribute the *security factor*) value to subject's access bits. Each attribute can be reduced to the subject's permissions or prohibitions. For example, the SEAP on the ownership level equals to *Full Control* available for the object. The SEAP on the membership level equals to the user's rights *plus* rights, provided by the user's group rights. In the same matter, we can bring any security factor from any level to the access permissions. In other words, we calculate an *effective* set of rights by taking into account the security factor of the corresponding level.

Therefore, in common case, $SEAP_i = f(SEAP_{i-1}, U_i)$, where $2 \le i \le N$, N is the number of levels in the framework, U_i denotes the unique security attribute on the given level i of the framework. Function f is the attribute projection function. It calculates the effective permissions with taking into account the level attribute. f is thus defined for every level separately. In the most common case, this is a logical *AND* or logical *XOR*. For example, for level 5 of the framework ACEs order is the security factor, and the $SEAP_5$ is calculated in the following way (*subject* is a user or a group):

$$SEAP_5(subject) = SEAP_4^{+}(subject)\ \textbf{XOR}\ SEAP_4^{-}(subject),$$

where $SEAP_4^{+}$ is the $SEAP_4$ calculated for the subject's permissions, $SEAP_4^{-}$— for its prohibitions;

$$SEAP_4(subject) = SEAP_3(subject)\ \ \textbf{AND}$$
$$SEAP_3(\{groups_j\} \mid \forall i\ groups_j \ni subject);\ \ldots\ldots\ldots\ldots\ldots;\ SEAP_1(subject) = \{bits_k\},$$

where $\{bits_k\}$ is the set of access bits which set in the ACE_k existing in the system for *subject* in the object's ACL. This procedure can calculate a set of the SEAPs for each subject, object on each level of the framework.

To evaluate the system security, we need to specify security criteria that would delimit secure and insecure configurations. According to tradition, we look at the security of the operating system configuration through the *security of the states* [9], [10]. We consider the *security configuration* of the state as a set of subjects, objects, and their security attributes. Since, as mentioned above, every security attribute can be reduced to SEAP depending on the framework level, the state can contain extra or missed access permissions hidden in the subjects and objects existence, or in the sets of the attributes. If we add a term of constraint, the set of access restrictions given for the '*subject-object-SEAP*' triple, to this schema, then we will detect the access permissions or prohibitions available or not available to the subject at the given level of framework in the given system. And the system *in the given state* will be *secure* if nothing breaks the security conditions settled by the constraint. If the system is *insecure* in correspondence to the given constraint, we can calculate the level of the multi-level framework and extract from it the factor that brings the permissions not desired in that configuration. It is possible, because one level of the framework projects corresponds to a single attribute.

In theory of security (e.g. in specification of MAC, DAC, and RBAC) security proof is based on a predicate that checks whether the access right available for the given subject. In real-world systems security verification has to be done on every level of the attributes hierarchy. Besides this, security must be proved for the set of the sets of the attributes (the power of security attributes). Our approach puts the structure on the set of attributes and inducts the security calculus from multiple attributes to effective rights.

5 Conclusion

In this paper, we addressed to security evaluation problem that is very actual today. We have reviewed the attributed nature of traditional access control released in Windows. These allowed us to discuss about the multi-level framework of security attributes. The variety of attributes can be structured in the form of pyramid, because one factor of influence is added on the newer security level. Each attribute brings additive permissions to the user. Projection of the security attribute to the access rights can be calculated programmatically. It allows us to build an automated security evaluation software that uses the multi-level framework (Fig. 2). Such evaluation system is the target of our future works.

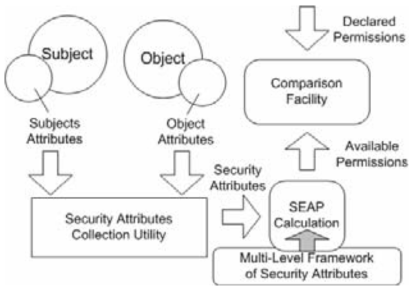


Fig. 2. The Schema of Security Evaluation System

References

1. http://www.kb.cert.org/vuls/id/181038, http://secunia.com/advisories/18255
2. Dillard, K., Maldonado, J., Warrender, B.: Microsoft Solutions for Security. Windows Server 2003 Security Guide. Microsoft (2003)
3. Common Criteria. ISO/IEC 15408. Information technology. Security techniques. Evaluation criteria for IT security (2005)
4. Jajodia, S., Samarati, P., Subrahmanian, V.S.: A Logical Language for Expressing Authorizations. In: Proc. of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (1997)
5. Hoagland, J.A., Panday, R., Levitt, K.N.: Security Policy Specification Using a Graphical Approach. Tech. report CSE-98-3 (1998)
6. Dai, J., Alves-Foss, J.: A Formal Authorization Policy Model. Proc. Software Engineering Research & Applications (2003)
7. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: A logic-based network security analyzer. In: 14th USENIX Security Symposium (2005)
8. Windows Access Control Demystified. Technical Report TR-744-06
9. Harrison, M.H., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Communications of the ACM 19(8) (1976)
10. LaPadula, L.J., Bell, D.E.: Secure computer systems: A mathematical model, ESD-TR-278, V. 2, The Mitre Corp. (1973)
11. Ferraiolo, D., Kuhn, R.: Role-based access controls. In: Proc. of the 15th NIST-NCSC National Computer Security Conference (1992)

Multi-agent Peer-to-Peer Intrusion Detection

Vladimir Gorodetsky, Oleg Karsaev, Vladimir Samoylov, and Sergey Serebryakov

St. Petersburg Institute for Informatics and Automation
39, 14 Liniya, St. Petersburg, 199178, Russia
{gor,ok,samovl,sergey_s}@iias.spb.su

Abstract. Ever increasing use of heterogeneous networks including mobile devices and ad-hoc sensor networks signifies the role of such information systems' properties as openness, autonomy, cooperation, coordination, etc. Agent-based service-oriented Peer-to-Peer (P2P) architecture provides attractive (if not unique) design and implementation paradigm for such systems. This trend implies coherent evolution of security systems, that put in use the notions of distributed security policy, distributed intrusion detection systems, etc.[1], requiring novel ideas. The paper proposes new architecture for such security systems. This architecture provides cooperative performance of distributed security means (agents) supported by distributed meta-knowledge base implemented as an overlay network of instances of P2P agent platform set up on top of P2P networking provider. The paper also analyzes new issues of P2P security systems with the main emphasis on P2P training of security agents to correlation of alerts produced by other relevant agents. An artificially built case study is used to highlight the essence of P2P security agent training to P2P decision combining and to exhibit new problems.

1 Introduction: Modern Information Technology Trends

Computer network security is very dynamic area that has to permanently adapt to new trends in modern information technology, hardware and software changing, communication environment changing. The latter permanently enlarge diversity and smartness of threats as well as toughen the requirements to computer and information systems security. Let us at first analyze the aforementioned trends and existing or potential solutions.

Modern computer and information systems tend to be composed *of many heterogeneous devices*: mobile devices (smart phones, PDAs, etc.), laptops, home appliances, desktop computers, etc. The above enlarges diversity of protocols to be used, and, as a consequence, enlarges the threats diversity. In such networks, communication channels and, therefore, devices' interfaces are becoming excessively manifold. E.g., a computer node may need to support TCP\IP and Bluetooth; Pocket PC node may need to support WiFi; Mobile phone node may need to support Bluetooth, etc. Ever increasing use of wireless networking is also a trend. Therefore,

[1] For open agent-based system, the notion of distributed trust should also be used: it provides a way to find a tradeoff between security and openness.

due to limited communication distance and devices' mobility, computer network becomes populated with dynamic nodes' set that necessitates wide use of *Peer-to-Peer* (*P2P*) computing. As a result, information technologies for ad-hoc and mobile networking systems rapidly evolve proposing new architectures of and design approaches to the network-based information systems. Among them, *agent-based notion of Web services* accentuating semantic search and dynamic service composition and corresponding agent-based technology looks very perspective and well suited to applications dealing with highly distributed entities.

An important trend is the necessity to provide modern systems with more *openness* and *autonomy* [11]. Indeed, nodes of mobile network may join and leave the system at any time thus either enriching the set of available services or cutting it down. Therefore, each node is autonomous and the network as a whole is an open system. Within agent technology, coherent operation (cooperation, coordination, competition, etc.) of autonomous entities (agents) within open P2P system can be provided with shared ontology and meta–knowledge specifying agent capabilities (e.g., services they can provide) and methods of access to these capabilities. These components can support *transparency of agents' interaction* that assumes that application agents may know little or even nothing about existence in the network of particular services and agents capable to provide with them. On the other hand, the agents may know nothing about particular implementation of P2P networking and have an impression of direct interaction with the network agents while requesting from or providing for other network agents the needed services.

Such agent-to-agent (P2P) interaction can be achieved by using adequate FIPA Message Transport Service (MTS) [1]. For P2P open networks of agents, FIPA MTS may be realized as an overlay network set up on top of a P2P transport protocols, for instance, a JXTA transport, a Bluetooth OBEX or over some other *P2P transport providers*. For a centralized case, FIPA proposed a solution for Agent and Service discovery component called Agent platform [2], composed of Agent Management System (AMS) and Directory Facilitator (DF). AMS is in charge of *White Pages* (WP) service (it specifies the list of agents coupled to the platform) and the agent life cycle maintenance. DF provides *Yellow Pages* (YP). But these solutions cannot support P2P interaction of agents in serverless environment. One of satisfactory solutions was proposed in [7], and its reusable implementation was described in [8].

The paper objective is to analyze influence of the current trends in modern computer information technologies on computer and information security and to propose new security system architecture intended to defend open agent-based service-oriented systems whose components semantically interact on P2P basis. This architecture should provide for cooperative operation of distributed security means through their P2P interaction in the alert correlation procedure, when an agent is capable to use local data and own knowledge as well as intelligence and decisions produced by other agents. The second objective is set to analyze some associated problems concerning design of P2P security system of the proposed architecture. In this analysis the main attention will be paid to P2P training of security agents for alert correlation.

In the rest of the paper section 2 analyzes impact of trends and evolution of modern information technologies on the security associated problems and potential solutions. Section 3 actually states the lack of the security-oriented research relevant to information systems composed of highly dynamic population of autonomous entities. Section 4 proposes new architecture of the security systems in question and middleware that is an overlay network of instances of P2P Agent platform providing for transparent P2P interaction of distributed security means implemented as agents. Section 5 outlines an example of multi-agent service oriented intrusion detection system (IDS). Section 6 discusses the P2P meta-learning of alert correlation problem. Conclusion summarizes the paper contribution and discusses future research.

2 Security of P2P Agent-Based Service-Oriented Systems

New trends and current evolution of information technologies implies coherent evolution of security systems, in particular, ones intended to defend P2P agent-based service–oriented systems. The latter introduces the notions of distributed security policy and distributed IDS leading to novel security system architectures. Let us analyze this issue.

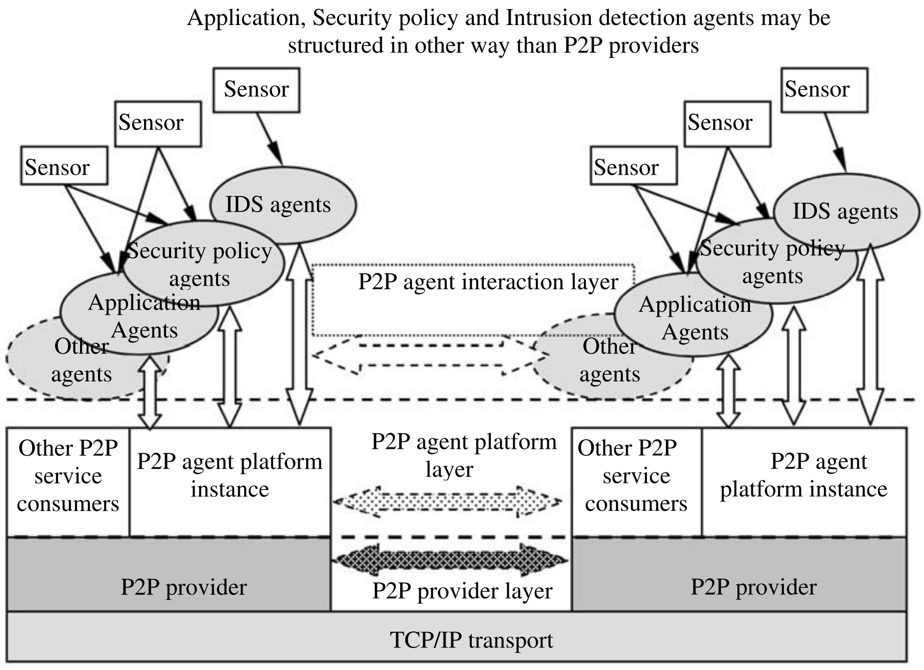

Application, Security policy and Intrusion detection agents may be structured in other way than P2P providers
Sensor
Sensor
Sensor
IDS agents
Security policy agents
Application Agents
Other agents
Sensor
Sensor
Sensor
IDS agents
Security policy agents
Application Agents
Other agents
P2P agent interaction layer
Other P2P service consumers
P2P agent platform instance
P2P agent platform layer
Other P2P service consumers
P2P agent platform instance
P2P provider
P2P provider layer
P2P provider
TCP/IP transport

Fig. 1. P2P provider, P2P Agent platform, and P2P application agent overlay networks

Indeed, considered agent systems are intended for cooperative problem solving. Within P2P agent-based service–oriented applications the cooperation concerns distributed execution of queries often assuming service composition. Thus, each process of (distributed) service execution may involve several nodes and that is why the defended object, the process of service execution, is distributed in this case. Therefore traces of the same attacks, e.g., attacks using self-replicating virus as well as other classes of attacks, may be detected by several agents set up on the same or different nodes of the network. On the other hand, the traces of the same attack may exhibit at different levels of query requesting and executing, e.g., at network level (in network traffic), in operating system and other system program audit trails (host-based level), and in various host–based application (data bases, ftp, http, etc.). It is assumed that aforementioned processes are monitored by particular agents (security policy model checking agents, intrusion detection agents, etc.). Each of them is tuned (trained), say, either to a node-based–model checking, or to detection of particular class of attacks, viruses or anomalies. Therefore it is reasonable to organize, in various situations, interactions of subsets of agents if one or several agents produce alert(s). Of course, an important challenging question is how to determine such subsets[3].

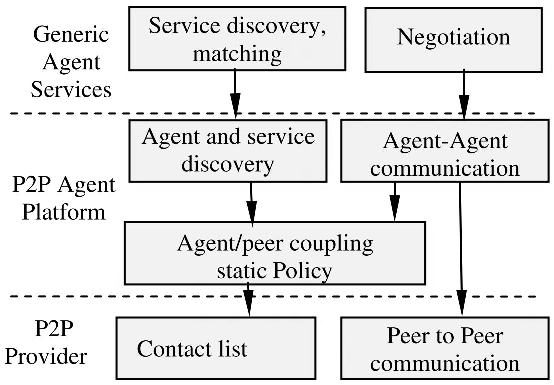

Generic Agent Services
Service discovery, matching
Negotiation
P2P Agent Platform
Agent and service discovery
Agent-Agent communication
Agent/peer coupling static Policy
P2P Provider
Contact list
Peer to Peer communication

Fig. 2. Functional architecture of the developed P2P agent platform and P2P provider

According to the modern view [11] aforementioned agents can be thought of as agents providing services on P2P basic. That is why all these agents can also be set up on top of P2P agent platform as regular service providers accessible on P2P basis using distributed White and Yellow page services of P2P Agent platform.

Corresponding architecture including P2P application agents and security components as well as their interaction is shown in Fig. 1. It consists of three overlay networks where P2P overlay network is set up on top of TCP/IP transport provider, P2P Agent platform instances overlay network is set up on top of P2P providers' network and, finally, application agents overlay network that also includes security related agents is set up on top of P2P Agent platform.

Such architecture of security system possesses many advantages, and the most important ones are scalability, autonomy of distributed security system components, capability to monitor heterogeneous distributed processes (data sources) within P2P work containing heterogeneous devices, expandability, fault tolerance and some other. Below a case study of security system operating in accordance with the proposed architecture and distributed multiple alert correlations is outlined.

[3] This aspect is not so far considered here although general ideas of how it can be done based on P2P data mining are mentioned in the end of the paper.

3 Related Works

Computer network security problem for multi-agent service–oriented P2P system is weakly highlighted in the literature. Existing papers mostly concern multi-agent architecture. AAFID [1], AHA! IDS [14], IDA [13], MAIDS [10] are well known examples of multi-agent IDS. There also exist a lot of papers presenting high level description of multi-agent IDS architecture and their prototype-level implementations. According to the authors' knowledge only [2] describes P2P agent-based approach and its implementation. Corresponding IDS is called MAPIDS (Mobile Agent based Peer-to-peer Intrusion Detection System). However, this system does not support service–oriented architecture and in many respects considers the problem in question in too simplified way.

4 P2P Agent Platform and P2P Provider

This section briefly presents functional architecture of the middleware component supporting transparent interaction of application agents as well as briefly outlines its implementation details. Its more detailed description can be found in [8]. Let us remind that security policy agents as well as IDS agents belong to the set of application agents (Fig. 1), i.e. this middleware is also used by security components of the defended system.

Three-layer functional architecture of this middleware preserving basic ideas of NA WG6 functional architecture [6] is shown in Fig. 2. *The bottom layer* of the architecture in question corresponds to *P2P service* provider. This service may be provided not only on request of P2P agent platform but also on request of any other consumer needed P2P communication. In general, standard software like JXTA, WiFi OBEX or other P2P provider can be used to support P2P communication. In the implemented case, ad-hoc P2P provider developed by the authors is used. *Contact list* of P2P provider contains the list of neighbor nodes (peers) of P2P provider. This list can be managed through a mechanism, a functionality of the latter. Its P2P communication component provides connection with neighbor nodes and P2P routing according to a protocol. In the implemented case, two typical protocols are available, flooding and gossip [13]. Detailed description of P2P provider is given in [8].

Intermediate layer corresponds to P2P Agent platform itself. It provides for a full decoupling of application agents layer and P2P provider. From P2P provider viewpoint, P2P agent platform can be thought of as a client, or consumer of its services. From application agents' viewpoint, this platform is considered as specific services provider. In particular, it provides the former for such specific services as agent and service discovery, agent–agent communication and agent–peer coupling.

The above services are provided through distributed P2P search according to the gossip protocol. The component "Agent and service discovery" contains also White and Yellow pages specifying services of agents registered on this particular instance of agent platform and list of agents associated with particular services. White and Yellow pages are implemented according to FIPA specification [5] as agents providing for corresponding services. The specification format used for these services can be found in [8, 9]. Jointly White and Yellow pages play the role of an instance of distributed meta–knowledge base available to other agents of the network. Exactly White and Yellow pages allow the semantic service search and dynamic service composition[4] that is the main difference between SOA and its agent-based variant. The P2P agent platform provides also agent–agent communication service supporting the capability of message passing. Agent–peer coupling functionality determines management policies between peer and agent platform. For example, in the implemented version, P2P agent platform manages P2P provider's contact list through the special mechanism of the bottom layer mentioned above.

At the top layer, generic agent services provided with basic capabilities of service discovery (matching) and negotiation are situated.

Interaction between P2P provider and a consumer comes to interaction through standard interfaces supporting access of P2P provider to consumer and vise versa. In order to use P2P service, consumer has to register at the local peer that is a node of P2P network. To register consumer has to specify its own type and identifier, which must be unique amongst all P2P network consumers. If particular application needs no longer the P2P provider service it has to deregister. Also peer provides with some functionality to manage its own contact list via adding and deleting records, etc. Agent platform can suspend and resume its own presence at a peer, for instance, during temporal unavailability. If some applications set on concrete device need to use P2P transport (e.g., to send message to a P2P AP) P2P provider has to be installed on that device.

Any application needed to be connected to the P2P network as peer client must identify peer running on the node (device) and register. Let us note that peer and P2P consumers are "weakly coupled" because the latter, in some scenarios, may work without P2P services. On the contrary, agents and P2P Agent platform are "tightly coupled", i.e. P2P agents cannot operate beyond the agent platform: the latter fully manages the agent life cycle, i.e. loads, creates, suspends, resumes and destroys it.

White and Yellow page agents can create associations of agents set up on different peers which may collaborate as "neighbors" at application agent layer (*within P2P agent network*) forming other structure than peers' one. In general case this structuring may be dynamic and results from learning algorithms intended to establish a better or optimal configuration of the *P2P agent network* in order to improve performance of distributed service search, e.g., via decreasing communication overhead. So far this functionality is not realized in the developed P2P agent platform.

Description of the operation scenarios of the P2P agent network supported by the developed middleware can be found in [8].

5 Multi-agent P2P Intrusion Detection

Fig. 3 presents an example of P2P Intrusion Detection system (P2P IDS) whose architecture corresponds to the proposed in section 3. It sets up on top of instances of P2P agent platform that are set up on of P2P network. P2P IDS is a particular case of application and that is why everything described in section 1, 2 and 4 is also true for it. In this figure, instances of P2P Agent platform are represented by rectangles while the agents are represented as circles.

[4] In the existing P2P agent platform, this functionality is not implemented yet.

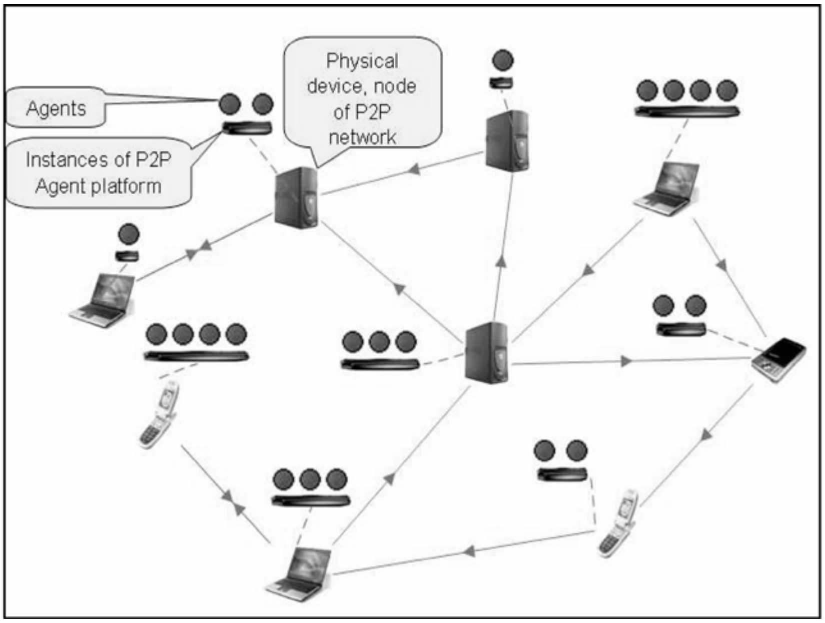
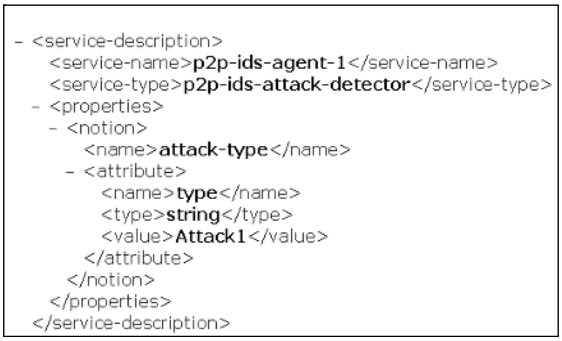

Agents
Instances of P2P Agent platform
Physical device, node of P2P network

Fig. 3. Example of a network topology and set up P2P IDS

This P2P IDS is composed of the agents of two classes, *agents-detectors* and *agent monitoring the system*, displaying progress of the P2P IDS performance. Every agent–detector is trained to detect suspicious activity (virus, attack, etc.) of a particular type. Every agent–detector is interpreted as provider of a particular service. It can produce corresponding alert when detects an attack of the "own" type and provide other agents with corresponding information upon their requests. The agents of the P2P IDS should have the capability to find the information of such kind when they need it. It is done through service search mechanism. Each IDS agent, during installation on the corresponding instance of the P2P agent platform, registers itself at White pages and registers its service specifications at Yellow pages.



```
- <service-description>
    <service-name>p2p-ids-agent-1</service-name>
    <service-type>p2p-ids-attack-detector</service-type>
  - <properties>
    - <notion>
        <name>attack-type</name>
      - <attribute>
          <name>type</name>
          <type>string</type>
          <value>Attack1</value>
        </attribute>
      </notion>
    </properties>
  </service-description>
```

Fig. 4. Service description example

An example of the service description is given in Fig. 4. The service description consists of service name, type and properties, i.e. the set of notions, describing the service at more detailed level. After the service is registered at Yellow pages, other agents can find it using special search mechanism provided by P2P Agent Platform. For instance, to find all agents providing for service

"*Detection_of_Attack_1*" (same as "Detection of attacks of class 1") the following search query is used:

```
service-type=='p2p-ids-attack-detector' AND attack-
                    type.type=='Attack1'.
```

The life cycle of P2P IDS agent and scenario of its operation are as follows:

1. Agent is installed on the P2P Agent platform. This can be done at any time independently of whether other agents are installed or not, whether they are operating or not. As it was mentioned above, during installation, the agent registers itself in White pages of P2P agent platform and registers its services in Yellow pages. No special advertisement of its service is needed. It is assumed that, at this time, agent is trained to detect particular kind of suspicious activity and is ready immediately to operate.
2. Using search capabilities of P2P Agent platform instance on which it sets up, the installed P2P IDS agent searches for agents trained to detect the same class of attack. The latter agents may set up on the same or other instances of P2P agent platform and operate with the same of other data sources. In general case, in its scenario of coalition formation, the installed P2P IDS agent can form such coalition dynamically via search for potential "neighbors" when detects a suspicious activity of its "own class" and forming the best coalition on-line or off-line through learning based on accumulated experience.
3. When P2P IDS agent is installed and appropriate coalition is formed it is ready to make decision using the results of processing of "its own data source" and also decisions of other agents of the coalition. For this purpose, any P2P IDS agent has to be trained to combine decisions produced by other agents of the coalition.

6 P2P IDS Agent Learning of Decision Combining

An important issue of P2P IDS system, as well as any other P2P decision making systems, is training of agents to combine decisions produced by other decision making entities on serverless basis. This problem is rather specific, multi-aspects and most of these aspects are challenging. A thorough analysis of this problem is the subject of P2P Data mining [4], and it is beyond the current paper scope. In this section, a particular aspect of this problem is highlighted, i.e. what P2P training and testing tool can look like and some previous results of experiments concerning serverless (without any centralized processing or using a super peer) combining of decision produced by P2P IDS agents. An objective of this consideration is to draw attention of researchers to the problem of P2P data mining within such important class of applications as P2P IDS.

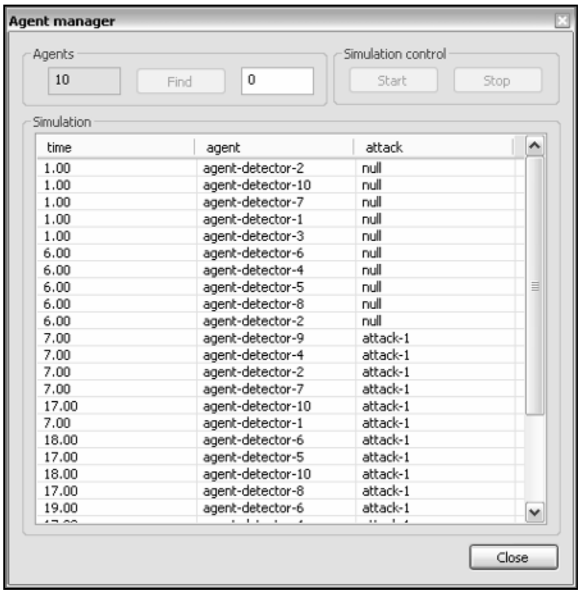
Each P2P IDS agent at any time either is "keeping silent" if it detects no suspicious activity of the "own class" or produces alert. The "colors" of alerts produced by different agents will be the same if they trained to detect the same class of suspicious activity. Therefore, output of a P2P IDS agent may be labeled as "*0*" in case of no alert or "*1*" otherwise.

Let us assume that a P2P IDS agent detects alert and in order to make decision asks another agents of the same "colors" about their decisions within a time interval close to the current time instant, and, if possible, concerning the same connection. As a result, it can produce an ordered binary sequence. This sequence is usually (in distributed data mining scope) called "meta-data". The agent can accumulate a sample of meta-data and, when these data are interpreted, use it for training and testing in off-line mode. Such training and testing samples can be collected by each P2P IDS agent and the latter can use them to learn decision combining.

Somewhat simplified version of a software tool intended for implementation of the aforementioned scenario of P2P training of P2P IDS agents was developed. The simplification is that each P2P IDS agent if it produces alert collects and combines the decisions only produced by the agents of the same "color". This software tool was used for investigating the dependence of the decision combing procedure quality on the number of P2P IDS agents of the same "color" set up in P2P IDS (Fig. 5).

In the experiments every P2P IDS agent preliminary trained for "local" decision making, computes contingency matrices corresponding to all agents of the same "color" based on the accumulated training meta–data samples. The set of these matrices is further used by IDS agents to evaluate competences of other agents whose decisions the former will combine.

In the simulation–based experiments (Fig. 5), 10 P2P IDS agents detecting attack of the same type (having the same "color") were involved in P2P intrusion detection. Training and testing samples were generated artificially (both containing 2000 records) using some probabilistic models for each of ten intrusion detection agents. The simulation objective was to investigate dependency of the averaged combined classification quality on the number of neighbors each agent communicates with


Agent manager

Agents: 10 | Find | 0

Simulation control: Start | Stop

Simulation

time	agent	attack
1.00	agent-detector-2	null
1.00	agent-detector-10	null
1.00	agent-detector-7	null
1.00	agent-detector-1	null
1.00	agent-detector-3	null
6.00	agent-detector-6	null
6.00	agent-detector-4	null
6.00	agent-detector-5	null
6.00	agent-detector-8	null
6.00	agent-detector-2	null
7.00	agent-detector-9	attack-1
7.00	agent-detector-4	attack-1
7.00	agent-detector-2	attack-1
7.00	agent-detector-7	attack-1
17.00	agent-detector-10	attack-1
7.00	agent-detector-1	attack-1
18.00	agent-detector-6	attack-1
17.00	agent-detector-5	attack-1
18.00	agent-detector-10	attack-1
17.00	agent-detector-8	attack-1
19.00	agent-detector-6	attack-1

Close

Fig. 5. Main window of the monitor agent managing simulation procedure

Simulation is performed in two steps. At the first step (decision combining training step) each agent randomly selects N neighbors whose decisions it will further combine. Then, using computed contingency matrices, each agent assigns the weights to its neighbors to use them as neighbors' competence metrics. The sense of the selected measure is a conditional probability that particular neighbor produces a correct decision when agent computing combined decision detects an attack.

Testing procedure realized on meta–data testing sample uses two decision combining rules, "*Sum rule*" and "*Max rule*" [12]. Sum rule computes

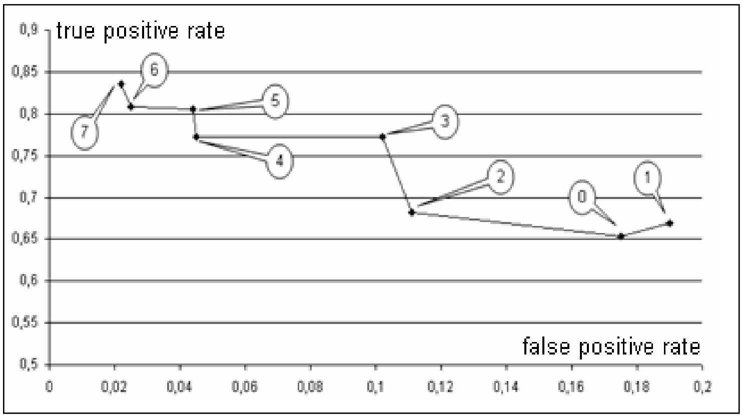


Fig. 6. ROC curve corresponding to decision combining using "Sum rule"

the sum of probabilities of correct decisions that may be one of two classes, "*Attack*" or "*Normal*". Corresponding weights in favor of each of these classes are computed as follows:

$$W_{Attak}(k) = \sum\nolimits_{i \in S(+)} w_i(k), \quad W_{Normal}(k) = \sum\nolimits_{i \in S(-)} w_i(k),$$

where $w_i(k)$ — the weight evaluating competence of the agent *i* from viewpoint of the agent *k*, $W_{Attack}(k)$ — the value of the "*Sum rule*" function voting in favor of the "*Attack*" class from the viewpoint of agent *k*, $W_{Normal}(k)$ — the value of the "*Sum rule*" function voting in favor of the "*Normal*" class, $S(+)$ — the set of agents voting in favor of the "*Attack*" class, and $S(-)$ — the set of agents voting in favor of the "*Normal*" class.

Fig. 6 presents ROC curve corresponding to decision combining using "Sum rule". In this curve, the *x-axis* corresponds to the *False Positive* rate while the *y-axis* corresponds to the *True Positiv*e rate. The marks in circles associated with the particular points of this curve indicate the number of neighbors whose decisions the decision combining agent takes into account. The presented ROC curve resulted from the averaging done over ROC curves of seven agents participating in simulation. For these particular simulation settings, the best P2P IDS performance corresponds to the case when seven neighbors are included in the agents' contact lists.

Analogous simulation was done using "Max rule"-based alert correlation that generated similar results.

7 Conclusions and Future Work

The main paper contribution is new architecture for open agent-based service-oriented IDS systems operating on P2P basis. The proposed architecture (Fig. 1) provides for cooperative performance of distributed security means supported by distributed meta-knowledge base implemented as an overlay network of instances of P2P agent platform set up on top of P2P networking provider, implemented in turn as overlay network set up on top of TCP/IP transport. Overlay network of P2P agent platform instances allows transparent semantic interaction of agents (including the agents of distributed security system) cooperating through security service exchange. The security system of such architecture can simply be populated with new agents, i.e. new security components when necessary. The resulting security system possesses such important properties as scalability, extendibility, efficiency and fault tolerance.

The problems associated with the design of P2P security system of the proposed architecture are analyzed. In this analysis, the main attention is paid to P2P training of security agents intended to provide for better quality of alert correlation aimed at more reliable detection of suspicious activity. Simplified P2P agent training for decision combining and testing software tool were developed. They were used in simulation-based experiments basically intended to highlight the essence of the training of P2P IDS agents to P2P decision combining and to exhibit potential problems and solutions.

Future work should concern enrichment of expressive capabilities of the distributed meta-knowledge base, further investigation of P2P decision combining procedures as well as joint operation of P2P security policy agents, P2P IDS agents and P2P agent platform being the core aspects of the proposed architecture.

References

1. AAFID http://www.cerias.purdue.edu/about/history/coast/projects/aafid.php
2. Xiao, R., Zheng, J., Wang, X., Xue, X.A.: A Novel Peer-to-Peer Intrusion Detection System Using Mobile Agents in MANETs. In: Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT-2005), pp. 441–445 (2005)
3. Asaka, M., Taguchi, A., Goto, S.: The Implementation of IDA: An Intrusion Detection Agent System. In: Proceedings of the 11th FIRST Conference 1999, Australia (1999)
4. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., Kargupta, H.: Distributed Data Mining in Peer-to-Peer Networks. IEEE Internet Computing special issue on Distributed Data Mining, 10(4), 18–26 (2006)
5. FIPA web site, http://www.fipa.org
6. FIPA P2P NA WG6: Functional Architecture Specification Draft 0.12. http://www.fipa.org/subgroups/ P2PNA-WG-docs/P2PNA-Spec-Draft0.12.doc
7. FIPA P2P Nomadic Agents Working Group (P2PNA WG6), http://www.fipa.org/subgroups/P2PNA-WG.html
8. Gorodetsky, V., Karsaev, O., Samoylov, V., Serebryakov, S.: P2P Agent Platform: Implementation and Testing. In: Proceedings of AP2PC Workshop at AAMAS 07, pp. 25–32 (2007)
9. Gorodetsky, V., Karsaev, O., Samoilov, V., Serebryakov, S.: Agent-based Service-Oriented Intelligent Networks for Distributed Classification. In: International Conference "Hybrid Information Technologies" (ICHIT-2006), pp. 224–233. IEEE Computer Press, Los Alamitos (2006)

10. Helmer, G.G., Wong, J.S.K., Honavar, V., Miller, L.: Intelligent agents for intrusion detection. In: Proceedings, IEEE Information Technology Conference, Syracuse, NY, September 1998, pp. 121–124. IEEE Computer Society Press, Los Alamitos (1998)
11. Kephart, J.: Multiagent Systems for Autonomic Computing. In: AAMAS 2007 (2007)
12. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
13. Lin, N., Marzullo, K., Masini, S.: Gossip versus Deterministic Flooding: Low Message Overhead and High Reliability for Broadcasting on Small Networks, Technical report CS1999-0637, http://citeseer.ist.psu.edu/563854.html
14. Ragsdale, D.J., Carver, C.A., Humphries, J.W., Pooch, U.W.: Adaptation Techniques for Intrusion Detection and Intrusion Response Systems. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Nashville, Tennessee, October 8-11, pp. 2344–2349 (2000)

An Interval Temporal Logic-Based Matching Framework for Finding Occurrences of Multi-event Attack Signatures

Elżbieta Nowicka[1] and Marcin Zawada[2]

[1] Chair of Computer Systems and Networks
[2] Institute of Mathematics and Computer Science
Wroclaw University of Technology
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
elzbieta.nowicka@plusnet.pl, marcin.zawada@pwr.wroc.pl

Abstract. Temporal logic has the potential to become a powerful mechanism for both modeling and detection of attack signatures. But, although recently some very expressive attack representations and on-line monitoring tools have been proposed, such tools still suffer from a lack of sufficiently precise detection mechanisms. In particular, they can report only the existence of an attack instance and cannot locate precisely its occurrence in a monitored event stream. Precise location is a key to enabling proper verification and identification of an attack. In this paper, we propose a formal framework for multi-event attack signature detection, based on Interval Temporal Logic. Our framework formalizes the problem of finding the localizations of a number types of attack signature occurrences: the first, all, k-insertion and the shortest one. In our approach, we use the existing run-time monitoring mechanism developed for the Eagle specification, and extend it by special rules to enable such localization tasks. Our approach works on-line, and our initial results demonstrate the effectiveness and efficiency of the proposed approach.

Keywords: Intrusion detection, attack signatures, interval temporal logic, approximate pattern matching.

1 Introduction

Intrusion detection systems (IDSs) are becoming more and more necessary protection against computer and network attacks. Generally, these systems are either anomaly-based or misuse-based [5]. As misuse-based IDS systems produce fewer false alarms, they are more widely used in practice. Such systems can identify known attacks using *attack signatures* that are special patterns used to find suspicious events that occur in a system or a network. Since, in general, the problem of attack signature detection has been converted into a pattern matching problem [3], many pattern-modeling and pattern-matching techniques have been proposed. Among patterns used we find *strings, classes of characters* [6], *regular expressions with back referencing* applied in the GASSATA system [8], *regular expressions with variables* used in the REE language [18], *extended* *regular expressions* and *Colored Petri-Nets* considered in [5] and applied in the IDIOT system, and *acyclic directed graphs* used in the MuSigs specification [7] developed for the ARMD system. However, as nowadays more and more attack techniques involve complex, time-correlated sequences of actions, these patterns turn out to be either insufficient or difficult to model effectively the properties of the sophisticated multi-event attack signatures and to enable their detection.

Recently, expressive and more convenient signature representations have been proposed, such as *chronicle*-based ADeLe applied in the GnG system [17] and various *temporal logic* expressions used in tools such as logWeaver [16], Orchids [14], Monid [12] and SDIDS [2], and in signature attack models: ISITL [15] and *Sig*ITL [13]. However, they still need improvement, particularly in the construction of the matching algorithms to be used. Chronicle-based GnG uses a nondeterministic automaton-based detection mechanism, which makes this approach insufficiently efficient to detect more complex signatures, especially those which are temporally ordered.

In turn, most existing temporal logic-based IDS tools are not fine enough to permit more complex multi-event signatures to be located precisely in event streams. Indeed, from the point of view of a system administrator, such tools at present work as "black boxes", because their output usually consists only of a statement whether or not an attack signature has been found. This makes the detection result insufficiently reliable to ensure that the identification of attack traces is sound, so further investigation is required, which makes the whole detection process much slower. This remains a major obstacle to effective application of these techniques.

However, despite this limitation, at least two main advantages of temporal logic-based approaches, such as the Eagle logic-based Monid, make them still very promising. First, the Eagle logic [1] is highly expressive, since it supports a large variety of existing logics and patterns. Second, it provides efficient real-time monitoring. Since these features are of great importance to intrusion detection systems, we argue that they represent good reasons to be interested in extending the approach applied in Monid in order to tailor better its capability to handle attack signature localization tasks. Additional motivation is also the encouraging initial work done in this direction that can be found in [16] used in logWeaver and recently also in Orchids. However, the authors considered formally only one type of a signature match (the shortest match) which can be insufficient. Moreover, this work was carried out for the case of a much less expressive logic than Eagle, and so the logWeaver specification does not allow the modeling of more complex temporal features of multi-event signatures.

In this paper, we extend the above results, concentrating on the general study of the problem. Namely, we propose a formal matching framework to create an unambiguous temporal logic-based specification for matching complex multi-event signatures. There are two main challenges in defining such a framework. Firstly, in the literature there is no general agreement on how much information about potential signature matches should be reported during the detection process. As a result, there are many different approaches to this problem such as reporting the first match, all matches, the k-insertion, shortest or time-limited match (see e.g. [4,6,16,17]). Secondly, to date there appears to be no formal definition of what it means to locate precisely a pattern in the form of a temporal formula in some data sequence in a manner appropriate for various types of signature matches. Our aim is therefore to develop general and formal principles to address this problem and explore its practical implementation.

In our approach presented in this paper, we use our own previous result, the *Sig*ITL model [13] based on the well-studied and theoretically sound Interval Temporal Logic (ITL) [9,10] which properties are embedded in the EAGLE logic. As we have showed in [13], the *Sig*ITL model provides a convenient and flexible tool for considering problems of modeling of complex temporal features of multi-event signatures. In this paper, we show that it can also support defining a matching framework to detect such signatures.

The structure of this paper is as follows. Section 2 reviews briefly existing approaches to attack signature match locating problems. In Section 3 we formally specify the signature matching problem in our *Sig*ITL model. In Section 4 we outline our matching algorithm. Section 5 describes our matching framework with initial experimentation in Section 6. Conclusions are in Section 7.

Because of space constraints proofs are omitted from the paper.

2 Locating Matches of Attack Signatures

Pattern matching approach to attack signature detection was initially introduced in [3] and can be viewed as an *on-line multiple approximate* pattern-matching problem [6]. In this approach, a *signature match* is considered as an occurrence of an attack signature to be located in some sequence of events (from system logs or network traffic). In case of multi-event signatures, as it is quite common that an intruder intentionally inserts some additional events or performs an attack during a long time period to avoid detection, matching mechanism allows insertions of irrelevant events between events that belong to a detected signature. In general, the problem is considered as either searching a *subsequence* or a subsequence with constraints (e.g. insertion distance [6] or timing constraints [4,17]). In this context, various approaches to the signature match locating problem can be identified in the literature:

- *Existence of a match*, i.e. only the fact of an occurrence of a signature match is reported. This approach, used in e.g. GASSATA and in most of temporal logic-based IDSs, requires manual localization of detected signatures.
- *First match*, i.e. only the first encountered signature match is reported. This is the most common approach, but may result in ignoring other real matches.
- *All matches*, i.e. all found signature match instances are reported. This feature, supported by e.g. GnG, guarantees not missing any potential match, but it can also need an exhaustive search, since maintaining and checking many partial matches are required. Another disadvantage is that there may be overlap and redundancy among the matches reported.

- *k-insertion match*, i.e. a signature match is reported, where at most k inserted events are allowed in an occurrence of a match. This kind of match was considered in [6]. This approach mainly addresses short-term attacks.
- *Shortest match*, i.e. minimal-length match is reported. This kind of match has been proposed in [16] and considered as the most significant. It is used in logWeaver and recently also in ORCHIDS. However, similarly to the first match case, it may also lead to a risk of missing relevant events.
- *Time-limited match*, i.e. a match where the time delay between certain events is specified. This feature is supported by e.g. IDIOT, GnG and SDIDS systems. It has a meaning in case of automated attacks (e.g. worms).

3 Matching *Sig*ITL Signatures

In this section, we formally define a multi-event signature matching problem in our *Sig*ITL model introduced in [13] and based on a slightly modified subset of the Interval Temporal Logic (ITL) [9].

3.1 *Sig*ITL Model

Let us fix a finite alphabet Σ. Let elements of Σ refer to all possible types of events that can occur in system logs or in network traffic. Given an alphabet Σ, an **event sequence** $\sigma = e_1 e_2 \ldots e_n$, where $e_i \in \Sigma$ for all $1 \leq i \leq n$, is an ordered sequence of events sorted with respect to their timestamps.

In order to put our consideration in a temporal logic context, we recall briefly the definition of the *Sig*ITL (details can be found in [13]). Let P be a set of propositions. Let each element from Σ correspond to some propositional property. Let $\Box$, $\Diamond$, $\bigcirc$ and $;$ refer to temporal operators such as *always*, *sometime*, *at the next moment* and *chomp*. For technical clarity, we should note here that the *chomp* operator is used in our *Sig*ITL model instead of *chop*, the original ITL operator. In fact, *chomp* is a slight variant of *chop* and expresses the conventional *concatenation* [11]. Our *Sig*ITL model is formally defined below.

Syntax. *Sig*ITL formulas are defined inductively as follows:

$$\varphi ::= P \,|\, \top \,|\, \bot \,|\, \neg\varphi \,|\, \varphi \wedge \varphi \,|\, \varphi \vee \varphi \,|\, \Box\varphi \,|\, \Diamond\varphi \,|\, \bigcirc\varphi \,|\, \varphi\,;\,\varphi.$$

Semantics. Let σ be a finite sequence of events of length n. By $\sigma(i)$ we denote i^{th} event of σ. The term $\sigma[i,j]$ denotes the interval of a sequence σ from position i to position j, where $1 \leq i \leq j \leq n$. If a *Sig*ITL formula φ holds in the interval $\sigma[i,j]$, we denote this by $\sigma[i,j] \vDash \varphi$, and say that an interval $\sigma[i,j]$ satisfies a formula φ. Let φ, ψ be *Sig*ITL formulas. Given an event sequence σ, satisfaction in *Sig*ITL is as follows:

$$\begin{array}{lll} \sigma[i,j] \vDash p & \text{iff} & \sigma(i) = p, \quad p \in P \\ \sigma[i,j] \vDash \top & \text{iff} & \texttt{true} \\ \sigma[i,j] \vDash \bot & \text{iff} & \texttt{false} \\ \sigma[i,j] \vDash \neg\varphi & \text{iff} & \neg(\sigma[i,j] \vDash \varphi) \end{array}$$

$$\begin{array}{lll}
\sigma[i,j] \models \varphi \wedge \psi & \text{iff} & \sigma[i,j] \models \varphi \texttt{ and } \sigma[i,j] \models \psi \\
\sigma[i,j] \models \varphi \vee \psi & \text{iff} & \sigma[i,j] \models \varphi \texttt{ or } \sigma[i,j] \models \psi \\
\sigma[i,j] \models \Box\varphi & \text{iff} & (\forall k \in [i,j])(\sigma[k,j] \models \varphi) \\
\sigma[i,j] \models \Diamond\varphi & \text{iff} & (\exists k \in [i,j])(\sigma[k,j] \models \varphi) \\
\sigma[i,j] \models \bigcirc\varphi & \text{iff} & i < j \texttt{ and } \sigma[i+1,j] \models \varphi \\
\sigma[i,j] \models \varphi\,;\psi & \text{iff} & (\exists k \in [i,j))(\sigma[i,k] \models \varphi \texttt{ and } \sigma[k+1,j] \models \psi).
\end{array}$$

Let $\varphi \in SigITL$ represent a multi-event attack signature and σ be an event sequence. Classically, a given temporal formula is only regarded as either satisfied or not. So, if σ satisfies φ, i.e. $\sigma[1,n] \models \varphi$, we know only that a match of a signature φ **exists** in a sequence σ, and not otherwise. However, we would like to know not only *whether* a certain match exists or not, but also *where* such a match exactly occurs within this sequence. Therefore, in the sequel, we will extend the problem of *Sig*ITL formula satisfaction into the problem of *Sig*ITL formula matching.

First let us introduce some assumptions we have made. The first assumption is a result of a simple but important observation in attack detection: if an attack is detected, no other later event can invalidate this detection. So, in particular, if a match of a signature φ exists in an interval $\sigma[1,j]$, where $1 \leq j \leq n$, then this fact is also true on the whole sequence $\sigma[1,n]$. Therefore we consider only *Sig*ITL formulas that are *monotonic* and formally define this property as follows.

Definition 1. *Formula $\varphi \in SigITL$ is **monotonic**, if*

$$(\forall i,j,k \in \{1,2,\ldots,n\})(((\sigma[i,j] \models \varphi) \wedge ([i,j] \subseteq [i,k])) \Rightarrow \sigma[i,k] \models \varphi).$$

Moreover, we assume that each monotonic formula is in a *negative normal form* (NNF). From now on, we use the term *Sig*ITL* to represent the *Sig*ITL model with only monotonic formulas in NNF form.

3.2 *Sig*ITL* Matching Model

We introduce now our formal definition of an occurrence of *Sig*ITL* signature match in an event sequence. Let $\varphi \in Sig\text{ITL}^*$ be a multi-event signature and $\sigma = e_1 e_2 \ldots e_n$ be an event sequence. If σ satisfies φ, i.e. $\sigma[1,n] \models \varphi$, then intuitively by an *occurrence* of a signature φ in an event sequence σ, we mean the information about which events of φ are satisfied in which positions of σ. Clearly, it is a *set of events* from φ with associated *indices* of σ, in which matched events occur.

Let us introduce this problem formally. Notice that $\sigma = e_1 e_2 \ldots e_n$ can be viewed as a set $s_\sigma = \bigcup_{i=1}^{n}\{(\sigma(i), i)\}$. Let $\mathcal{L}(\sigma) = P(\bigcup_{i=1}^{n}\{(\sigma(i), i)\})$ be a collection of all subsets of s_σ. Then of course, any set $L = \{(e_{i_1}, i_1), (e_{i_2}, i_2), \ldots, (e_{i_m}, i_m)\} \in \mathcal{L}(\sigma)$ can be interpreted as a **subsequence** of σ. We say that a set $L \in \mathcal{L}(\sigma)$ is an **occurrence** of a signature φ in an event sequence $\sigma = e_1 e_2 \ldots e_n$ if the following condition holds

$$(\forall \sigma' \in \Delta(L))(\sigma' \models \varphi) \wedge (\forall 1 \leq k \leq m)(\exists \sigma' \in \Delta(L \setminus \{(e_{i_k}, i_k)\}))(\sigma' \not\models \varphi), \quad (1)$$

where an operator $\varDelta\colon P(\varSigma \times \{1, 2, \ldots, n\}) \rightarrow P((\varSigma \cup \triangle)^{\{1,2,\ldots,n\}})$ transforms a set L into a set of event sequences as follows:

$$\varDelta(L) = \{e'_1 \ldots e'_{i_1-1} e_{i_1} e'_{i_1+1} \ldots e'_{i_m-1} e_{i_m} e'_{i_m+1} \ldots e'_n \colon (\forall i)(e'_i \in \{e_i, \triangle\})\},$$

where $\triangle$ denotes any symbol which does not belong to $\varSigma$.

Before we go any further, we need to introduce some more terminology. First, we define the following strict orderings $<_{lex}$, $\prec$ and $<_{fst}$ on $\mathcal{L}(\sigma)$. For every $L = \{(e_{i_1}, i_1), \ldots, (e_{i_l}, i_l)\} \in \mathcal{L}(\sigma)$ and $L' = \{(e_{i'_1}, i'_1), \ldots, (e_{i'_{l'}}, i'_{l'})\} \in \mathcal{L}(\sigma)$ such that $1 \leq i_1 < i_2 < \ldots < i_l \leq n$ and $1 \leq i'_1 < i'_2 < \ldots < i'_{l'} \leq n$, we say that $L <_{lex} L'$ if and only if $(\exists 1 \leq j \leq \min(l, l'))(i_1 = i'_1 \wedge \ldots \wedge i_{j-1} = i_{j-1} \wedge i_j < i'_j)$. Similarly, following the notation from [16], we say that $L \prec L'$ if and only if $([i_1, i_l] \subsetneq [i'_1, i'_{l'}]) \vee ([i_1, i_l] = [i'_1, i'_{l'}] \wedge L <_{lex} L')$. Besides, we say that $L <_{fst} L'$ if and only if $L \prec L' \vee (L \nprec L' \wedge L <_{lex} L')$.

We also define an insertion distance in our *Sig*ITL* model. Let $L = \{(e_{i_1}, i_1), (e_{i_2}, i_2), \ldots, (e_{i_l}, i_l)\} \in \mathcal{L}(\sigma)$ be any occurrence such that $1 \leq i_1 < i_2 < \ldots < i_l \leq n$, then an **insertion distance** we define as follows: $id(L, \sigma) = i_l - i_1 - l + 1$ if $L \neq \emptyset$ and ∞ otherwise.

We show now that our foregoing definition of an occurrence of *SigITL** signature can be naturally extended for various match locating problems, namely for all matches, the first, k-insertion and shortest match, which we formally define below.

Definition 2. *Let $\varphi \in SigITL^*$ be a signature and $\sigma = e_1 e_2 \ldots e_n$ be an event sequence. Then we can define the following types of signature occurrences in our SigITL* matching model:*

- ***All Matches.*** *All occurrences of a signature φ, denoted by* $\textsc{AllMatch}(\varphi, \sigma) \subseteq \mathcal{L}(\sigma)$*, is defined as a collection of all sets $L \in \mathcal{L}(\sigma)$ which are occurrences of a signature φ in σ, i.e. such sets $L \in \mathcal{L}(\sigma)$ for which condition (1) holds.*
- ***First Match.*** *Let j be such a natural number that $\sigma[1, j-1] \nvDash \varphi \wedge \sigma[1, j] \vDash \varphi$. Then the first occurrence of a signature φ we call a set $L \in \textsc{AllMatch}(\varphi, \sigma[1, j])$ if $\neg(\exists L' \in \textsc{AllMatch}(\varphi, \sigma[1, j]))(L' <_{fst} L)$, and we denote it as $\textsc{FirstMatch}(\varphi, \sigma) = L$.*
- ***k-Match.*** *Let $L \in \textsc{AllMatch}(\varphi, \sigma)$ be any occurrence of a signature φ in σ. Let k be the maximal number of insertions that are allowed in an occurrence L. Then an occurrence of a signature φ with at most k insertions is defined as: $k\text{-}\textsc{Match}(\varphi, \sigma) = \{L \in \textsc{AllMatch}(\varphi, \sigma)\colon id(L, \sigma) \leq k)\}$.*
- ***Shortest Match.*** *If the number of insertions that are allowed in an occurrence of a signature φ has to be minimal, it is defined as:* $\textsc{ShortestMatch}(\varphi, \sigma) = \{L \in \textsc{AllMatch}(\varphi, \sigma) : \neg(\exists L' \in \textsc{AllMatch}(\varphi, \sigma))(L' \prec L)\}$.

The following example illustrates how various types of multi-event signature occurrences are identified in our *Sig*ITL* matching model.

Example 1. Let us consider an attack scenario in which an attacker exploits an old `sendmail` bug in Unix and gains the root privilege. This scenario consists of the following steps:

```
A  cp /bin/sh /usr/spool/mail/root
B  chmod 4755 /usr/spool/mail/root
C  touch x
D  mail root < x
```

There can be, however, a few variants of the above attack scenario (see [13] for details). Here, for illustration, we consider its partially ordered *Sig*ITL* signature: $\varphi = (\Diamond(\mathsf{A}\,;\,\mathsf{B}) \wedge \Diamond \mathsf{C})\,;\,\mathsf{D}$. It specifies that `cp` must precede `chmod`, `chmod` must precede `mail`, and `touch` must occur before `mail`. Let $\sigma = \mathsf{AABACBD}$. Then we get $\textsc{AllMatch}(\varphi, \sigma) = \{L_1, L_2, L_3, L_4, L_5\}$, where $L_1 = \{(\mathsf{A}, 1), (\mathsf{B}, 3), (\mathsf{C}, 5), (\mathsf{D}, 7)\}, L_2 = \{(\mathsf{A}, 1), (\mathsf{C}, 5), (\mathsf{B}, 6), (\mathsf{D}, 7)\}, L_3 = \{(\mathsf{A}, 2), (\mathsf{B}, 3), (\mathsf{C}, 5), (\mathsf{D}, 7)\}$, $L_4 = \{(\mathsf{A}, 2), (\mathsf{C}, 5), (\mathsf{B}, 6), (\mathsf{D}, 7)\}$ and $L_5 = \{(\mathsf{A}, 4), (\mathsf{C}, 5), (\mathsf{B}, 6), (\mathsf{D}, 7)\}$. As we have $(\forall i \in \{2, 3, 4, 5\})(L_1 <_{fst} L_i)$, so $\textsc{FirstMatch}(\varphi, \sigma) = L_1$. Let $k = 2$. Since $id(L_1, \sigma) = id(L_2, \sigma) = 3, id(L_3, \sigma) = id(L_4, \sigma) = 2, id(L_5, \sigma) = 0$, so we get $2\text{-}\textsc{Match}(\varphi, \sigma) = \{L_3, L_4, L_5\}$. Because $(\forall i \in \{1, 2, 3, 4\})(L_5 \prec L_i)$, so $\textsc{ShortestMatch}(\varphi, \sigma) = \{L_5\}$.

4 Algorithm *Sig*ITL*-Match

In this section, we introduce our matching mechanism that handles the localization problem of various types of multi-event signature occurrences defined in *Sig*ITL* matching model, proposed in the previous section. Our computation approach is based on the rewriting system proposed for monitoring Eagle logic formulas. But although, some of our basic rules are similar to the rules proposed by Barringer *et al.* for the Eagle logic [1], we largely extend them to provide the precise mechanism which not only detects the existence of signature matches, but also precisely locates detected signatures in an event sequence.

4.1 Basic Notation

Before presenting our algorithm in detail, let us introduce some more terminology. Let $\Sigma = \{\alpha_1, \alpha_2, \ldots, \alpha_{|\Sigma|}\}$. At first we need to extend the existing set P of variables. Therefore, we define an auxiliary set of special variables Φ which represents all possible appearances of events in the event sequence σ as follows: $\Phi = \bigcup_{i=1}^{n} \{l_\alpha^i : \alpha \in \Sigma\} \cup \{l^0\}$. We say that a variable $l_\alpha^i \in \Phi$ represents an appearance of an event α in the event sequence σ at a position i, i.e. $\alpha = \sigma(i)$. Besides, we define a function $\xi\colon \Sigma \times \{1, 2, \ldots, n\} \to \Phi$ as follows: $(\forall \alpha \in \Sigma)(\forall 1 \le i \le n)(\xi(\alpha, i) = l_\alpha^i)$. We would like to extend the definition of ξ for all subsets of $\Sigma \times \{1, 2, \ldots, n\}$. We proceed as follows. Let $L = \{(e_1, i_1), (e_2, i_2), \ldots, (e_m, i_m)\} \in P(\Sigma \times \{1, 2, \ldots, n\})$. Then, we have

$$\xi(L) = \{\xi(e_1, i_1), \xi(e_2, i_2), \ldots, \xi(e_m, i_m)\} = \{l_{e_1}^{i_1}, l_{e_2}^{i_2}, \ldots, l_{e_m}^{i_m}\}.$$

Notice that the extended function ξ is bijective. We also introduce partial and linear orders on the set Φ of variables. We define a strict partial order $\prec$ as follows: $(\forall e_1, e_2 \in \Sigma)(\forall x, y \in \{1, 2, \ldots, j\})(l_{e_1}^x \prec l_{e_2}^y \equiv e_1 = e_2 \wedge x < y)$ and a linear order $\leqslant$ as follows: $(\forall e_1, e_2 \in \Sigma)(\forall x, y \in \{1, 2, \ldots, j\})(l_{e_1}^x \leqslant l_{e_2}^y \equiv x \le y)$.

$$
\begin{array}{lll}
\textbf{match}\langle\top, e\rangle & \rightarrow \top & \\
\textbf{match}\langle\bot, e\rangle & \rightarrow \bot & \\
\textbf{match}\langle l, e\rangle & \rightarrow l, \quad \text{if } l \in \Phi & \\
\textbf{match}\langle p, e\rangle & \rightarrow \begin{cases} l_p^i \in \Phi, \text{ if } p = e \wedge p \notin \Phi \\ \bot, \qquad \text{if } p \neq e \wedge p \notin \Phi \end{cases} & \\
\textbf{match}\langle \neg\varphi, e\rangle & \rightarrow \neg\textbf{match}\langle\varphi, e\rangle & \\
\textbf{match}\langle \varphi \wedge \psi, e\rangle & \rightarrow \textbf{match}\langle\varphi, e\rangle \wedge \textbf{match}\langle\psi, e\rangle & \\
\textbf{match}\langle \varphi \vee \psi, e\rangle & \rightarrow \textbf{match}\langle\varphi, e\rangle \vee \textbf{match}\langle\psi, e\rangle & \\
\textbf{match}\langle \bigcirc\varphi, e\rangle & \rightarrow \varphi & \\
\textbf{match}\langle \Diamond\varphi, e\rangle & \rightarrow \textbf{match}\langle\varphi, e\rangle \vee \textbf{match}\langle\bigcirc\Diamond\varphi, e\rangle & \\
\textbf{match}\langle \Box\varphi, e\rangle & \rightarrow \textbf{match}\langle\varphi, e\rangle \wedge \textbf{match}\langle\bigcirc\Box\varphi, e\rangle & \\
\textbf{match}\langle \varphi\,;\,\psi, e\rangle & \rightarrow \text{if } \textbf{value}\langle\varphi\rangle = \texttt{true} \text{ then} & \\
 & \qquad \text{let } \eta = \textbf{conj}(\textbf{extract}\langle\textbf{remove}\langle\varphi\rangle\rangle) \text{ in} & \\
 & \qquad\quad (\textbf{match}\langle\varphi, e\rangle\,;\,\psi) \vee (\eta \wedge \textbf{match}\langle\psi, e\rangle) & \\
 & \quad \text{else } (\textbf{match}\langle\varphi, e\rangle\,;\,\psi). &
\end{array}
$$

Fig. 1. Match rules

$$
\begin{array}{lll}
\textbf{value}\langle\top\rangle & = \texttt{true} & \\
\textbf{value}\langle\bot\rangle & = \texttt{false} & \\
\textbf{value}\langle l\rangle & = \texttt{true}, & \text{if } l \in \Phi \\
\textbf{value}\langle p\rangle & = \texttt{false}, & \text{if } p \notin \Phi \\
\textbf{value}\langle\neg\varphi\rangle & = \neg\textbf{value}\langle\varphi\rangle &
\end{array}
\qquad
\begin{array}{ll}
\textbf{value}\langle\varphi \wedge \psi\rangle & = \textbf{value}\langle\varphi\rangle \texttt{ and } \textbf{value}\langle\psi\rangle \\
\textbf{value}\langle\varphi \vee \psi\rangle & = \textbf{value}\langle\varphi\rangle \texttt{ or } \textbf{value}\langle\psi\rangle \\
\textbf{value}\langle\bigcirc\varphi\rangle & = \texttt{false} \\
\textbf{value}\langle\Diamond\varphi\rangle & = \texttt{false} \\
\textbf{value}\langle\Box\varphi\rangle & = \texttt{true} \\
\textbf{value}\langle\varphi\,;\,\psi\rangle & = \textbf{value}\langle\varphi\rangle \texttt{ and } \textbf{value}\langle\psi\rangle.
\end{array}
$$

Fig. 2. Value function

4.2 Matching Mechanism

We are ready to present our matching algorithm. Our algorithm, called SigITL*-Match, works as follows. At first a signature φ, which is represented as SigITL* formula, is rewritten by **match** rewrite rules into another formula φ' (see Fig. 1). Our **match** rules refer to the *eval* function from Eagle. They, however, work according to the semantics of SigITL* and in our matching approach, we replaced the rule of evaluation of variables with two our new rules: $\textbf{match}\langle l, e\rangle$ and $\textbf{match}\langle p, e\rangle$ to enable remembering events from a signature φ that have been matched. For the same reason we have modified a rule $\textbf{match}\langle\varphi\,;\,\psi, e\rangle$, because at each time if the $\textbf{value}\langle\varphi\rangle = \texttt{true}$, we also need to remember matched events of φ. The **conj** function, used in this rule, creates a conjunction of variables of Φ from a set of event appearances. For example, for a set $\{l_{e_1}^{i_1}, l_{e_2}^{i_2}, \ldots, l_{e_m}^{i_m}\}$, we obtain a formula $l_{e_1}^{i_1} \wedge l_{e_2}^{i_2} \wedge \ldots \wedge l_{e_m}^{i_m}$. Formally, the evaluation of a formula φ on an event $e = \sigma(i)$ of an event sequence σ results in another formula $\textbf{match}\langle\varphi, e\rangle$ with the property that $\sigma[i, n] \vDash \varphi$ if and only if $\sigma[i+1, n] \vDash \textbf{match}\langle\varphi, e\rangle$. This property is formalized in the following theorem.

Theorem 1. *Let $\varphi \in$ SigITL* be a signature and $\sigma = e_1 e_2 \ldots e_n$ be an event sequence. Then by applying the rules* **match** *on the formula φ and an event $e_i = \sigma(i)$, for $i \in \{1, 2, \ldots, n\}$, we obtain a formula $\varphi' = \textbf{match}\langle\varphi, e_i\rangle$ such that*

$$\sigma[i,n] \vDash \varphi \Leftrightarrow \sigma[i+1,n] \vDash \varphi'. \quad (2)$$

After applying **match** rules, we apply **value** function (see Fig. 2). Our function **value** is a modified version of the *value* function from EAGLE. Similarly, like in case of **match** rules, we introduced **value**$\langle l \rangle$ and **value**$\langle p \rangle$. The **value** function, when applied on φ, at the end of an event sequence σ, returns `true` if and only if signature φ occurs in σ and `false` otherwise. Thus, given an event sequence $\sigma = e_1 e_2 \dots e_n$, a signature φ is said to be satisfied by σ if and only if $\mathbf{value}\langle \mathbf{match}\langle \dots \mathbf{match}\langle \mathbf{match}\langle \varphi, e_1 \rangle, e_2 \rangle \dots, e_n \rangle \rangle$ is `true`.

Corollary 1. *Condition (2) can be generalized as follows:*

$$\sigma[1,n] \vDash \varphi \Leftrightarrow \mathbf{value}\langle \mathbf{match}\langle \dots \mathbf{match}\langle \mathbf{match}\langle \varphi, e_1 \rangle, e_2 \rangle \dots, e_n \rangle \rangle = \mathtt{true}.$$

At this stage, we introduce a set of new rules and a procedure to enable precise localisation of an occurrence of a signature φ in case a formula φ holds along a sequence σ (i.e. a signature match exists). Our basic idea for this task is to keep the information of a potential signature occurrence within the transformed formula. For this task we introduce **remove** rules and a function **extract** (see Fig. 3 and 4) which allow us to extract occurrences of a signature directly from the transformed formula. Notice that basically, **remove** rules are similar to the function **value** and are also used at the end of an event sequence σ (just after applying **value** function). However, their role is different. The task of **remove** rule is to remove temporal operators from a formula. This operation gives us a state formula that has only propositional operators $\wedge$, $\vee$, $\neg$ and variables from the set Φ. Afterwards, on such a state formula, we apply an **extract** function to get a set of variables which indeed represents an occurrence of a signature φ. We can formalize it by the following theorem.

Theorem 2. *Let $\varphi \in SigITL^*$ be a signature and $\sigma = e_1 e_2 \dots e_n$ be an event sequence. Then*

$$E = \mathbf{extract}\langle \mathbf{remove}\langle \mathbf{match}\langle \dots \mathbf{match}\langle \mathbf{match}\langle \varphi, e_1 \rangle, e_2 \rangle \dots, e_n \rangle \rangle \rangle$$

and $\xi^{-1}(E) \in \mathcal{L}(\sigma)$ is an occurrence of φ in σ.

5 *Sig*ITL* Matching Framework

In this section, we describe how various types of signature occurrences defined in our *Sig*ITL* model in Section 3 are identified by our matching algorithm presented in previous section.

5.1 All Matches

We start by introducing a procedure which allows us to find all occurrences of a signature φ in an event sequence σ, i.e. to find a set ALLMATCH(φ, σ). We proceed as follows. Let φ_i be a signature φ after i steps of evaluation, i.e. $\varphi_i =$

$$\begin{array}{llll}
\mathbf{remove}\langle\top\rangle \to \top & & \mathbf{remove}\langle\varphi \wedge \psi\rangle \to \mathbf{remove}\langle\varphi\rangle \wedge \mathbf{remove}\langle\psi\rangle \\
\mathbf{remove}\langle\bot\rangle \to \bot & & \mathbf{remove}\langle\varphi \vee \psi\rangle \to \mathbf{remove}\langle\varphi\rangle \vee \mathbf{remove}\langle\psi\rangle \\
\mathbf{remove}\langle l\rangle \to l, & \text{if } l \in \Phi & \mathbf{remove}\langle\bigcirc\varphi\rangle \to \bot \\
\mathbf{remove}\langle p\rangle \to \bot, & \text{if } p \notin \Phi & \mathbf{remove}\langle\Diamond\varphi\rangle \to \bot \\
\mathbf{remove}\langle\neg\varphi\rangle \to \neg\mathbf{remove}\langle\varphi\rangle & & \mathbf{remove}\langle\Box\varphi\rangle \to \top \\
 & & \mathbf{remove}\langle\varphi\,;\,\psi\rangle \to \mathbf{remove}\langle\varphi\rangle \wedge \mathbf{remove}\langle\psi\rangle.
\end{array}$$

Fig. 3. Remove rules

$$\begin{array}{ll}
\mathbf{extract}\langle\top\rangle = \{l^0\} & \mathbf{extract}\langle\varphi \wedge \psi\rangle = merge_1(\mathbf{extract}\langle\varphi\rangle, \mathbf{extract}\langle\psi\rangle) \\
\mathbf{extract}\langle\bot\rangle = \emptyset & \mathbf{extract}\langle\varphi \vee \psi\rangle = merge_2(\mathbf{extract}\langle\varphi\rangle, \mathbf{extract}\langle\psi\rangle) \\
\mathbf{extract}\langle l_e^i\rangle = \{l_e^i\} & \mathbf{extract}\langle\neg l_e^i\rangle = \emptyset.
\end{array}$$

```
$merge_1(M_1, M_2) =$
1. if $M_1 = \emptyset$ or $M_2 = \emptyset$ then return $\emptyset$
2. $M := M_1 \cup M_2$
3. while $(\exists l, l' \in M)(l \prec l')$
4.     $M := M \setminus l'$
5. return $M$
```

```
$merge_2(M_1, M_2) =$
1. if $M_1 = \emptyset$ and $M_2 = \emptyset$ then return $\emptyset$
2. if $M_1 = \emptyset$ then return $M_2$
3. if $M_2 = \emptyset$ then return $M_1$
4. if $(\forall l \in M_1)(\exists l' \in M_2)(l \leqslant l')$ then return $M_1$
5. else return $M_2$
```

Fig. 4. Extract function

$\mathbf{match}\langle \ldots \mathbf{match}\langle\mathbf{match}\langle\varphi, e_1\rangle, e_2\rangle \ldots, e_i\rangle$. At each step i, if in the formula φ_i there is some variable $p \in P$ that has matched an event e_i (here a function **check** is applied), then we additionally create a new instance φ_i' and evaluate the formula φ on an event $\triangle$, i.e. $\varphi_i' = \mathbf{match}\langle \ldots \mathbf{match}\langle\mathbf{match}\langle\varphi, e_1\rangle, e_2\rangle \ldots, \triangle\rangle$. To find out if some event has been matched, we use a new introduced function **check** (see Fig. 5). The function $\mathbf{check}\langle\varphi, e\rangle$ returns `true` if some variable in a formula φ has matched an event e and `false` otherwise.

Let $\varphi \in SigITL^*$ be a signature and $\sigma = e_1 e_2 \ldots e_n$ be an event sequence. We define formally the above described procedure as follows:

$$\Psi_1 = \{\varphi\},$$
$$\Psi_{i+1} = \{\mathbf{match}\langle\psi, e_i\rangle : \psi \in \Psi_i\} \cup \{\mathbf{match}\langle\psi, \triangle\rangle : \psi \in \Psi_i \wedge \mathbf{check}\langle\psi, e_i\rangle = \mathtt{true}\}.$$

Then, we define a set AllMatch* as follows:

$$\text{AllMatch}^*(\varphi, \sigma) = \{\mathbf{extract}\langle\mathbf{remove}\langle\psi\rangle\rangle : \psi \in \Psi_n\}.$$

Corollary 2. *Let $\varphi \in SigITL^*$ be a signature and $\sigma = e_1 e_2 \ldots e_n$ be an event sequence. Then* AllMatch$(\varphi, \sigma) =$ AllMatch$^*(\varphi, \sigma)$.

5.2 First Match, k-Match and Shortest Match

Notice that the set Ψ_i can grow exponentially. Since reporting all occurrences of a given signature can become computationally expensive, therefore, in practice we usually do not want to find the set AllMatch(φ, σ), but we could prefer to reduce the problem to finding the first occurrence or occurrences that satisfy some constraints.

$$
\begin{array}{llll}
\mathbf{check}\langle \top, e\rangle &= \mathtt{false} & & \mathbf{check}\langle \bot, e\rangle = \mathtt{false} \\
\mathbf{check}\langle l, e\rangle &= \mathtt{false}, & \text{if } l \in \Phi & \mathbf{check}\langle \neg l, e\rangle = \mathtt{false}, \quad \text{if } l \in \Phi \\
\mathbf{check}\langle p, e\rangle &= \mathtt{true}, & \text{if } p = e \text{ and } p \notin \Phi & \mathbf{check}\langle \bigcirc\varphi, e_i\rangle = \mathtt{false} \\
\mathbf{check}\langle \neg p, e\rangle &= \mathtt{false}, & \text{if } p = e \text{ and } p \notin \Phi & \mathbf{check}\langle \Diamond\varphi, e\rangle = \mathtt{false} \\
\mathbf{check}\langle \varphi \wedge \psi, e\rangle &= \mathbf{check}\langle \varphi, e\rangle \ \mathtt{or}\ \mathbf{check}\langle \psi, e\rangle & & \mathbf{check}\langle \Box\varphi, e\rangle = \mathtt{false} \\
\mathbf{check}\langle \varphi \vee \psi, e\rangle &= \mathbf{check}\langle \varphi, e\rangle \ \mathtt{or}\ \mathbf{check}\langle \psi, e\rangle & & \mathbf{check}\langle \varphi\,;\,\psi, e\rangle = \mathbf{check}\langle \varphi, e\rangle\,.
\end{array}
$$

Fig. 5. Check function

Finding the first occurrence using our algorithm is very easy. We need to find $\textsc{FirstMatch}(\varphi,\sigma) = \xi^{-1}(\mathbf{extract}\langle\mathbf{remove}\langle\mathbf{match}\langle\ldots\mathbf{match}\langle\varphi, e_1\rangle\ldots, e_j\rangle\rangle\rangle)$, where $1 \leq j \leq n$ such that $\mathbf{value}\langle\mathbf{match}\langle\ldots\mathbf{match}\langle\varphi, e_1\rangle\ldots, e_{j-1}\rangle\rangle = \mathtt{false}$ and $\mathbf{value}\langle\mathbf{match}\langle\ldots\mathbf{match}\langle\varphi, e_1\rangle\ldots, e_j\rangle\rangle = \mathtt{true}$. This can be done straightforwardly by applying our rules.

However, in case of some attack scenarios, finding only the first occurrence could be insufficient and we would prefer to find all occurrences that have events appropriately close to each other. Therefore, we can restrict the problem to finding occurrences from either set k-$\textsc{Match}(\varphi,\sigma)$ or $\textsc{ShortestMatch}(\varphi,\sigma)$. To find the former set, we proceed similarly like in case of finding the set $\textsc{AllMatch}(\varphi,\sigma)$, with the exception that now, at each step i, we delete all formulas $\psi \in \Psi_i$ for which the corresponding (possibly partially) occurrence set $L_\psi \in \mathcal{L}(\sigma)$ has an insertion distance larger then k, i.e. $\Psi_i := \Psi_i \setminus \{\psi \in \Psi_i \colon id(L_\psi, \sigma[1,i]) > k\}$. In practice, this can be done very efficiently during the evaluation. At each step of evaluation we can also move formulas $\psi \in \Psi_i$ for which $\mathbf{value}\langle\psi\rangle = \mathtt{true}$ to set Ψ'. It means that we can stop evaluating formulas if they are satisfied. We can do this since we assume that formulas are monotonic.

To find the set $\textsc{ShortestMatch}(\varphi,\sigma)$ we proceed as follows: at each step i we delete all formulas $\psi \in \Psi_i$ for which there exists another formula ψ' such that their corresponding (possibly partial) occurrence sets $L_\psi, L_{\psi'} \in \mathcal{L}(\sigma)$ satisfied $L_{\psi'} \prec L_\psi$, i.e. $\Psi_i := \Psi_i \setminus \{\psi \in \Psi_i \colon (\exists \psi' \in \Psi)(L_{\psi'} \prec L_\psi)\}$.

6 Simulation Experiments

To verify our formal conceptual matching framework, we have implemented our algorithm *Sig*ITL*-Match in OCAML functional language and tested it experimentally. We evaluated its performance on a PC computer (Pentium M 1.5 GHz processor with 512 MB memory). For our simulations we used experimental values from [6], assuming that the size of Σ may vary from 60 to 80 and attack signatures consist of no more than 8 events. We generated a random event sequence σ over an alphabet of size $\Sigma = 80$ and measured runtimes of *Sig*ITL*-Match for searching a set of signatures from [13]. Our preliminary experimental results for various sizes of σ are given in Table 1. We have skipped results of tests for finding a set all occurrences, because even for small $|\sigma|$ this set was indeed very large. In cases of finding the first match and the shortest match, the algorithm was very fast even for very long event sequences. Similarly, finding occurrences from the set k-Match was also very quick. For example, our

Table 1. Runtimes of *Sig*ITL*-Match algorithm for finding various types of occurrences of signatures: $\varphi_1 = (\Diamond(\mathsf{A}\,;\,\mathsf{B})\wedge\Diamond\mathsf{C})\,;\,\mathsf{D}$, $\varphi_2 = \Diamond\mathsf{A}\,;\,(\mathsf{B}\wedge\Box\neg\mathsf{C})\,;\,\mathsf{D}$ and $\varphi_3 = ((\Diamond\mathsf{A}\,;\,(\mathsf{B}\wedge\Box\neg\mathsf{E}))\wedge\Diamond(\mathsf{D}\vee\mathsf{F}))\,;\,\mathsf{C}$. The t_φ stands for runtime (in seconds) and n_φ for $|k$-Match$(\varphi,\sigma)|$

$\|\sigma\|$	FirstMatch			ShortestMatch		
	t_{φ_1}	t_{φ_2}	t_{φ_3}	t_{φ_1}	t_{φ_2}	t_{φ_3}
1000	0.003	0.012	0.0007	0.004	0.008	0.012
10000	0.0035	0.015	0.0009	0.024	0.036	0.064
100000	0.0038	0.019	0.0012	0.232	0.296	0.564
1000000	0.0052	0.025	0.0019	2.34	2.97	4.63

$\|\sigma\|$	25-Match						50-Match					
	n_{φ_1}	t_{φ_1}	n_{φ_2}	t_{φ_2}	n_{φ_3}	t_{φ_3}	n_{φ_1}	t_{φ_1}	n_{φ_2}	t_{φ_2}	n_{φ_3}	t_{φ_3}
1000	1	0.008	4	0.012	3	0.02	14	0.032	6	0.024	19	0.12
10000	2	0.04	9	0.056	6	0.108	21	0.092	24	0.116	34	0.336
100000	24	0.328	61	0.46	35	0.912	150	0.744	180	1.124	206	2.648
1000000	241	0.396	539	4.76	336	9.29	1615	7.96	1844	10.66	2072	25.85

algorithm found all 2072 occurrences from the set 50-Match, in the sequence of size $|\sigma| = 10^6$ in the time less then 26 seconds, what gives approximately 38 thousands of events per second. At this point, we already have some more ideas how to better optimize our algorithm, but nevertheless at this stage, our experimental results look very promising.

7 Conclusion and Future Work

In this paper, we have presented the way of extending the existing methods of intrusion detection systems based on a temporal logic approach to precisely localize the multi-event attack signatures in an event sequence. Our research was mainly motivated by the fact that the Eagle logic and Interval Temporal Logic in particular are well established approaches in real IDSs, but their inability to locate accurately the signatures detected is a major limitation for practical runtime systems. To address this limitation, we have formally introduced the notion of an occurrence of a signature match in the *Sig*ITL logic, which allows us to develop a logical approach to localizing signatures in the event sequence. In our single formal matching framework we have included a few different types of signature matches existing in the intrusion detection literature.

The performed experiments show that our matching algorithm using basic simplification techniques of temporal logic formulas is very fast and efficient. However, the choice of a kind of matching set reported should always depend on IDS performance limits, the type of an attack and the consequent risk to the system or network that is under protection.

In our future work on logic based intrusion detection systems, we plan to introduce the localization mechanism for the Eagle-based Monid tool by extending our framework with parameters to enable reasoning about data values for more advanced event correlation. A more user-friendly interface will be provided and published, and the tool will be evaluated on real case studies.

Acknowledgments. We would like to thank Ben Moszkowski who introduced us to Interval Temporal Logic and whose scientific assistance was very helpful in our research. We are most grateful to Neil Mitchison for improving an early draft of this paper and for his invaluable support. We also thank the anynomous reviewers for their useful comments.

References

1. Barringer, H., Goldberg, K., Havelund, K., Sen, K.: Rule-based Runtime Verification. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, Springer, Heidelberg (2004)
2. Couture, M., Ktari, B., Mejri, M., Massicotte, F.: A Declarative Approach to Stateful Intrusion Detection and Network Monitoring. In: Proceedings of the 2nd Annual Conference on Privacy, Security and Trust (PST) (2004)
3. Kumar, S., Spafford, E.: A Pattern Matching Model for Misuse Intrusion Detection. In: Proceedings of the 17th National Computer Security Conference (1994)
4. Kumar, S., Spafford, E.: An Application of Pattern Matching in Intrusion Detection. Purdue Technical Report CSD-TR-94-013 (1994)
5. Kumar, S.: Classification and Detection of Computer Intrusions. PhD Thesis, Purdue University (1995)
6. Kuri, J., Navarro, G., Me, L.: Fast Multipattern Search Algorithms for Intrusion Detection. Fundamenta Informaticae, Special Issue on Computing Patterns in Strings 56(1/2) (2003)
7. Lin, J.L., Wang, X.S., Jajodia, S.: Abstraction-Based Misuse Detection: High-Level Specifications and Adaptable Strategies. In: Proceedings of the 11th Computer Security Foundation Workshop, IEEE Computer Society Press, Los Alamitos (1998)
8. Me, L.: A Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis. In: Proceedings of the 1st International workshop on the Recent Advances in Intrusion Detection (RAID) (1998)
9. Moszkowski, B.: Executing Temporal Logic Programs. Cambridge University Press, Cambridge, England (1986)
10. Moszkowski, B.: A Hierarchical Completeness Proof for Propositional Interval Temporal Logic with Finite Time. Special Issue of a Journal of Applied Non-Classical Logics on Interval Temporal Logics and Duration Calculi 14(1-2) (2004)
11. Moszkowski, B.: A Hierarchical Analysis of Propositional Temporal Logic based on Intervals. Journal of Logic and Computation (2006)
12. Naldurg, P., Sen, K., Thati, P.: A Temporal Logic Based Framework for Intrusion Detection. In: de Frutos-Escrig, D., Núñez, M. (eds.) FORTE 2004. LNCS, vol. 3235, Springer, Heidelberg (2004)
13. Nowicka, E., Zawada, M.: Modeling Temporal Properties of Multi-event Attack Signatures in Interval Temporal Logic. In: Proceedings of the IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006) Tuebingen, Germany (2006), available at `http://www.diadem-firewall.org/workshop06/papers/monam06-paper-37.pdf`

14. Olivain, J., Goubault-Larrecq, J.: The Orchids Intrusion Detection Tool. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, Springer, Heidelberg (2005)
15. Ouyang, M-G., Pan, F., Zhangy, Y.T.: ISITL - Intrusion Signatures in Augmented Interval Temporal Logic. In: Proceedings of the 2nd International Conference on Machine Learning and Cybernetics (2003)
16. Roger, M., Goubault-Larrecq, J.: Log Auditing Through Model Checking. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, Los Alamitos (2001)
17. Totel, E., Vivinis, B., Me, L.: A Language Driven Intrusion Detection System for Events and Alerts Correlation. In: Proceedings of the 19th IFIP International Information Security Conference, Kluwer Academic Publishers, Dordrecht (2004)
18. Uppuluri, P., Sekar, R.: Experiences with Specification Based Intrusion Detection System. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, Springer, Heidelberg (2001)

Towards Fully Automatic Defense Mechanism for a Computer Network Emulating Active Immune Response

V. Skormin, O. Shiryayeva, A. Tokhtabayev, and J. Moronski

Binghamton University, Binghamton NY, USA
vskormin@binghamton.edu

Abstract. Modern information attacks are perpetrated by the deployment of computer worms that propagate extremely fast leaving little or no time for human intervention. This paper presents the concept of a fully automatic computer network security system capable of timely detection and mitigation of information attacks perpetrated by self-replicating malicious software. The system will detect an attack and synthesize and deploy specialized self-replicating anti-worm software for attack mitigation with a capability to alter the network topology to quarantine infected portions of the network. Special technologies allowing for the observability and controllability of the overall process will be implemented thus facilitating the deployment of advanced control schemes to prevent an overload of the network bandwidth. Particular components of this system have been developed by the authors or suggested in literature thus suggesting its feasibility. The implementation aspects of the described system are addressed. The technology described herein emulates immune defenses honed to perfection by million-year evolution to assure the safety and dependability of future computer networks. It presents a new paradigm in computer network security.

Keywords: Computer network, computer worms, immune response, information attacks, automatic systems.

1 Introduction

Our ever-growing dependence on computer networks is accompanied by ever-growing concerns about the networks vulnerability to information attacks and the dependability of the existing network security systems. Major threats, well recognized by government, private institutions and individual users, are stemming primarily from self-replicating malicious software. Modern worms and viruses propagate through the Internet much faster and cause more damage than their predecessors. In 2001 the Code Red worm propagated faster than the Melissa virus in 1999 and much faster than Morris worm in 1988. In the case of the Code Red worm, only several days passed from the moment of its first detection to a wide spread propagation of malicious activity. Several months later, the Nimda worm caused severe damage within one hour of the detection of infection. The Slammer worm caused harm in only a few minutes. Since Code Red, the development of complex infection strategies has progressed to such a level that in January 2003, the W32.SQLExp worm propagated so fast that human intervention could not prevent its spread.

Effective counter-measures to worm attacks consist of revealing the infected hosts as quickly as possible in order to minimize damage and searching for the vulnerabilities in security systems. However, there are many factors that decrease the efficiency of these efforts. Every year about four thousand new "holes" in security systems are revealed. At present more than 200 million computers are connected to the Internet and their numbers are growing rapidly. Every moment of every day millions of vulnerable computers are interconnected through the Internet. Sophisticated attacks can provide resources to adversaries allowing them to utilize the vulnerable computers to aide in carrying out future wide spread attacks. Many attacks are performed in a completely automatic fashion and are deployed at the speed of light throughout the Internet without regard to geographical and national borders. Technologies utilized by malicious software are becoming more and more complex and in some cases are completely concealed from detection that ultimately has the effect of increasing the time necessary for the detection and analysis of an attack. The combination of these factors results in a situation where even with the fastest response involving human intervention a major cyber attack would have enough time to cause significant damage to the national infrastructure. This reality is well recognized by a number of researchers and justifies the necessity for the development of novel, fully automatic, decentralized computer network defenses emulating known immune mechanisms honed to perfection by multi-million year evolution.

A successful solution to this problem cannot be achieved without understanding the complex phenomena of propagation of self-replication entities in the network in conjunction with their effect of the network bandwidth. An AFOSR project [1] resulted in a mathematical model that provided not only a quantitative basis for the understanding and simulation of these phenomena, but also a basis for the development of a control mechanism for mitigation of viral activity by deployment of an anti-worm. While any deployment of self-replicating software could be highly detrimental to the network, it is necessary to develop a technology for making the process of co-replication of worm and anti-worm observable and controllable. It will utilize the previously developed Dynamic Code Analyzer [2] deployed in a number of specially designated machines distributed within the network for the detection and correlation of instances of self-replication behavior of software indicative of computer worm propagation. The detection would trigger synthesis on demand of a specialized anti-worm that could be released in the network. A statistical selective sampling procedure is proposed for the estimation of the number of hosts in the network affected by the worm and anti-worm by periodic scanning a relatively small group of randomly chosen hosts. This approach, minimizing the impact on the network bandwidth, is suggested as the means for the generation of a feedback loop that is crucial for the implementation of any control mechanism. A technology facilitating a controlled time-varying rate of self-replication of the anti-worm is proposed. Finally, an advanced control law relating the propagation rate of the anti-worm to the feedback information will be established in full compliance with modern control theory. The resultant technology is being implemented in an experimental computer network testbed built at Binghamton University under the AFOSR DURIP funding as a fully operational prototype of a fully automatic defense mechanism for a computer network emulating an active immune response.

2 Existing Research

In 1991 Kephart pioneered the application of epidemiological models for the mathematical description of the complex dynamic phenomenon of propagation of self-replicating software such as simple file viruses [3]. File viruses distribution in networks was formalized in terms of probability laws [4, 5] for homogeneous, localized and random replication patterns. He should be credited for the introduction of the very concept of immune system for computers in [6] and its further development in [7] and [8], [9]. Worms have received true recognition after the attack of the Code Red worm in July, 2001. Consequently, the first propagation case study was presented in [10], where authors utilized the collected data for the analysis of the infection and disinfection rates. More fundamental analysis of the worm propagation dynamics was performed for SQL Slammer worm in [11].

Applications of various epidemiological models for modeling and analysis of real and theoretical worms with respect to different network topologies and scanning techniques could be found in [12], [13], [14], [15], [16], [17], [18], [19]. In [15] and [18] special studies were conducted to investigate several key characteristics of infection, including the rate of infection through the network, the rate at which individual nodes are re-infected during an attack, and the effect of immunization of certain nodes in the network. The propagation intensity of theoretical worms was studied for different scanning techniques in [16]. Zou et al. [17] introduced a two-factor worm model that considered the effect of human countermeasures as well as the effect of the network congestion caused by extensive worm scanning activities. Authors of [20] presented a dynamic quarantine system for infected processes based on two principles, the preemptive quarantine and feedback adjustment.

The concept of an active defense mechanism for a computer network attacked by a worm was presented by Liljenstam and Nicol in [21], [22], where authors discussed different models of active defenses and their effect on the network throughput. The active immune-like network response was called reactive antibody defense in [23]. The consideration of active defense mechanisms has immediately prompted the consideration of the effect of limited network resources, such as bandwidth. It was pointed out that the deployment of a so-called anti-worm may significantly consume network bandwidth as well as in the case of a malicious worm [24]. In [24] authors emphasize the importance of applying optimal reactive response that takes into account the infection and treatment costs in terms of network resources. Authors discuss possible ways to determine optimum level of the defense efforts to be applied for a given rate of infection spread that would minimize some total cost function.

Recently, a new generation of defense mechanisms for computer networks, minimizing the need for human intervention, became increasingly popular. Authors in [25] presented a technique for automatic generation of an anti-worm through detecting and substituting payload of the malicious worm. As a result, they proposed a method that has a potential for transforming a malicious worm into an opposing anti-worm. A system for automatic revealing susceptible points and generating a patch for target application is presented in [26]. A feasibility of automatic signature generation for worms perpetrating buffer overflow attacks has been studied in [27], [28].

In summary, it could be noted that it is recently recognized that only a fully automatic defense mechanism can protect computer networks from modern, fast propagating worms. As with any automatic process, this defense mechanism cannot exist without the implementation of a stable negative feedback control scheme. Technical literature, however, does not offer any examples of such schemes.

In 2001-02 the authors were engaged in the project BASIS (Biological Approach to System Information Security) funded by Air Force Research Laboratory. This effort was aimed at the establishment of important similarities between a biological immune system and a computer network subjected to an information attack that could be explored for the development of the next generation of computer network defenses. This project resulted in a number of publications [29], [30] and provided the team with valuable new concepts in computer security.

Project Recognition of Computer Viruses by Detecting their Gene of Self-Replication, also funded by the air Force Office of Scientific Research (AFOSR), has been exploring the notion that while most malicious software self-replicates in order to create a computer epidemic maximizing its destructive effect, self-replication of legitimate software is very uncommon. At the same time, the number of practical self-replication techniques utilized in viruses and worms is quite limited and requires the developers of new attacks to utilize the same old self-replication techniques in new viruses and worms. Consequently, the detection of self-replication functionality in computer code provides the basis for the detection of both known, and what is more important, previously unknown malicious software. It was found that monitoring and analysis of system calls during the execution time provides the most dependable approach for the detection of attempted self-replication [31], [32]. This has resulted in the development of a Dynamic Code Analyzer (DCA) [31], a resident software tool that monitors system calls and detects specific subsequences (patterns in the system call domain) indicative of self-replication. The process engaged in self-replication would be suspended and the user is given an authority to continue or terminate the process. The DCA has been successfully tested against both known and previously unknown malicious software. It could be seen that while DCA may not prevent the damage caused to an individual host, it surely prevents the development of computer epidemics.

The AFOSR project Emulation of Active Immune Response in Computer Networks allowed the authors to investigate the effect of self-replicating anti-viruses or anti-worms on a computer network subjected to an information attack. The obtained mathematical model of the networks Active Immune Response (AIR), consistent with the main concepts of biological immunology, will be presented below. . It effectively represents the major properties of a computer network describing the complex interplay of the factors responsible for the propagation of self-replicating software in the network, both worms and anti-worms, and reflects typical strategies of the information attacks perpetrated by computer worms.

It is expected that a self-replicating anti-worm would implement the most advanced propagation strategies resulting in disinfection and/or immunization of individual hosts. Consequently, both the worm and anti-worm activity has to be quantified by the number of infected hosts and the number of disinfected or immunized hosts correspondingly. Worm activity has a strong impact on the bandwidth of networks. The bandwidth of a network can be considered as the most relevant network resource that affects both the quality of network operation as well as the propagation of self-replication software. Consequently, the bandwidth of a network becomes a key factor to be addressed in a mathematical model of the networks immune response. While there are many alternative ways to quantify the bandwidth of a network, it typically represents the amount of transmitted information per unit of time.

The resultant AIR model follows the principles of operation of a biologic immune system describing the interaction between resources of the organism, antigens, and the immune system. The model variables represent (1) network resources, expressed as the available capacity (bandwidth) of the information channels (bits/sec) utilized by the worm and anti-worm software, (2) number of disinfected/ immunized hosts, and (3) number of infected hosts. It could be seen that the above variables are interrelated: differential increments of infected and disinfected hosts directly affect the network bandwidth, propagation rates of the worm and anti-worm depend on the available network bandwidth, the number of infected computers depends on the propagation rates of worm and anti-worm, etc. The developed model comprises several nonlinear stochastic differential equations; the stochastic nature of the AIR model is reflected by the fact that the numbers of infected and disinfected/immunized hosts are statistical estimates of the respective quantities that in reality could be obtained by the scanning of a relatively small group of randomly chosen hosts (selective sampling).

The mathematical model provides the basis for the implementation of various control schemes facilitating the deployment of an observable and controllable anti-worm within the limited bandwidth of the network thus achieving sustainable operation of a network subjected to an information attack.

3 Mathematical Model of the Immune-Type Response of the Network

Mathematical modeling the immune response necessitates analyzing the factors responsible for the propagation of viruses and revealing the most significant of them in order to define strategies which have a high probability of being used by the creators of worms in the future.

In order to propagate, computer worms scan a wide-area network, striving to find vulnerabilities in software of individual hosts. Having found the IP-address of a vulnerable host, the worm simply injects itself from the network into the unprotected machine using previously revealed vulnerabilities. Once installed in the hosts memory, the worm, searches for other vulnerable computers and sends itself to their IP addresses. While different types of worms use different propagation strategies and exhibit different propagation rates, all of them successfully propagate, using the slightest vulnerability to penetrate existing security systems. It could be seen that the number of infected (or scanned) hosts is one of the variables quantifying the attack.

Worm activity has a negative impact on the bandwidth of networks. For example, the code size of the quickly propagating Slammer is very small, 404 bytes including the header. The small size accounts for the high speed of its propagation. During the first minute of attack, the quantity of infected computers grew exponentially, doubling every 8.5 seconds. By searching for potential victims during a 10-minute time period, the worm scanned about 3.6 billion out of approximately 4 billion existing Internet addresses, reaching the scanning rate of 55 million hosts per second during the first three minutes of the attack. As the attack progressed, this rate decreased due to the limited networks bandwidth. Although the worm did not carry destructive instructions, as did Code Red and Nimda which changed files and damaged web-sites, it caused serious damage during the peak of the epidemic consuming a significant portion of the Internet bandwidth, and interfering with the operation of many servers. Consequently, the bandwidth constitutes the main network resource relevant to our problem, and properly quantified presents another important characteristic of the attack.

A mathematical model of the network response cannot be established without describing the effects of anti-worm activity, i.e. the worm-like propagation of anti-worm programs. Such anti-worm technology could be exemplified by Welchia, which is one of a few worms intended for the neutralization of another malicious program, Blaster worm. In the same way as Blaster, Welchia penetrates into a computer through a gap in Windows firewall, first having verified that the computer is infected with Blaster. Welchia then deletes Blaster, completely restores the attacked system, and loads the Microsoft update thereby fixing the vulnerability. Understandably, the propagation of anti-worms is affected by the limited networks bandwidth and can be described by the number of disinfected (or scanned) hosts or released anti-worm software units.

Now it could be seen that the mathematical model of the networks immune-type response should describe the complex, dynamic interaction of three factors, the number of infected hosts (or scanned by the worm), the number of disinfected hosts (scanned by the anti-worm, or released anti-worm software units) and their combined effect on the network bandwidth. Following the principles of operation of the immune system describing the interaction between resources of the organism, antigens, and the immune system, we bring into consideration the basic variables of the mathematical model establishing the correspondence between the biological and computer network concepts.

Number of infected hosts: Let variable $X(t)$ represent the number of infected computers that would exponentially increase due to the worm propagation and decrease due to the anti-worm activity. The equation describing the dynamics of the number of infected computers at the absence of anti-worm and with infinite bandwidth could be described in the following way:

$$X(i) = \frac{n_w (N - X(i-1)) X(i-1)}{N} + X(i-1), \quad X(0) = X_0 \tag{1}$$

where n_w is the rate of sending attack packets by every infected host, X_0 initial population (number of attack sources), N is total number of susceptible hosts in the network. The first multiplicative term in (1) represents approximate probability of hitting vulnerable host.

Number of deployed units of anti-worm software: Let $Y(t)$ represent the number of deployed units of anti-worm software (number of hosts containing anti-worm). It is expected that the anti-worm population size, governed by some control mechanism, will exponentially increase to oppose the propagating worm and decrease as the worm being defeated. The dynamics of the appropriate process at the early stage of the attack can be defined as:

$$Y(i) = k_i Y(i-1) \beta_i, \quad Y(0) = Y_0 \tag{2}$$

where $k_i = X(i-1)\mu$ is the rate of sending attack packets by every unit of anti-worm at time i, $\beta_i = (N - Y(i-1))/N$ is an approximate probability of hitting a vulnerable host or a host infected by the worm at time i (a victim for the anti-worm), Y_0 is initial anti-worm deployed population. It could be seen that the rate of sending attack packets by anti-worm depends on worm population size what reflects the control mechanism.

System resources: The bandwidth represents the joint capacity of the networks communication channels and could be expressed in bit/sec. During the active network response the bandwidth is utilized by the worm as well as anti-worm. Introduce variable $W(t)$ that denotes the amount of consumed bandwidth by the worm and anti-worm during the attack. Note that before or after attack this variable exhibits all properties of a characteristic of a stable, inertial system and exponentially returns to the value of zero with time (network inertia accounts for the finite connectivity within the network, its distributed nature, and various phenomena slowing its operation). The equation describing the dynamics of the consumed network bandwidth could be written as:

$$W(i) = -W(i-1) \cdot \psi + X(i-1) \cdot n_w \cdot \theta + k_i \cdot Y(i-1) \cdot \theta, \quad W(0) = 0 \tag{3}$$

where $\psi > 0$ represents the natural rate of change of bandwidth, θ - an amount of consumed bandwidth due to transmitting of one attack packet of the worm or anti-worm. As it could be seen that the first term represents natural motion, the second and third terms reflect current bandwidth consumed by the worm and anti-worm respectively. To achieve realistic scenario of the worm and anti-worm propagation one should set ψ to be much smaller than the amount contributed by the worms.

Now let us modify equation (2) accounting for the effect of network bandwidth on the propagation of the anti-worm and assuming that the anti-worm generation effort is proportional to the number of infected computers (simple proportional control law). Then the number of instances of the anti-worm (hosts infected by anti-worm) will be changing according to the equation:

$$Y(i) = k_i \cdot Y(i-1) \cdot \delta_i \cdot \beta_i \; - \eta \cdot Y(i-1) + Y(i-1) \tag{4}$$

where $\delta_i = (1 - W(i-1)/W_{\max})$ is packet delivery probability, η is the rate of the anti-worm population decrease in natural motion. The natural anti-worm population decrease was introduced in order to achieve response decay in the absence of the intruder. The packet delivery probability decreases if the consumed bandwidth increases up to the maximum level ($W_{\max}$) which results in total network congestion. This term decreases the anti-worm propagation rate due to the bandwidth shortage.

Reconsider equation (1) taking into account that the bandwidth consumption slows down the worm propagation process and the anti-worm activity can reverse it due to disinfecting and patching. This process could be described as follows:

$$X(i) = \alpha_i \cdot n_w \cdot X(i-1) \cdot \delta_i - k_i \cdot Y(i-1) \cdot \delta_i \cdot \gamma_i + X(i-1) \tag{5}$$

where $\alpha_i = (N - X(i-1) - Y(i-1))/N$ is approximate probability of hitting a vulnerable host at time i by one instance of the worm(nor infected by the worm, nor the anti-worm), $\gamma_i = X(i-1)/N$ is probability of hitting a host infected by the worm. The first term in (5) represents potential increase of the number of infected hosts due to worm replication during one time tick, the second term represents number of hosts disinfected and recovered by the anti-worm during one time tick.

Hence, the mathematical model of the immune-type response of a computer network is described by the following system of discrete-time nonlinear equations, describing the inertial properties of the network and complex interaction between its key variables, the consumed network bandwidth, the number of infected hosts, and the number of deployed units of the anti-worm software:

$$\begin{cases} X(i) = \alpha_i \cdot n_w \cdot X(i-1) \cdot \delta_i - k_i \cdot Y(i-1) \cdot \delta_i \cdot \gamma_i + X(i-1) \\ Y(i) = Y(i-1) + k_i \cdot Y(i-1) \cdot \delta_i \cdot \beta_i - \eta \cdot Y(i-1) \\ W(i) = -W(i-1) \cdot \psi + X(i-1) \cdot n_w \cdot \theta + k_i \cdot Y(i-1) \cdot \theta \end{cases} \tag{6}$$

with the initial conditions

$$W(0) = 0, \;\; X(0) = X_0 > 0, \;\; Y(0) = Y_0 > 0 \tag{7}$$

Table 1 summarizes the parameters of the model (6).

Table 1. Model parameters

Parameter	Formula	Description
α_i	$\alpha_i = (N - X(i-1) - Y(i-1))/N$	Probability of hitting a vulnerable (nor infected by the worm, nor the anti-worm) host at time i by one instance of the worm
δ_i	$\delta_i = (1 - W(i-1)/W_{\max})$	Packet delivery probability
k_i	$k_i = X(i-1) \cdot \mu$	Rate of sending attack packets by every unit of anti-worm at time i
β_i	$\beta_i = (N - Y(i-1))/N$	Probability of hitting a vulnerable host or a host infected by the worm at time i (a victim for the anti-worm)
γ_i	$\gamma_i = X(i-1)/N$	Probability of hitting a host infected by the worm
μ	static	Proportional coefficient used in control law of the anti-worm generation rate
n_w	static	Rate of sending attack packets by every unit of the worm
θ	static	Amount of consumed bandwidth due to transmitting of one attack packet of a worm or an anti-worm
ψ	static	Coefficient determining the rate of decreasing of the amount of consumed bandwidth in natural motion.
N	static	Total number of susceptible hosts in the network (before worm and anti-worm activity)
$W_{\max}$	static	Maximum available bandwidth
η	static	Rate of the anti-worm population decrease in natural motion

In the process of designing of the model (6) several reasonable assumptions were made and some simplifications were introduced. Firstly, probabilities α_i, β_i and γ_i were approximated assuming distribution identity and independence, as well as neglecting network topology and infrastructure limitations. Secondly, for the sake of simplicity, we assumed that the worm and anti-worm have only one packet attack and utilize a QOS free transport protocol such as UDP. As an example of such a worm in real life, on can refer to the well known SQL.Slammer worm. Moreover, we assumed that network natural activity is much smaller than the worm and anti-worm combined activity. These simplifications allowed us to

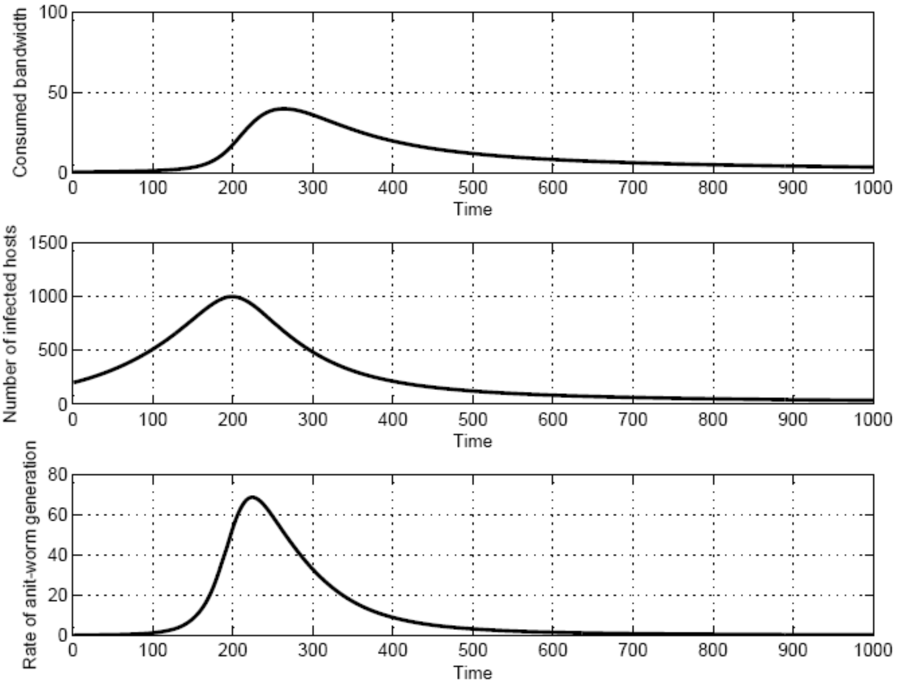


Fig. 1. Worm and Anti-worm dynamics

use the linear law for packet delivery probability. While the linear formula may not capture network congestion phenomena in all phases, it is still a good choice considering the network bandwidth as a single value.

Analysis of the above equations indicates that they represent the natural motion of a nonlinear system that in the case of global asymptotic system stability and regardless of the initial severity of the attack, signified by the value of X_0, results in $W(\infty) = 0,\ X(\infty) = 0,\ Y(\infty) = 0$, i.e. full recovery of the network. It is important to realize that equations (6) are stochastic by their very nature: their parameters reflect the combined behavior of numerous hosts of the network and therefore the obtained model can reliably describe the network in general but not any individual host. Moreover, numerical values of the model parameters reflect properties of a particular network and are subject to change depending on many factors including the propagation engine and payload of the particular worm.

The outcome of the simulation experiment is presented in the Figure 1 and static parameter values are listed in the Table 2. The upper plot in figure 1 demonstrates consumed bandwidth as a percentage of the maximum (total congestion) level ($W_{\max}$). The middle plot shows the number of infected hosts. Finally, the lower graph demonstrates the rate of newly generated instances of anti-worm, in other words number of infected hosts by anti-worm during the current time tick. The simulation results indicate that with the appropriate choice of the model parameters, it realistically describes the nature of the active network response: the increase of the number of infected hosts from some initially introduced value, increase of the rate of deployed anti-worm software units in response to the attack, drop of the bandwidth resource of the network (or increase of the consumed bandwidth) due to the worm and anti-worm propagation, and finally, the full recovery of the network from the attack.

Table 2. Model parameters

μ	0.00005
n_w	0.01
θ	0.5
ψ	0.02
N	10000
$W_{\max}$	240
$X(0)$	200
$Y(0)$	10
η	0.00001

The value of the obtained mathematical model is in its demonstrated ability to describe the interaction between the major variables capturing the behavior of a network equipped with anti-worm defenses under the attack. Consequently, the model provides a basis for the development and validation of advanced feedback control mechanisms of the network defenses.

4 Principle of System Operation and Major System Components

The principle of operation of an automatic defense mechanism capable of controlled deployment of a self-replicating anti-worm in response to an information attack on a computer network by self-replicating software is shown in Figure 2. This figure depicts malicious software deployed from an attackers computer as it propagates within the network. Eventually, it is detected by one of the specialized attack detection/identification stations dispersed within the network that automatically extracts a binary signature of the detected malicious software and transmits it to the Control station. Then a specialized anti-worm (self-replicating patch) is deployed from the control station and also propagates through the network disinfecting and immunizing individual hosts. Based upon the topology of infected/uninfected hosts the Control station will ether send self-replicating anti-worm software to prevent the propagation of the worm or logically disconnect hosts from the network.

The propagation rate of the anti-worm is time-dependent and is defined by a specially designed control law. The anti-worm is targeting the malicious software

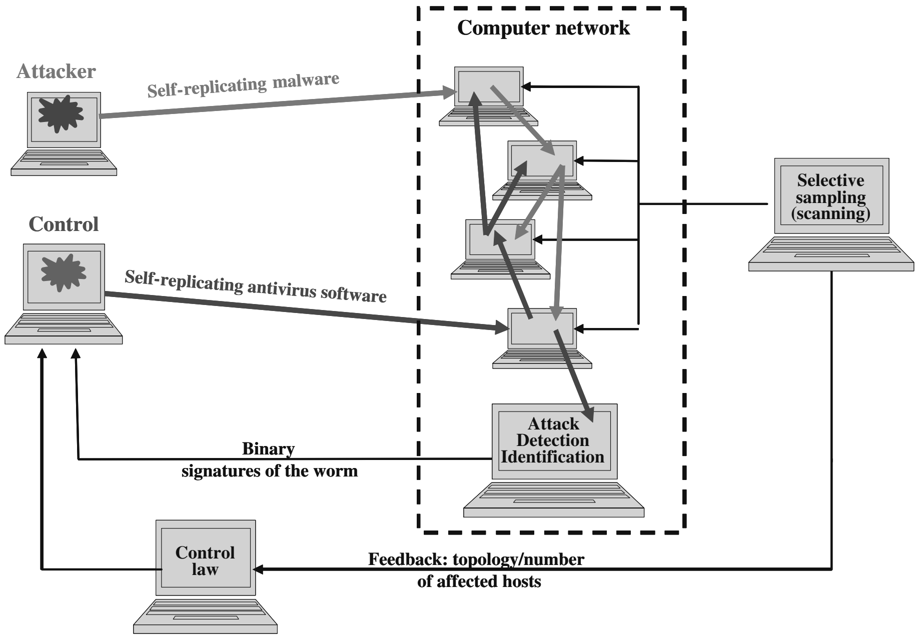

Computer network
Attacker
Self-replicating malware
Selective sampling (scanning)
Control
Self-replicating antivirus software
Attack Detection Identification
Binary signatures of the worm
Control law
Feedback: topology/number of affected hosts

Fig. 2. Principle of operation of automatic security system for a computer network

specified by a binary signature loaded in its targeting mechanism. The network computers (routers) are subjected to periodic selective scanning in order to estimate the number of infected and disinfected (immunized) hosts. The control law converts the feedback information on the overall status of a computer network subjected to both the information attack and the effects of the self-replicating anti-worm into numerical values of some parameters governing the propagation rate of the anti-worm. This control law is synthesized on the basis of nonlinear differential equations describing complex interplay between the number of infected hosts, number of operating anti-worm units, and the available resources of the network the remaining capacity of its communication channels.

4.1 Attack Detection/Identification

Attack detection/identification stations are interconnected host-based worm detection systems dispersed within the computer network. Such a station is visualized as a cluster of independent virtual computers, vulnerable to attacks, being run on the same physical host. In a way, the virtual computers of the detection stations form a network within the network and exchange security related information that is important for reliable detection of the attack and identification of the attacker. Each virtual computer of the station is equipped with the Dynamic Code Analyzer (DCA) capable of detecting attempted self-replication [31]. When attacked by a computer worm or virus, the virtual computer of the detection station executes the instructions of the malicious software including those resulting in its self-replication. It is commonly known that most malicious software self-replicates in order to maximize the severity of the attack, and this functionality typically invokes specific sequences (patterns) of system calls. The DCA monitors system calls and is capable of detecting the specific patterns of system calls indicative of self-replication. Upon detection of the attempted self-replication, the DCA suspends the process in question, generates a specific warning identifying the detected specific self-replication pattern and transmits it to all virtual machines of the detection station. While the system calls-based IDS are known to have a high rate of false positives, no further action is taken until the same alarm is generated by another virtual machine, thereby drastically reducing the probability of false alarms. When the alarm is substantiated by a warning from an additional source, a special procedure deployed at one of the detection stations, capable of extraction of the binary signature of the troublesome software is activated. The feasibility and general algorithms of such procedures are discussed by several authors [25], [26], [27], [28]. The extracted binary signature will serve as an ID of the worm/virus attacking the network and it will be transmitted to the control station.

4.2 Generation of the Feedback Signal

A successfully implemented feedback signal is crucial for any self-regulating system. In our situation the feedback signal represents the number of infected and disinfected hosts that varies as the attack and countermeasures progress. While scanning of the individual hosts of the entire network in a worm-like fashion is clearly unacceptable, a selective sampling (scanning) approach commonly utilized in quality control is adopted, and then the resultant problem is solved using statistical inference.

For a given population size and fixed sample size this problem is usually solved under some assumptions about the underlying statistical distribution of the sample that is expected to be geometric. In reality, one does not know the population size, i.e. the total number of hosts, since at any time computers may be arbitrarily connected and disconnected from the network. As a result, the geometric distribution assumption is not truly applicable. However, if the number of susceptible hosts is sufficiently large, we do not have to know the exact number of vulnerable computers to apply the binominal distribution. Presumably, the share of infected computers in a fixed size sample confirms to binominal distribution with the mean value equal to the share of infected computers in the whole population. In order to use statistical inference techniques to estimate the mean value one has to ensure that the sample is identically and independently distributed. In other words, the source must be stationary during the sample selecting process that could be assured by two quite realistic assumptions, (1) hosts whose status is polled are chosen uniform randomly, and (2) during the scanning session the number of vulnerable hosts and number of infected machines does not change.

The independence requirement can be achieved using a random number generator. The identical property can be satisfied only if computers would be scanned pseudo simultaneously, i.e. by a very short duration scanning session.

Under the assumption of binominal distribution, the percentage of infected computers is estimated as a proportion parameter. The required statistical significance of the estimate could be assured by the choice of the sample size. While a closed-form solution for the exact confidence interval by the acceptance region inversion does not exist, approximate solutions for a confidence interval based on Central-Limit Theorem could be considered. Then a Clopper-Pearson (exact) confidence interval for the estimate that provides an assigned coverage probability with any specified precision can be found via standard numerical methods [32]. Its computation results in an iterative process leading in the definition of the minimum sample size providing a specified relative error (statistical significance) of the estimate. Ultimately, this provides a theoretical foundation for the utilization of selective scanning as the means of establishing a dependable feedback representing the dynamics of the worm/anti-worm interaction process in the network without a significant impact on the network bandwidth.

4.3 The Control Station

A control station is a computer equipped with software for generation and controlled dissemination on demand of a specialized anti-worm (anti-virus). An anti-worm is an already existing software entity that consists of a propagation engine, a targeting mechanism and a payload, see Figure 3. The propagation engine implements the most efficient propagation techniques of computer worms that imply scanning of the network in the search of susceptible hosts and transmitting

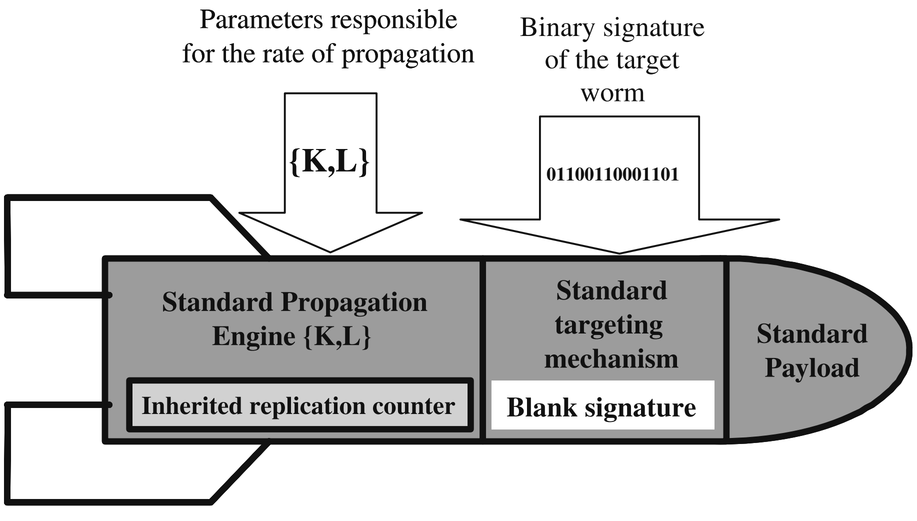

Parameters responsible for the rate of propagation
Binary signature of the target worm
{K,L}
01100110001101
Standard Propagation Engine {K,L}
Inherited replication counter
Standard targeting mechanism
Blank signature
Standard Payload

Fig. 3. Composition of an anti-worm

the appropriate code to such hosts [18], [20]. However, unlike the common worm intended to maximize the effect on the network, the propagation rate of the anti-worm will be controlled according to a control law that takes into account the number of infected hosts and the availability of the network bandwidth. This will be achieved by an inherited replication counter that controls the allowable number of generations, K, and the allowable number of offsprings in each generation, L, as the anti-worm propagates. The K and L numbers are time-dependent and are defined by the control law and communicated to the control station. It could be seen that having such a finite and controllable K, L combination results in the limited life span of the anti-worm so that it will cease to exist as the attacking worm is defeated, and due to the properly chosen control law does not congest the network.

The standard payload of the worm implements functions resulting in the neutralization of a worm if the host is infected and/or immunization of the host by making it immune to the worm. Both functions could only be performed in response to a particular worm that is uniquely identifiable by its binary signature and detected by the standard targeting mechanism operating in the traditional anti-virus software fashion. This could be achieved by loading the binary signature of the propagating worm extracted by one of the detection stations and transmitted to the control station.

The anti-worm is designed to propagate in the same environment utilizing some of the same vulnerabilities exploited by worms. This strategy is applicable to anti-worm propagation in vulnerable systems, as well as systems that are already infected with the worm. Certain worm propagation scenarios may include a self-patching worm behavior that would prevent an anti-worm from replicating onto already infected machines. This situation may be resolved by implementing such methods as establishing a dedicated communication channel for anti-worm propagation, or disabling self-patching at the system level by utilizing a run-time modification prevention approach. While such counter-measures are less favorable in the global net environment due to local system modification requirements, they would further enhance the functionality of the proposed active immune response system.

In an effort to speed the recovery of an infected network, the various control stations will also have control of administratively close routers allowing for quarantine and isolation of the infected portions of the network. If the infected portions of the network can be isolated quickly, not only will further spread of the worm be reduced, but the transmission of the anti-worm will be more successful as networking resources will be worm free and available. In addition, knowledge of the infection concentration will aid in guiding the anti-worm propagation.

Several ways of disconnecting or isolating hosts by the control station will be investigated. Possibilities include disconnecting the host from the network by disabling its various network interfaces, altering network address resolution tables, or by dynamically changing host routing information and reconfiguration of network routers and switches. Some of these techniques are rather simple to implement but may not offer much flexibility in the recovery period while others can be quite complex, involving networking hardware from several vendors with very different configuration methods.

4.4 The Control Law

The control law is responsible for the stability and efficient operation of the described system. It will establish a relationship between the control parameters of the anti-worm K, L, be responsible for the rate of its propagation, and provide feedback information representing the severity of the attack, namely, the estimated number of infected hosts, disinfected hosts and the available network bandwidth. The control law is formulated on the basis of the mathematical model of the active network response developed under the existing AFOSR project [1]. After the expansion, linearization of equations (6), this model could be written in a traditional discrete-time state-variable notation as

$$\begin{bmatrix} w[n+1] \\ x[n+1] \\ y[n+1] \end{bmatrix} = A \begin{bmatrix} w[n] \\ x[n] \\ y[n] \end{bmatrix} + B \begin{bmatrix} K[n] \\ L[n] \end{bmatrix}, \quad \begin{bmatrix} K[n] \\ L[n] \end{bmatrix} = -F \begin{bmatrix} w[n] \\ x[n] \\ y[n] \end{bmatrix}$$

where w[n], x[n] and y[n[are state variables of the active network response that correspondingly represent the consumed network bandwidth, number of infected hosts, and number of active (i.e. engaged in scanning activities) anti-worm units at particular moments of discrete time, n=1,2,

K[n] and L[n] are the control efforts of the active network response representing the allowable number of generations and the allowable number of offsprings in each generation of the anti-worm,

A is a matrix representing linearized complex interrelationships between the number of infected hosts, number of active anti-worm units and the network bandwidth,

B is a matrix representing linearized effects of the control parameters of the anti-worm on the number of infected hosts, number of active anti-worm units and the network bandwidth,

F =F[n] is a matrix of the state-variable controller,

A combination of the matrix-vector equations written above results in the discrete-time description of the closed-loop system that effectively is the equation of natural motion of an inertial system

$$\begin{bmatrix} w[n] \\ x[n] \\ y[n] \end{bmatrix} = (A - BF) \begin{bmatrix} w[n] \\ x[n] \\ y[n] \end{bmatrix}$$

It is known that the steady-state regime of a dynamic system described by such an equation is dependent on the eigenvalues of matrix $A^{CL} = A - BF$ and such a system is stable, i.e. all eigenvalues of matrix A^{CL} are located in the left-hand half of the complex plane, is

$$\begin{bmatrix} w(\infty) \\ x(\infty) \\ y(\infty) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

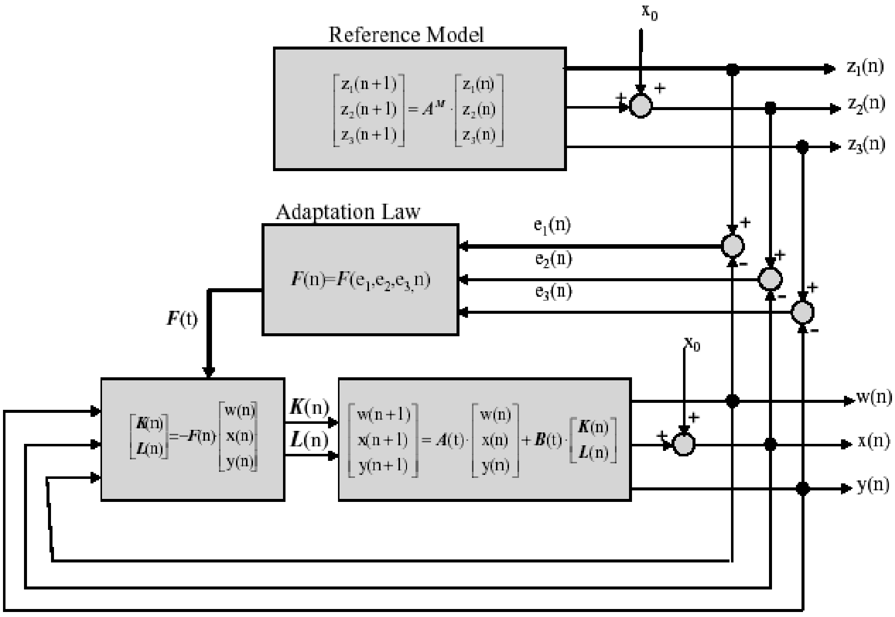

Reference Model
$\begin{bmatrix} z_1(n+1) \\ z_2(n+1) \\ z_3(n+1) \end{bmatrix} = A^M \cdot \begin{bmatrix} z_1(n) \\ z_2(n) \\ z_3(n) \end{bmatrix}$
x0
z1(n)
z2(n)
z3(n)
Adaptation Law
F(n)=F(e1,e2,e3,n)
e1(n)
e2(n)
e3(n)
F(t)
$\begin{bmatrix} K(n) \\ L(n) \end{bmatrix} = -F(n) \cdot \begin{bmatrix} w(n) \\ x(n) \\ y(n) \end{bmatrix}$
K(n)
L(n)
$\begin{bmatrix} w(n+1) \\ x(n+1) \\ y(n+1) \end{bmatrix} = A(t) \cdot \begin{bmatrix} w(n) \\ x(n) \\ y(n) \end{bmatrix} + B(t) \cdot \begin{bmatrix} K(n) \\ L(n) \end{bmatrix}$
w(n)
x(n)
y(n)

Fig. 4. Adaptive control of the active network response

Therefore, it could be seen that for arbitrary matrix A and matrix B appropriate selection of the controller matrix F would assure that the network engaged in active response returns to its status preceding the attack. Moreover, the selection of specific eigenvalues of matrix A^{CL} utilizing eigenvalue assignment techniques common in modern control theory enables the designer to manipulate the duration of the entire attack mitigation process.

The authors are aware of the fact that the above situation is overly optimistic due to the fact that matrix A and matrix B are poorly known and their values are time-dependent due to changing properties of the network and moving point of linearization. It is proposed to estimate these matrices experimentally by periodic deployment of a specially designed short-lived worm. The poor knowledge of matrices A and B and their time-dependence could be addressed by the application of a model-reference (adaptive) control law [34] that results in a time-dependent controller matrix $F = F[n]$. The resultant control system configuration is depicted in Figure 4.

Figure 4 shows a reference model (a simulation module) representing the required closed-loop dynamics of the active response assured by the selection of its fundamental matrix A^M,

$$\begin{bmatrix} z_1(n+1) \\ z_2(n+1) \\ z_3(n+1) \end{bmatrix} = A^M \cdot \begin{bmatrix} z_1(n) \\ z_2(n) \\ z_3(n) \end{bmatrix}$$

It could be seen that the state variable of the active response, $x(n)$, representing the number of infected hosts and the corresponding state variable of the reference model, $z_1(n)$, have the same initial conditions, $x(0) = z_1(0) = x_0$ equal to the initial number of infected hosts estimated at the detection of the attack. The error vector, $[e_1(n), e_2(0), e_3(0)]^T$, is calculated as a discrepancy between corresponding states of the reference model and active response and provides the input information for the adaptation block implementing the adaptation law. This law implements one of several techniques described in [34] resulting in perfect asymptotic adaptation that implies that all three states of the active immune response, the number of infected hosts, the number of active anti-worm units, and the consumed network bandwidth will converge to zero regardless of initial conditions, i.e. x_0.

The control law is currently being researched under the AFOSR project [33]. There are several approaches leading to the definition of the control law. The hyperstability and positivity approach [34], developed in model reference adaptive control is based on the linearization of the mathematical model of the network response, design of a state-variable controller with adjustable parameters, and the utilization of adaptation laws to assure the conformance of the parameters of the controller to the immediate operational regime of the inherently nonlinear, time-varying system. The method based on Liapunov stability conditions deals directly with the nonlinear mathematical model of the controlled process and also leads to the synthesis of a model-reference adaptive controller [34]. The model reference controllers have been successfully designed and verified by the application to the nonlinear mathematical model of the network response implemented in MATLAB.

5 System Implementation

The described system is being implemented in a computer network testbed at Binghamton University equipped with powerful workstations and servers capable of emulating up to 1000 interconnected, highly reconfigurable, independent hosts, see Figure 5. Should less computationally intensive security tasks be investigated, the number of virtual hosts could be further increased allowing for emulation of an even larger network environment.

The testbed provides important data for computer network security research including analysis of malicious software, vulnerability analysis of network components, efficiency and dependability of various security mechanisms, and network administration under attacks, thus facilitating the development of advanced network defenses as well as technology that could be used in information warfare. It provides a controlled physical environment in which a virtual network could be established allowing for the realization and close monitoring of every phenomenon observed in computer networks subjected to information attacks, including:

- Scalable heterogeneous computer network facilities. The controlled physical environment provides a safe, realistic and flexible foundation on which virtual networks can be constructed. To provide this capability, the facility includes computers having different architectures (PCs, Suns, and Macs) running a variety of operating systems in diverse configurations and interconnected by both a high-speed wired network as well as a wireless network.

- Means of inter-computer communication and secure network administration. The physical network must not only support a real-time virtual networked environment but also provide security and safety, containing attacks within the virtual network. Dedicated servers are used for network support and security analysis administration. High-speed network switches, routers and hubs as well as dedicated isolated wireless networking hardware provides the physical backbone for the network. Network management and monitoring software assure safety and ease of administration.
- Capabilities for establishing virtual environments enabling network simulation, monitoring, and testing. A virtual environment comprising hardware virtualization software emulating a variety of hardware configurations and a diverse set of operating systems is established within the physical testbed infrastructure. The use of hardware virtualization ensures scalability and fast recovery. In addition, hardware and software means will be provided for the emulation of the Internet infrastructure: virtual Internet service providers, virtual users on various connections, virtual DNS servers, web servers, file servers, peer-to-peer communication environments, etc.

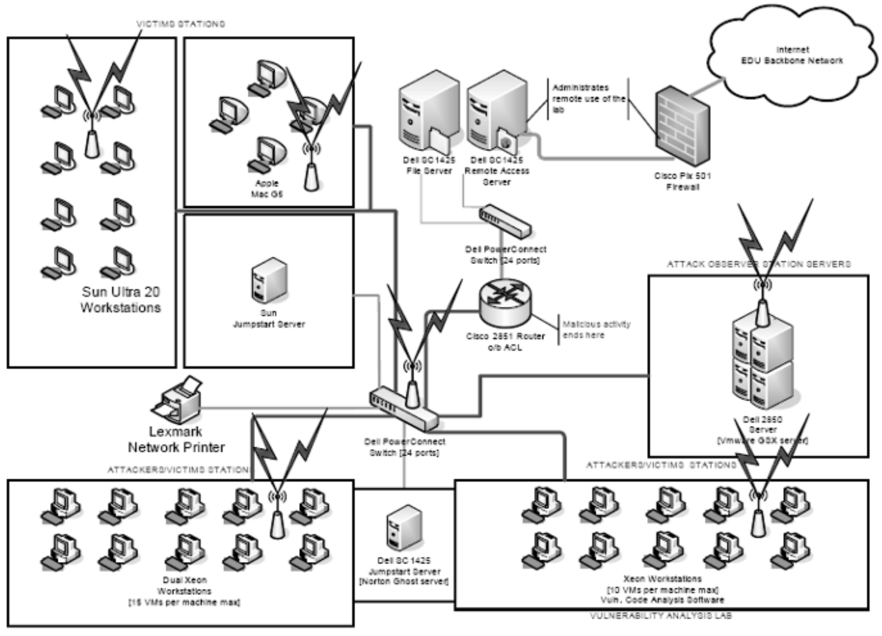

VICTIMS STATIONS
Internet
EDU Backbone Network
Administrates remote use of the lab
Dell SC1425 File Server
Dell SC1425 Remote Access Server
Cisco Pix 501 Firewall
Apple Mac G5
Dell PowerConnect Switch [24 ports]
Sun Ultra 20 Workstations
Sun Jumpstart Server
ATTACK OBSERVER STATION SERVERS
Cisco 2851 Router o/b ACL
Malicious activity ends here
Lexmark Network Printer
Dell PowerConnect Switch [24 ports]
Dell 2850 Server [Vmware GSX server]
ATTACKERS/VICTIMS STATIONS
ATTACKERS/VICTIMS STATIONS
Dual Xeon Workstations [15 VMs per machine max]
Dell SC 1425 Jumpstart Server [Norton Ghost server]
Xeon Workstations [10 VMs per machine max] Vuln. Code Analysis Software
VULNERABILITY ANALYSIS LAB

Fig. 5. Computer network testbed at Binghamton

- Support for simulation, analysis, and testing of computer defense mechanisms. A variety of hardware and software tools, including access control, intrusion detection and anti-virus tools, will provide the defensive component of the simulated networks as needed.
- Support for real-time emulation of network attacks. This component supports realistic attack scenarios to be tested within a controlled environment and includes computers deploying various types of information attacks (hardware and software attack tools).
- Support for real-time monitoring and forensic analysis of attacks This component provides the means for monitoring and post-attack damage assessment including code analysis software, forensic tools, and hi-end code disassemblers and debuggers for fast analysis of new viruses, worms and exploits, as well as for vulnerabilities analysis of legitimate software

  It could be seen that the above described computer network testbed provides an ideal environment for the implementation of the active immune system for a computer network described herein.

Acknowledgement. The authors are grateful to Dr. Robert Herklotz the Air Force Office of Scientific Research (USA) for supporting this research and to Alex Volynkin and Stephen Mazur of Binghamton University for their valuable suggestions and comments.

References

1. Skormin, V.: AFOSR CONTRACT # FA9550-05-1-0361, $ 599k, Principal Investigator
2. Skormin, V., Summervillev, D., Moronski, J., McGee, D.: Detecting Malicious Codes by the Presence of their Gene of Self-Replication. In: Gorodetsky, V., Popyack, L.J., Skormin, V.A. (eds.) MMM-ACNS 2003. LNCS, vol. 2776, pp. 195–205. Springer, Heidelberg (2003)
3. Kephart, J.O., White, S.R.: Directed–graph epidemiological models of computer viruses. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (1991)
4. Kephart, J.O., White, S.R.: Measuring and Modeling Computer Virus Prevalence. In: Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos (1993)
5. Kephart, J.O., White, S.R., Chess, S.R.: Computers and Epidemiology. In: IEEE Spectrum, IEEE Computer Society Press, Los Alamitos (1993)
6. Kephart, J.: A Biologically Inspired Immune System for Computers, IBM Thomas J. Watson Research Center, High Integrity Computing Laboratory (1994)
7. Kephart, J.O.: How topology affects population dynamics. In: Langton, C. (ed.) Artificial Life III. Studies in the Sciences of Complexity, pp. 447–463 (1994)
8. Kephart, J., Sorkin, G., Chess, D., White, S.: Fighting Computer Viruses, Scientific American (November 1997)
9. Kephart, J., Sorkin, G., Chess, D., Swimmer, M., White, S.: Blueprint for a Computer Immune System. In: The Virus Bulletin International Conference in San Francisco (October 1997)

10. Moore, D., Shanning, C., Claffy, K.: CodeRed: a case study on the spread and victims of an Internet worm. In: Proceedings of the 2nd Internet Measurement Workshop (2002)
11. Moore, D., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the Slammer worm. IEEE Security and Privacy (2003)
12. PastorSatorras, R., Vespignani, A.: Epidemics and immunization in scalefree networks. Handbook of Graphs and Networks: From the Genome to the Internet (2002)
13. PastorSatorras, R., Vespignani, A.: Immunization of complex networks. Physical Review E 65 (2002)
14. Boguna, M., PastorSatorras, R.: Epidemic spreading in correlated complex networks. Physical Review E 66 (2002)
15. Wang, C., Knight, J.C., Elder, M.C.: On Computer Viral Infection and the Effect of Immunization. In: Proceedings of the 16th Annual Computer Security Applications Conference (2000)
16. Zou, C.C., Towsley, D., Gong, W.: On the Performance of Internet Worm Scanning Strategies. Univ. Massachusetts Amherst Technical Report TR-03-CSE-07 (2003)
17. Zou, C., Gong, W., Towsley, D.: Code Red Worm Propagation Modeling and Analysis. In: Proceedings of 9th ACM Conference on Computer and Communication Security (2002)
18. Kim, J., Radhakrishnan, S., Dhall, S.: Measurement and analysis of worm propagation on Internet Network Topology. School of Computer Science, University of Oklahoma, USA (2003)
19. Wang, Y., Chakrabati, D., Wang, C., Faloutsos, C.: Epidemic spreading in real networks: an Eigen value viewpoint. In: Proceedings of 22nd International Symposium on Reliable Distributed Systems, October, 2003 (2003)
20. Zou, C.C., Gong, W., Towsley, D.: Worm Propagation Modeling and Analysis under Dynamic Quarantine Defense. In: Proceedings of WORM'03, October 2003 (2003)
21. Liljenstam, M., Nicol, D.M.: Comparing passive and active worm defenses. In: Proceedings of the First International Conference on the Quantitative Evaluation of Systems, September 2004 (2004)
22. Nicol, D., Liljenstam, M.: Models of Active Worm Defenses, Coordinated Science Laboratory, University of Illinois (2004)
23. Brumley, D., Liu, L., Poosankam, P., Song, D.: Taxonomy and Effectiveness of Worm Defense Strategies. School of Computer Science, Carnegie Mellon University (June 2005)
24. Kim, J., Radhakrishnan, S., Dhall, S.: Optimal Control of Treatment Costs for Internet Worm. In: Proceedings of WORM'04, October 2004 (2004)
25. Castañeda F., Sezer E., Xu J.: WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism. In: Proceedings of WORM'04 (October 2004)
26. Sidiroglou, S., Keromytis, A.D.: Countering network worms through automatic patch generation. IEEE Security and Privacy (2005)
27. Liang, Z., Sekar, R.: Fast and automated generation of attack signatures: A basis for building self-protecting servers. In: Proceedings of CCS (2005)
28. Liang, Z., Sekar, R.: Automatic Generation of Buffer Overflow Attack Signatures: An Approach Based on Program Behavior Models. In: Proceedings of ACSAC (2005)
29. Skormin, V., Summervillev, D., Moronski, J., McGee, D.: Biological Approach to System Information Security (BASIS): A Multi-Agent Approach to Information Security. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 435–444. Springer, Heidelberg (2003)

30. Tarakanov, A., Skormin, V., Sokolova, S.: Immunocomputing: Principles and Applications. Springer, New York (2003)
31. Volynkin, A., Skormin, V., Summerville, D., Moronski, J.: Evaluation of Run-Time Detection of Self-Replication in Binary Executable Malware. In: Proceedings of the 7th IEEE Systems, Man and Cybernetics Information Assurance Workshop (June 2006)
32. Skormin, V., Volynkin, A., Summerville, D., Moronski, J.: Prevention of Information Attacks by Run-Time Detection of Self-Replication in Computer Codes. Computer Security Journal (to appear)
33. Brown, L.D., Cai, T.T., DasGupta, A.: Interval Estimation for a Binomial Proportion. Statistical Science 16, 1101–1174 (2001)
34. Landau, Y.D.: Adaptive Control. The Model Reference Approach. Marcel Dekker, Inc.

Mathematical Models of Intrusion Detection by an Intelligent Immunochip

Alexander O. Tarakanov

St. Petersburg Institute for Informatics and Automation, Russian Academy of Sciences
14-line 39, St. Petersburg, 199178, Russia
tar@iias.spb.su

Abstract. Based on mathematical models of immunocomputing, this paper proposes an approach to intrusion detection that allows both low-level signal processing (feature extraction) and high-level ("intelligent") pattern recognition. The key model is the formal immune network (FIN) including apoptosis (programmed cell death) and immunization both controlled by cytokines (messenger proteins). FIN can be formed from the raw signal using discrete tree transform, singular value decomposition, and the proposed index of inseparability in comparison with the Renyi entropy. The speed and the accuracy of the approach probably mean a further step toward placing more of the intelligent functions on the chip.

Keywords: formal immune network, immunochip, intrusion detection.

1 Introduction

Artificial immune systems (AIS) [1], [2] and immunocomputing (IC) [3] are developing as the approaches of computational intelligence [4] like genetic algorithms and artificial neural networks (ANNs) also called neurocomputing. Recent advances in AIS [5] include a stochastic model of immune response [6], an aircraft fault detection [7], and intrusion detection [8]. Recent advances in IC include the mathematical models for biomolecular immunocomputer [9], brain research [10], [11], [12], reconstruction of hydrophysical fields [13], signal processing [14], [15], and intrusion detection [16], [17].

The present paper reports a set of mathematical models of IC specified for intrusion detection by intelligent signal processing. The key model of the approach is the formal immune network (FIN). In the training mode, FIN is formed from the raw signal using discrete tree transform (DTT) [14], [18] and singular value decomposition (SVD) [19]. After the procedures of apoptosis (programmed cell death) and immunization, the result of such feature extraction by FIN is estimated by the proposed index of inseparability in comparison with the Renyi entropy [20]. In the recognition mode, the current signal is processed by DTT, mapped to the FIN, and recognized by the cytokine class of the nearest point cell of the FIN.

The models have been implemented as a computing scheme (architecture) of an immunochip for intrusion detection and tested on a fragment of the UCI KDD archive [21].

2 Mathematical Models

2.1 Formal Immune Network

According to [22], FIN is defined as a set of cells $W = (V_1,...,V_m)$. Cell is a pair $V = (c, P)$, where class c (cytokine type of cell) is natural number $c \in N$, whereas $P = (p_1,..., p_q)$ is a point of the q-dimensional Euclidian space $P \in R^q$, which lies within the unit cube $\|P\| \le 1$, where $\|P\| = \max\{p_1,..., p_q\}$ is the Tchebyshev norm. Let the metric $d_{ij} = \|P_i - P_j\|$ be distance between cells V_i and V_j (their "affinity" or separability). Let $h \in R$ be given threshold. Let cell V_i recognizes cell V_k if they have different classes and the distance between them is lower than the threshold:

$$c_i = c_k\,,\ d_{ik} < h\,,\ d_{ik} < d_{ij}\,,\ \forall V_j \in W\,,\ j \ne i\,,\ k \ne j\,.$$

Fix some FIN1 ("innate immunity") as a non-empty set of cells: $|W_1| \ne 0$. Let FIN $W \subseteq W_1$ be a subset of FIN1. Let the behavior ("affinity maturation") of FIN is defined by the following rules.

Rule 1 (Apoptosis). If cell $V_i \in W$ recognizes cell $V_k \in W$ then remove V_i from W.

Rule 2 (Immunization). If cell $V_k \in W$ is nearest to cell $V_i \in W_1 \setminus W$ among all cells of W: $d_{ik} < d_{ij}$, $\forall V_j \in W$, whereas $c_i \ne c_k$, then add V_i to W.

Let W_A be FIN as a consequent of application of apoptosis to all cells of FIN1. Let W_I be FIN as a consequence of immunization of all cells of W_A by all cells of FIN1. Note that the resulting FIN depends on the ordering of cells in FIN1. Further it will be assumed that the ordering is given.

Consider some general mathematical properties of FIN.

It is obvious that neither the result of apoptosis W_A nor the result of immunization W_I can overcome W_1 for any FIN1 and any threshold:

$$W_A \subseteq W_1,\quad W_I \subseteq W_1,\quad \forall W_1, h.$$

It can be also shown that for any FIN1 there exists minimal threshold h_A such that apoptosis does not change W_1 for any h lower than h_A: $W_A = W_1, \forall h < h_A$.

The following Proposition gives more important and less evident property of FIN.

Proposition. For any W_1 there exists threshold h_I such that consequence of apoptosis and immunization $W_2(h_I)$ provides the minimal number of cells $|W_2|$ for given W_1 and any h: $|W_2(h_I)| \le |W_2(h)|$, $\forall h$, $\forall W_2 \subseteq W_1$.

The proof of this proposition is given in [22].
Actually, the proposition states that the minimal number of cells in FIN2 after apoptosis and immunization is a kind of "inner invariant" of any FIN, which depends on the innate immunity FIN1 but does not depend on the affinity threshold. Practically, it means that such invariant can be found for any FIN by apoptosis and immunization without considering any threshold at all.

Now we can define a general model of molecular recognition in terms of FIN.

Let "epitope" (antigenic determinant) be any point $P \in R^q$. Note that any cell of FIN also contains an epitope, which lies within the unit cube.

Let cell V_i recognizes epitope *P* by assigning him class c_i if the distance $d(V_i, P)$ between the cell and the epitope is minimal among all cells of FIN:

$$d(V_i, P) = \min\{d(V_j, P)\}, \forall V_j \in W .$$

Let pattern ("molecule") be any *n*-dimensional column-vector $Z = [z_1,..., z_n]'$, $Z \in R^n$ where $[]'$ is symbol of vector-matrix transposing (so that Z' is row-vector). Let pattern recognition be mapping of the pattern to an epitope: $Z \to P$, and recognition of the epitope by the class c of the nearest cell of FIN.

2.2 Singular Value Decomposition

Consider the mathematical model of mapping any pattern to FIN. Let $Z_1,...,Z_m$ be *n*-dimensional training patterns with known classes $c_1,...,c_m$. Let $A = [Z_1,...,Z_m]'$ be training matrix of dimension $m \times n$. Consider SVD of this matrix:

$$A = s_1 X_1 Y_1' + s_2 X_2 Y_2' + s_3 X_3 Y_3' + ... + s_r X_r Y_r',$$

where *r* is the rank of the matrix, s_k are singular values and X_k, Y_k are left and right singular vectors with the following properties:

$$X_k' X_k = 1,\ Y_k' Y_k = 1,\ X_k' X_i = 0,\ Y_k' Y_i = 0,\ i \neq k,\ k = 1,...,r .$$

Consider the following map $Z \to R^q$ of any *n*-dimensional pattern $Z \in R^n$:

$$p_k = \frac{1}{s_k} Z' Y_k,\ k = 1,...,q . \qquad (1)$$

According to [3], formula (1) can be computed as binding energy between two formal proteins: *Z* ("antigen") and Y_k ("antibody").

2.3 Discrete Tree Transform

Consider the mathematical model of forming pattern from any one-dimensional signal (time series). Let $T = \{t_1,...,t_n\}$ be a fragment of signal, where $t \in R$ is real value in general case, $n = 2^{N_0}$ and N_0 is some number exponent so that n is a power of 2.

Let $u = 2^{N_1}$, $N_1 \le N_0$. According to [14], the dyadic DTT of T is the following map:

$$T \to \{a_{u,k}\} \text{ , } a_{u,k} = \frac{1}{n} \sum_{1+(k-1)u}^{ku} t_i \text{ , } k = 1,..., 2^{N_0 - N_1} .$$

Let $l = N_1$ be DTT level: $0 \le l \le N_0$. Let us denote the DDT map as follows:

$$T \to T^{(l)} \text{ , } T^{(l)} = \{t_1^{(l)},...,t_n^{(l)}\} \text{ , } t_i^{(l)} = a_{u,k} \text{ , } 1 + (k-1)u \le i \le ku .$$

Consider the following modification of DTT.

Consider n as the size of running window. Consider a fragment of signal $T_{n,i} = \{t_{i+1},...,t_{i+n}\}$, which size is equal to n. Consider a triplet of fragments $T_{n,-n+1}T_{n,0}T_{n,n-1}$, which size is equal to $3n$. Consider the sequence of DTT of the middle fragment $T_{n,0}$ with level l and window n: $T_{n,i} \to T_{n,i}^{(l)}$, $i = -n+1, \dots, n-1$. This running window processes n times any count of signal $t_j \in T_{n,0}$ of the middle fragment: $t_j(T_{n,j-n+1}^{(l)}),..., t_j(T_{n,j-1}^{(l)})$, where $j = 1,...,n$. Thus, the result of processing $z_j = t_j(T_{n,j-1}^{(l)})$, apart from the count t_j , depends also upon $n-1$ previous and $n-1$ next counts of raw signal.

Consider the values $z_1,..., z_n$, obtained by the processing of any fragment $T_{n,i}$, as the pattern: $Z = [z_1,..., z_n]'$.

According to [14], [18], the proposed approach to signal processing is inspired by a mode of biomolecular computing [9], when immune cells chop unknown antigen to its local singularities and expose them to the immune system. Analogously, our approach represents unknown signal as a tree of data, and chop the branches of the tree at the level l to detect local singularities of the signal.

2.4 Entropy and Separability

According to the above models, the feature extraction method in general is as follows.

1. Extract training patterns from the signal.
2. Form q -dimensional FIN1 (m_1 cells).
3. Find its inner invariant FIN2 ($m_2 \le m_1$ cells).

As the result, the q -dimensional points of FIN2 $P_1,..., P_{m_2}$ are considered as the feature vectors that represent the signal.

The following task is to estimate a quality of such feature extraction. This can be done, e.g., using the special entropy proposed in [20] and proved to be rather useful metric of very large networks regarding the task of intrusion detection [23].

According to [20], [23], the Renyi entropy of *j*-th dimension of FIN can be defined as follows:

$$e_j = -\frac{1}{m}\sum_{i=1}^{m}\log_2(p_{ij}^2),$$

where $p_{1j},..., p_{mj}$ are the values of j-th coordinate of the points of FIN $P_1,..., P_m$.

Let us consider the maximal entropy as the Renyi entropy of FIN:

$$e = \max\{e_j\},\ j = 1,..., q.$$

Usually, entropy is considered as a measure of disorder. The lower is entropy the lower is disorder of the system.

Consider another metric which is more specific to FIN. Let h_I be the minimal distance between any pair of cells of FIN which have different classes:

$$h_I = \min\{d_{ij}\},\ i \neq j,\ c_i \neq c_j.$$

Let us define index of inseparability of FIN as follows:

$$f = \ln\left(\frac{m_2}{m_1 h_I}\right). \tag{2}$$

Note that m_1 is number of cells in FIN1, whereas m_2 is number of cells in FIN2 after apoptosis and immunization. Note also that $m_2 = m_1$ for FIN1. Thus, the greater is minimal distance h_I the lower is the index and the better is the separability between different classes of any FIN.

3 Computing Scheme

General computing scheme (architecture) of the approach is show in Fig. 1.

In both the training and the recognition modes, the fragment $T_{n,i}$ is extracted from the signal. Using the running window size n, this signal fragment is processed by DTT level l to extract pattern (antigen) $Z = [z_1,..., z_n]'$.

In the training mode, training matrix $A = [Z_1,..., Z_m]'$ is formed from the training patterns. Using SVD of this matrix, q right singular vectors (antibodies) $Y_1, ..., Y_q$ are determined and q-dimensional FIN1 is formed, which contains m_1 cells. Using apoptosis and immunization, FIN1 is reduced to FIN2, which contains m_2 cells. As the result, any signal fragment is mapped to the nearest point of q-dimensional FIN2: $T_{n,i} \to P_j$, $j = 1,..., m_2$, $i = 1,..., N_T$, $N_T = N_S - n + 1$, where N_S is total counts in the signal, whereas N_T is number of fragments of length n. To obtain the best feature extraction, the size of the running window n and the DTT level l are selected by the minimal value of the index of inseparability (2) of FIN2.

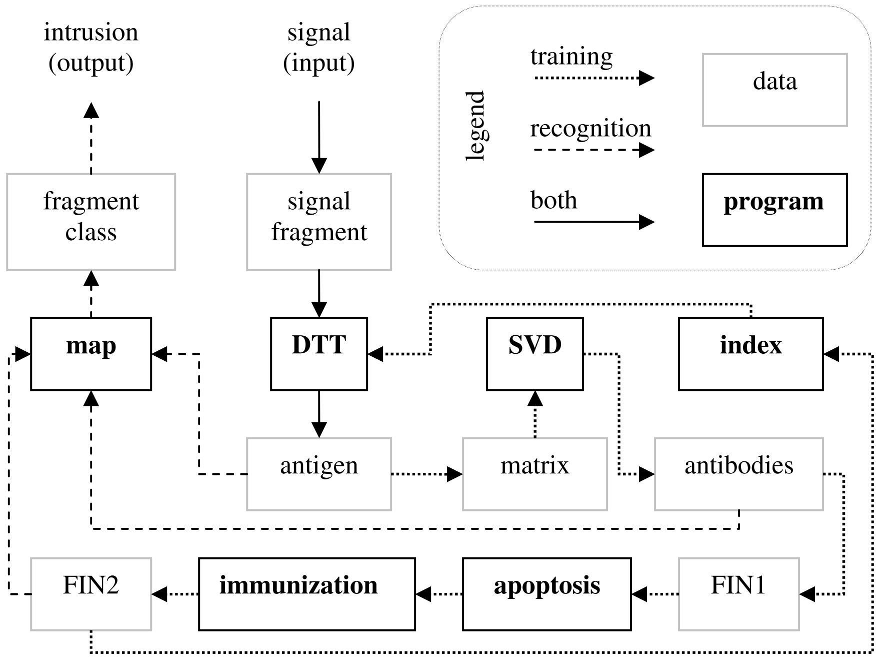


Fig. 1. Computing scheme of intrusion detection by an immunochip

In the recognition mode, the current antigen is mapped to FIN2, using the binding energies (1), and recognized by the nearest point of FIN2. If the antigen coincides with any fragment of the training signal, then it is exactly recognized. The proof of this feature is given in [22].

4 Test Examples

The file <kddcup_data_10_percent_gz.htm> (7.7 Mb) from the UCI KDD archive [21] has been utilized for testing the approach. This file contains 51608 network connection records. Any record (file string) contains 42 parameters: 1) duration, 2) protocol_type, 3) service, 4) flag, 5) src_bytes, …, 41) dst_host_srv_rerror_rate, 42) attack_type. Parameters ## 2, 3, 4, 42 are symbolic, the other ones are numerical.

Parameter #5 (src_bytes) has been arbitrary utilized as a source of signal. A value of this parameter is the number of bytes sent from a source IP address to a target IP address in the past two seconds under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type.

Consider a short example of signal (signal-1) in Fig. 2, where $N_S = 510$ counts (i.e. 17 minutes of network traffic). The type of intrusion or normal count are given in Tab. 1 where the FIN cell class is $c = \{0,...,31\}$, $c = 0$ corresponds to the normal counts, $c > 0$ corresponds to the intrusion counts and the classes of the intrusions have been taken from [16].

Consider $N_{train} = 400$ training counts and $N_{test} = N_S = 510$ test counts. Consider three-dimensional FIN: $q = 3$. Compute the length of the minimal normal (without any intrusion count) fragment of signal-1: $n_{\min} = 53$. This determines the restrictions on the size of running window: $n \geq q$, $3n \leq n_{\min}$. Thus, $n = \{4, 8, 16\}$.

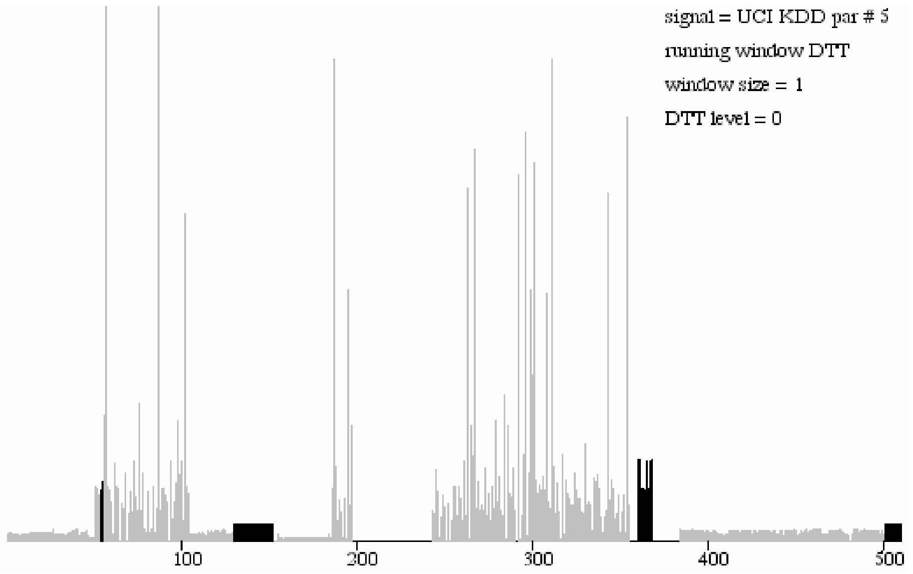


Fig. 2. Counts of signal-1 (gray – normal, black – intrusion)

Table 1. Types of the counts of signal-1

counts	attack_type	cell class	counts	attack_type	cell class
1-53	normal	0	292-355	normal	0
54-55	buffer_overflow	3	356-359	ipsweep	0
56-129	normal	0	360-368	buffer_overflow	3
130-152	smurf	22	369-370	land	8
153-197	normal	0	371-383	neptune	12
198-242	portsweep	17	384-400	normal	0
243-290	normal	0	401-500	normal	0
291	land	8	501-510	smurf	22

The mathematically rigorous feature of any FIN is the exact recognition of any pattern it has been trained. However, the training without DTT ($l = 0$) and with window $n = 4$ gives 12 errors, whereas $n = 8$ gives 8 errors. This means the ambiguities in the training data when two identical patterns correspond to different classes: $Z_i = Z_j$, $c(Z_i) \neq c(Z_j)$, $i \neq j$. Such ambiguities in the raw signal-1 can be eliminated by DTT. The results of the training and recognition are given in Tab. 2

Table 2. Feature extraction and intrusion detection in signal-1

window size	DTT level	cells in FIN	minimal distance	entropy of FIN	index of FIN	test errors	false alarms	missed intrusions
4	1	150	4.7E-4	5.5	6.7	3	3	0
	2	170	5.4E-4	6.4	6.7	1	1	0
8	1	175	2.1E-3	5.4	5.3	0	0	0
	2	192	8.3E-4	6.1	6.3	0	0	0
	3	204	2.7E-4	6.8	7.5	52	50	2
16	0	185	1.3E-3	>E10	5.9	5	5	0
	1	210	1.6E-3	6.6	5.8	1	0	1
	2	238	3.6E-4	7.2	7.4	27	26	1
	3	230	3.9E-4	6.6	7.3	18	14	4
	4	218	3.3E-4	6.4	7.4	3	3	0

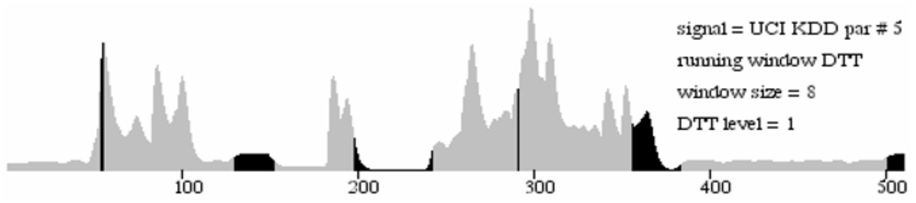


Fig. 3. DTT of signal-1

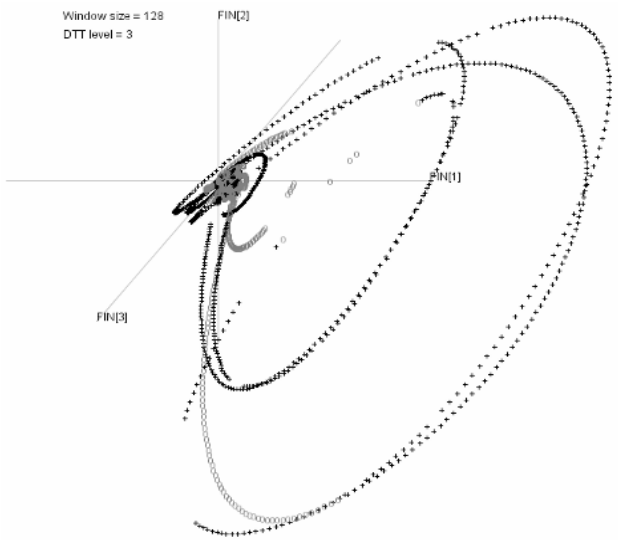


Fig. 4. Cells of the best FIN for signal-2 (gray "0" – normal, black "+" – intrusion)

where minimal distance h_I is computed between the normal cells with $c = 0$ and the intrusion cells $c > 0$, whereas the test errors (false alarms and missed intrusions) occur only on the untrained counts (## 401-510). The best FIN2, according to both the Renyi entropy and index of inseparability, has been obtained for $n = 8$, $l = 1$ (see

Table 3. Feature extraction from signal-2

window size	DTT level	cells in FIN	minimal distance	entropy of FIN	index of FIN
128	1	16518	1.6E-7	11.5	14.5
	2	37460	1.5E-7	>E10	15.4
	3	16137	1.2E-6	10.8	12.5
	4	38238	1.1E-6	>E10	13.4
	5	42998	1.2E-6	>E10	13.5
	6	42905	1.7E-6	198.9	13.1
	7	42054	8.1E-7	112.7	13.8

the selected row in Tab. 2). The signal-1 after the corresponding DTT processing is shown in Fig. 3.

Consider the full signal-2 with $N_S = 51608$ counts (i.e. 27 hours of network traffic) and all 31 types of intrusions. In this case, $n_{\min} = 744$. Thus, $n = \{4, 8, 16, 32, 64, 128\}$. The training results are best for $n = 128$ and they are given in Tab. 3. The best FIN2 has been obtained for $n = 128$, $l = 3$ (selected row in Tab. 3). This feature extraction reduces 51481 128-dimensional training patterns to 16137 cells of three-dimensional FIN (i.e. by 136 times), which is shown in Fig. 4.

5 Discussion

The developed mathematical models guarantee, e.g., that the FIN in Fig. 4 recognizes exactly any intrusion in any fragment of 128 counts of the 51608-counts signal even if the fragment contains only one count of intrusion on the background of all other counts normal. The feature extraction by this FIN provides the compression rate near 136 for the number of real values which are needed to store for the exact recognition. The training time of this FIN is about 1.5 minutes (AMD 1.5 GHz).

Such faultless recognition of FIN with rather low training time look unobtainable for its main competitors like ANNs in the field of intelligent signal processing. For example, a comparison in [13] shows that a FIN needs just 21 runs (about 20 seconds) to determine its optimal parameters where an ANN trained by the error back propagation needs 1750 runs (about 24 hours!) for the same purpose still without any guarantee that, say, a huge number of hidden neurons may not minimize the obtained errors of the ANN.

The mathematically rigorous feature of FIN is the exact recognition of any pattern it has been trained. This feature allows using FIN to disclose the ambiguities in any data, e.g., like in the task of identification of cellular automata [24]. This feature is beyond the capabilities of ANNs, e.g., due to irreducible training errors and the known effect of overtraining when the attempts to reduce the errors may lead to their drastic increase [13].

The proposed index of inseparability looks more convenient than the Renyi entropy for FIN. Tables 2 and 3 show that the index takes quite reasonable values when the entropy tends to infinity though the corresponding FIN remains quite capable. On the other hand, the values of the index and the entropy are quite comparable when the FIN appears to perform the best feature extraction.

It is worth noting that any parameter (##1-41 of [21]) can be utilized (in Section 4) as a source of signal. In such task statement, the IC approach is capable to select the most useful parameter(s) for intrusion detection (e.g. by index of inseparability of FIN). The approach is especially effective over any combination of the parameters. For example, intrusion detection by the combination of all parameters (##1-41) was reported in [16].

It is also worth noting that the proposed approach has nothing common with a statistical analysis technique. Moreover, it is well-known that statistical methods (e.g. Markov chains, Bayesian networks etc.) are too slow and inaccurate to cope with real-world signals. For example, our comparison of image recognition in [25] shows that the IC works 10 time faster and more accurate than conventional statistics. No wonder that more and more of modern approaches to signal processing are based on wavelet analysis, where the dyadic DTT (Section 2.3) is also a wavelet-type transform for signal analysis [26], [27].

Hardware implementation of the developed approach can be provided by two steps: 1) its hardware emulation on digital signal processor (DSP) of super Harvard architecture (SHARC) of new TigerSHARC family (which is capable to cope with floating point operations) and 2) modification of the hardware emulator to a proper immunochip. At the first step, such hardware emulator can be coupled with a network router and thus tested and adjusted using data of real traffic.

6 Conclusion

The presented IC approach of computational intelligence [15] is based essentially on the mathematical models of information processing by proteins and immune networks [3]. It is worth highlighting that the models have already appeared to be useful also in such fields as biomolecular computing [9], [25] and brain research [10], [11], [12].

On such background, the mathematical model of FIN with apoptosis and immunization controlled by cytokines [22] represents the key model of the IC approach to intelligent signal processing and intrusion detection [17]. This means that the other models reported here (DTT, SVD, Renyi entropy) are not so critical for feature extraction and pattern recognition since they were utilized just to provide a kind of convenient zero-approach ("innate immunity" of FIN – see Section 2.1) to be optimized then by FIN using apoptosis and immunization.

For example, any set of fixed-length fragments of unprocessed signal can be considered as the points of FIN (Section 2.1) without any preprocessing, e.g., by methods of DTT, SVD, statistics, Fourier, wavelets etc. Then the reduced set of points after apoptosis and immunization can represent the feature extraction by FIN and the quality of such FIN can be estimated by the index of inseparability and thus compared with other FINs (e.g., those obtained by a preprocessing of the signal).

On the other hand, let us note once again that the SVD (Section 2.2) can model the binding energy between two proteins [3], whereas the dyadic DTT (Section 2.3) can model an immune-type antigen processing [14], [27].

Last but not least, the results of numerical experiments reported in [13], [16] suggest that the speed and accuracy of the IC approach is probably unobtainable for other robust methods of computational intelligence (in particular, neurocomputing and evolutionary algorithms). These advances of the IC approach together with its biological nature probably means a further step toward placing more of the intelligent functions on the chip.

References

1. Dasgupta, D. (ed.): Artificial Immune Systems and Their Applications. Springer-Verlag, Berlin (1999)
2. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag, London (2002)
3. Tarakanov, A.O., Skormin, V.A., Sokolova, S.P.: Immunocomputing: Principles and Applications. Springer-Verlag, New York (2003)
4. Fogel, D.B., Robinson, C.J. (eds.): Computational Intelligence: The Experts Speak. IEEE Press, Piscataway NJ (2004)
5. Dasgupta, D.: Advances in artificial immune systems. IEEE Computational Intelligence Magazine 1/4, 40–49 (2006)
6. Chao, D.L., Davenport, M.P., Forrest, S., Perelson, A.S.: A stochastic model of cytotoxic T cell responses. Journal of Theoretical Biology 228, 227–240 (2004)
7. Dasgupta, D., Krishna-Kumar, K., Wong, D., Berry, M.: Negative selection algorithm for aircraft fault detection. In: Nicosia, G., Cutello, V., Bentley, P.J., Timmis, J. (eds.) ICARIS 2004. LNCS, vol. 3239, pp. 1–13. Springer, Heidelberg (2004)
8. Dasgupta, D., Gonzalez, F.: Artificial immune systems in intrusion detection. In: Rao Vemuri, V. (ed.) Enhancing Computer Security with Smart Technology, pp. 165–208. Auerbach Publications (2005)
9. Goncharova, L.B., Jacques, Y., Martin-Vide, C., Tarakanov, A.O., Timmis, J.I.: Biomolecular immune-computer: theoretical basis and experimental simulator. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 72–85. Springer, Heidelberg (2005)
10. Agnati, L.F., Tarakanov, A.O., Ferre, S., Fuxe, K., Guidolin, D.: Receptor-receptor interactions, receptor mosaics, and basic principles of molecular network organization: possible implications for drug development. Journal of Molecular Neuroscience 26/2-3, 193–208 (2005)
11. Agnati, L.F., Tarakanov, A.O., Guidolin, D.: A simple mathematical model of cooperativity in receptor mosaics based on the "symmetry rule". BioSystems 80/2, 165–173 (2005)
12. Goncharova, L.B., Tarakanov, A.O.: Molecular networks of brain and immunity. Brain Research Reviews (in press) (2007)
13. Tarakanov, A., Prokaev, A., Varnavskikh, E.: Immunocomputing of hydroacoustic fields. International Journal of Unconventional Computing (in press) (2007)
14. Atreas, N.D., Karanikas, C.G., Tarakanov, A.O.: Signal processing by an immune type tree transform. In: Timmis, J., Bentley, P.J., Hart, E. (eds.) ICARIS 2003. LNCS, vol. 2787, pp. 111–119. Springer, Heidelberg (2003)
15. Tarakanov, A., Nicosia, G.: Foundations of immunocomputing. In: Proc. 1st IEEE Symposium on Foundations of Computational Intelligence (FOCI'07), Honolulu, Hawaii, pp. 503–508 (2007)

16. Tarakanov, A.O., Kvachev, S.V., Sukhorukov, A.V.: A formal immune network and its implementation for on-line intrusion detection. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 394–405. Springer, Heidelberg (2005)
17. Tarakanov, A., Kryukov, I., Varnavskikh, E., Ivanov, V.: A mathematical model of intrusion detection by immunocomputing for spatially distributed security systems. RadioSystems 106, 90–92 (2007) (in Russian)
18. Karanikas, C., Prios, G.: A non-linear discrete transform for pattern recognition of discrete chaotic systems. Solitons and Fractals 17, 195–201 (2003)
19. Horn, R., Johnson, C.: Matrix Analysis. Cambridge University Press, Cambridge (1986)
20. Renyi, A.: On measures of entropy and information. In: Proc. 4th Berkeley Symposium on Mathematics, Statistics and Probability, vol. 1, pp. 547–561 (1961)
21. Bay, S.D.: The UCI KDD Archive Irvine, CA: University of California, Dept. of Information and Computer Science (1999), http://kdd.ics.uci.edu
22. Tarakanov, A.O., Goncharova, L.B., Tarakanov, O.A.: A cytokine formal immune network. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 510–519. Springer, Heidelberg (2005)
23. Johnson, J.E.: Networks, Markov Lie monoids, and generalized entropy. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 129–135. Springer, Heidelberg (2005)
24. Tarakanov, A., Prokaev, A.: Identification of cellular automata by immunocomputing. Journal of Cellular Automata (in press) (2007)
25. Tarakanov, A., Goncharova, L., Gupalova, T., Kvachev, S., Sukhorukov, A.: Immunocomputing for bioarrays. In: Proc. 1st Int. Conf. on Artificial Immune Systems (ICARIS'02), Univ. of Kent at Canterbury, UK, pp. 32–40 (2002)
26. Kozyrev, S.V.: Wavelet theory as p-adic spectral analysis. Izvestia: Mathematics 66(2), 367–376 (2002)
27. Atreas, N.D., Karanikas, C., Polychronidou, P.: Signal analysis on strings for immune-type pattern recognition. Comparative and Functional Genomics 5, 69–74 (2004)

A Novel Intrusion Detection System for a Local Computer Network

A. Tokhtabayev[1], A. Altaibek[2], V. Skormin[1], and U. Tukeyev[2]

[1] Binghamton University, Binghamton NY, USA
vskormin@binghamton.edu
[2] Kazakh National University, Almaty, Kazakhstan
wtoke@kazsu.kz

Abstract. Local computer networks at major universities are routinely plagued by self-replicating malicious software. Due to the intensive exchange of data and information within the network, when modern viruses, worms and malicious software are introduced they propagate very quickly, leaving little or no time for human intervention. Such environments are ideal for the implementation of the automatic IDS described herein. It employs the Dynamic Code Analyzer (DCA) that detects malicious software during run time by monitoring system calls invoked by individual processes and detecting subsequences (patterns) of system calls indicative of attempted self-replication. A similar approach, also utilizing system calls, is developed for the detection of network worms. Both techniques have the potential for detecting previously unknown malicious software and significantly reducing computer resource utilization. Unfortunately, in comparison with traditional signature based antivirus software, both approaches have a much higher rate of false alarms. To address this shortcoming the authors propose a method to search for evidence of the alarm propagation within the network. This is achieved by aggregating alarms from individual hosts at a server where these alarms can be correlated, resulting in a highly accurate detection capability. Such a system, implementing the presented technology, and capable of significantly reducing the downtime of networked computers owned by students and faculty, is being implemented at the computer network at the Kazakh National University.

Keywords: decision-making under uncertainty, utility, possibility theory, inclusion index, comonotone fuzzy sets, Choquet integral.

1 Background

The first Intrusion Detection System (IDS) utilizing system calls was proposed in [5]. Today, these systems utilize two main approaches, signature-based and anomaly-based detection. Both of these approaches use descriptions of known attacks expressed in terms of system calls. Although signature-based systems can provide high level of accuracy, they fail to detect previously unknown attacks and rely on an ever-growing database. Anomaly detection systems utilize models of normal behavior of legitimate processes. These systems check the consistency between the invoked system calls and the profile of normality for a given process and have the potential to detect unknown attacks. The main drawback with these systems lies in the fact that they frequently suffer from a high rate of false positives. The feasibility of anomaly detection based solely on system calls is shown in [5,3,10]. The efficiency of this approach can be enhanced further by the analysis of system call attributes [9,8,11,2,13,7]. The signature-based IDS could be best exemplified by [1,6].

This research attempts to provide a solution for anomaly-based IDSs that are impractical due to a high rate of false positives. In our view, the limited success of known research aimed at the alleviation of this problem [3,10,11,13] can be traced to the fact that research was primarily aimed at improving the accuracy of normality models rather than achieving a higher level of confidence in classifying the anomaly itself. The two contributions of this paper are as follows:

a). A novel host-level anomaly detection mechanism is proposed.

b). We declare a unique but rather simple idea called the anomaly propagation principle, that is suggested as the basis for establishing, with a high degree of confidence, whether or not, a detected anomaly is a manifestation of an information attack.

In the anomaly detection mechanism we utilize non-stationary Markov models. While shell codes (in buffer overflow attack) may use only 50-60 system calls that would certainly be concealed in a histogram, Markov models are clearly preferable to other order-insensitive techniques (such us frequency histograms) used to model normality profiles [5,10]. However, we deviate from the common assumption that the source (application or service) is a stationary stochastic process.

2 Dynamic Code Analyzer

Most information attacks are carried out via the Internet through the transmission of files that contain the code of a computer virus or worm. Upon receipt, the target computer executes the malicious code resulting in the propagation of the virus or worm and the delivery of its potentially destructive payload. Self-replication, which is uncommon in legitimate programs, is vital to the spread of computer viruses and worms.

As with any function, self-replication is programmed and the sequence of operations is present in the computer code of the virus. The implementation of the self-replication function is not unique and there are several sequences of operations that can perform this task. Moreover, it is expected that these sequences are dispersed throughout the entire body of the code and cannot be detected as an explicit pattern. However, self-replication can only be achieved using a finite number of methods. Consequently, developers of new malicious software are forced to utilizing the same self-replication techniques repeatedly.

A novel virus detection technology based on these principles, known as the detection of the gene of self-replication (GSR), is presented in [9]. Detection is conducted at run-time during normal code execution under regular conditions by monitoring the behavior of every process with regards to the operating systems system calls, their input and output arguments and the result of their execution. Unlike existing anti-virus software, this methodology facilitates proactive protection from both known and previously unknown attacks.

A computers operation is facilitated by an operating system, which abstracts details of the hardware from application software. Programs interface with the operating system through the Kernel Application Programming Interface via system calls. System calls play a major role in characterizing the behavior of both malicious and legitimate computer programs. Unlike a computer program, system calls provide unambiguous information on what the computer actually does. Since self-replication is common and fundamental property of the most viruses, the search for malicious behavior can be narrowed down to the search for self-replication activity, or the GSR, in the sequences of system calls. This concept is generic in its nature; therefore, it can be applied to any computer system without necessarily binding it to a specific operating system.

The GSR is viewed as a specific sequence of commands passed to the computer operating system by program code that causes this code to replicate itself through the system or networks. Computer viruses can employ different software APIs, from hijacking a simple email client to interfacing with very complex OSs. Nevertheless, the most sophisticated and versatile viruses are still implemented in assembly language (ASM) and assembled into executable files. Since computer viruses are expected to self-replicate, and this task cannot be accomplished without interfacing the operating system, monitoring and analyzing system calls to certain OS APIs provides the means for detecting this common feature of malicious software.

Virtually every process produces system calls, however, they can easily be differentiated for every process and thread. In all cases, system calls represent a direct timeline sequence of events, which can be analyzed during execution. The GSR is contained within the sequence produced by a malicious process, and possibly dispersed throughout that sequence. Since none of the system calls alone can be considered malicious, only the particular sub-sequences of calls can form the GSR. The GSR is best described using the concept of building blocks, where each block performs a part of the chosen self-replication procedure. Any piece of software for a variety of legitimate reasons can individually perform most of the building blocks involved in malicious self-replication activity. Only when integrated into larger structures and based on their inter-functional relationships, are these building blocks indicative of attempts to self-replicate [9].

The GSR can be composed of such blocks in various ways. Therefore, its structure can be viewed as a regular sentence being built up by concatenating phrases, where phrases are built up by concatenating words, and words are built up by concatenating characters. One of the major reasons for applying such a syntactic approach to describing the GSR is to facilitate the recognition of sub-patterns.

This implies the recognition of smaller building blocks first, establishing their relevance and contribution to the replication, and then considering the next subpattern. This process is consistent with text analysis that includes recognizing characters first, and then concatenating them into words, running a spell checker on an entire word to check for mistakes, and then continuing to concatenate words into phrases and sentences checking for correct grammar and punctuation. The syntactic description of the GSR provides a capability for describing and detecting large sets of complex patterns by using small subsets of simple pattern primitives. It is also possible to apply such a description any number of times to express the basic structures of a number of gene mutations in a very compact way. The relationship between different blocks of the GSR and system calls could be very complex and can be accurately established based solely on system calls attributes. The number of attributes that make up the basic building blocks is approximately 40. Some of these attributes indicate direct relations among different system calls that could be utilized to bind system calls together to define blocks and bind blocks to establish the particular mutation of the GSR [9].

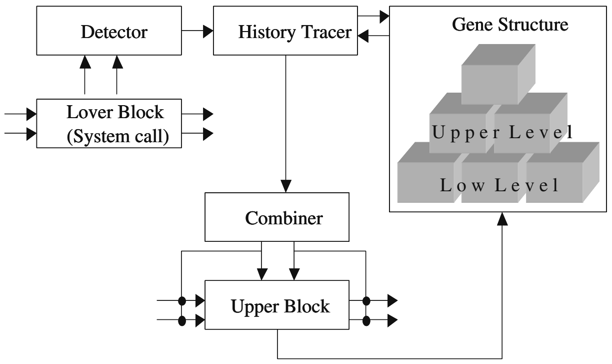


Fig. 1. GSR Detection on the Process Level

Figure 1,2 illustrates the GSR detection processes. Every system call is directed to the Replication Detector, where it is analyzed using a complete range of different detection and filtration mechanisms. Following the concept of decoupling of the GSR definition, the detection process is also highly decoupled to ensure compatibility and to reduce false detections. Just like the GSR is formed from many different building blocks, the detection mechanism observes and makes decisions regarding every block separately, until it finally reaches the top of the GSR pyramid structure and triggers the alarm mechanism. As soon as a system call is detected, the History Tracer communicates with the database, where the GSR structure is defined, to determine if this system call can be combined with any other lower level blocks to form a larger structure. When such a combination is possible, the Combiner takes two chosen lower level blocks and forms a single upper level block so that its inputs are identical to the inputs

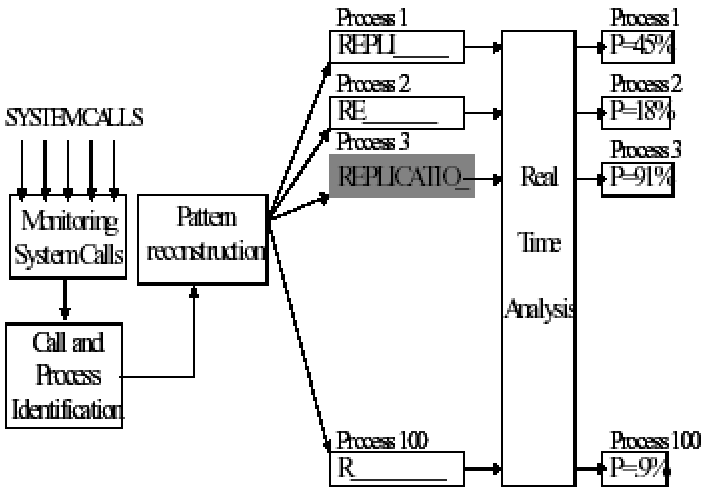


Fig. 2. GSR Detection on the Computer Level

of the Lower Block taken from the history, and the outputs are inherited from the newly detected Lower Block. When the new Upper Block is finally formed, the history is updated and the algorithm repeats itself, but does so cognizant of this newly created block. At every repetition, the detection is taking place at a higher level, as though climbing up the pyramidal structure.

As per Figure 2, the DCA detects and suspends a process that almost completes a GSR sequence, allowing the user to resume or terminate the process. Although application of the DCA may not prevent damage caused by malicious code to a particular individual computer, it allows for containment of the virus within the computer, preventing it from spreading over the network and creating a costly computer epidemic. It is equally important that the DCA is continuously running in the background and consumes less than 5% of a modern PCs resources. A prototype DCA system was successfully developed and implemented.

3 Dealing with Computer Worms

Unlike a virus that copies itself to a new file or appends itself to some victim file on local host, a worm does not have to create any new files or access existing files locally. Hence, worms may avoid leaving extensive traces of file manipulation activity in the system call domain. However, in order to perform self-replication on the network level, the worm must send a shell code to the victim process, which will execute the shell code, invoking system calls, what will certainly be reflected in the process behavior and could be detected as an anomaly.

Attack packets of a worm consist of an exploit vector followed by a shell code being a propagation engine. Modern worms could be multi-exploit and packet polymorphic, however they are expected to utilize the same propagation engine for every instance of the attack. The biggest collection of exploit payloads (www.metasploit.com) presents only a few propagation engines including: bind shell, reverse shell thread injection, and remote call of upload procedure. For instance, for bind shell, a propagation engine runs the command shell having the input bound to a socket associated with a port created by the compromised process. Worm developers usually employ the same propagation engine in their new worms. As a result, the same worm functionality would be carried out at every instance of an attack. One can see the advantage of tracing system calls instead of the packet contents. While an adversary may write a polymorphic worm, whose attack packet payload may be different from instance to instance (causing significant changes in the worms binary signature), every attack would utilize a functionally-invariant propagation engine, thus leading to the utilization of the same system calls. In this sense, a worm (similar to a virus), exhibits its gene of self-replication defined in the system call domain, the only difference is that the GSR signature of a worm offers very little information for signature matching.

To address this specificity of computer worms, the following approach to worm detection, different from the DCA was suggested in [12]. As a worm propagates, it alters the system call profile of the victim process. This altered profile could be detected as an anomaly at the host level. Every copy of the worm would carry out the same activity resulting in similar system calls traces, and similar anomalies that could be detected in the victim processes of other hosts. Therefore, the presence of self-propagation of a worm could be revealed through abnormal sequences of system calls occurring at different hosts on the network. We call this phenomenon the anomaly propagation. To distinguish the anomaly propagation from a set of coincidences (meaningless occurrences of the same anomaly at different nodes of the network), one has to analyze the connectivity pattern of the nodes. If the anomaly propagation pattern is consistent with the connectivity pattern between hosts, we claim that anomaly propagation takes place due to malicious activity. On the other hand, if anomaly propagation is inconsistent with the connectivity pattern, we declare a false positive. It could be seen that anomaly detection is the crucial component of the described approach.

Applications and services utilize high-level functions intended to solve various tasks. When an application executes several correlated tasks or a group of tasks that affect each other, it operates in one of its distinct phases (distinguished by functionality) and achieves different goals. These phases have their own realizations with respect to system calls and inevitably have different unconditional, as well as conditional, statistical distributions of system calls. Therefore, the system call profile for an entire application or service should be modeled as a non-stationary stochastic process.

Operation phases consist of many system calls and implement some strictly prescribed high-level tasks. This consideration assumes the source to be stationary during each operation phase. Since properties of the underlying stochastic process appear to be invariant over an operation phase, one can model individual operation phases by Markov chains. Therefore, the entire process or application would be modeled by a set of Markov models corresponding to particular operation phases. In this context, the sequence of system calls is considered anomalous if it is not likely to happen according to the specific Markov model (corresponding to a specific operation phase). Hence, the anomaly score can be chosen as the predicted performance characteristic of the Markov model over the observed sequence of system calls and quantified by a chi-square likelihood function of the observed sequence of a certain window. If the chi-square likelihood function exceeds a specified threshold, we declare the observed sequence to be an anomaly.

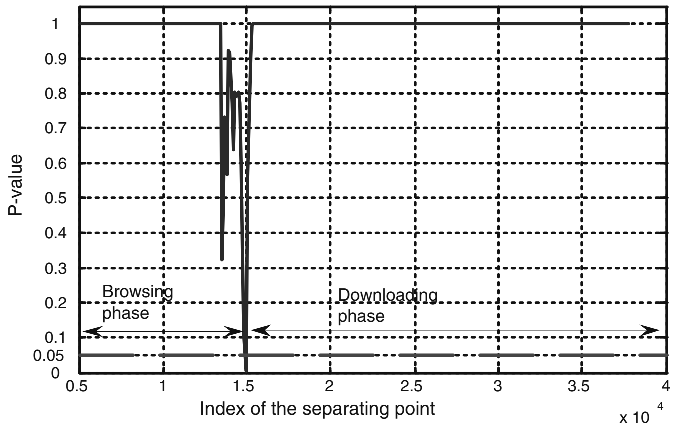


Fig. 3. Bifurcation point detection

Prior to deriving Markov models, operation phases must be distinguished automatically in an unsupervised fashion. This could be done by applying a technique that would determine bifurcation points (moments of dramatic change in process properties) within the given sequence of observations (system calls). These bifurcation points would certainly correspond to the moments of operation phase switching. One of the most efficient techniques for detecting bifurcation points is the moving omnibus method that is a simple extension of the classical omnibus method utilizing Pearsons χ^2 hypothesis test [4]. Figure 3 above illustrates the application of the approach [4] for the detection of the cut-off point between the browsing and downloading phases of operation of the Internet Explorer (p-value is a parameter quantifying the appropriate significance analysis).

Local stationary segments of the observed system call sequence are to be used to derive stationary Markov models corresponding to operation phases of the most common processes. To achieve superior consistency the application of high order Markov models is suggested. The methods for defining the order of Markov models for the given observation are described in [4].

During the testing regime, the phase of operation of the application in question is recognized in real time, and the corresponding Markov model is applied. The problem of matching current outcomes to a set of models was addressed in several publications (e.g., Stolfo [10]). A simple approach utilizing different models from the set and selecting the most appropriate one, according to a certain distance metric, was employed. It implied that if the current Markov model is not consistent with the observed system calls according to some performance metric (likelihood ratio), the system searches for the model having the best performance. To avoid undesired frequent model change we introduced a constraint on the minimum number of system calls before model switching is allowed. Figure 4 below illustrates the concept of the local anomaly detection.

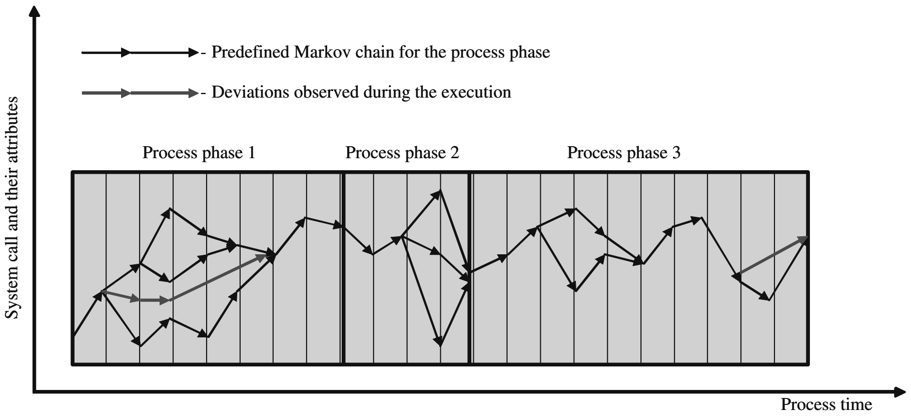

- Predefined Markov chain for the process phase
- Deviations observed during the execution
System call and their attributes
Process phase 1
Process phase 2
Process phase 3
Process time

Fig. 4. Anomaly detection on a multi-phase process

4 Server-Level Analysis of Local Alarms

Conventional anti-virus tools, available to every user, detect malicious software by matching a binary signature of the process in question to the binary signatures of known viruses and worms stored in the database. While this approach is very dependable, it has major drawbacks: it relies on an ever-growing database; requires periodic updates; consumes ever-growing share of computer resources; and is incapable of detecting previously unknown malicious codes. At the host level, implementing the techniques described above are free of these drawbacks (at least to some extent), however, they typically have a high rate of false positives that significantly limits their practicality. The utilization of the anomaly/alarm propagation concept provides a realistic opportunity for a significant reduction of the false alarm rate resulting in a new generation of IDSs operating at the server level and capable of such advanced features as the detection of new viruses and worms and the detection of distributed and evolving attacks.

Figure 5 below illustrates the principle of operation of a network security system operating on the server level where anomalies and alarms detected in the individual hosts are reported to the server level, and are correlated with the network topology and connectivity history.

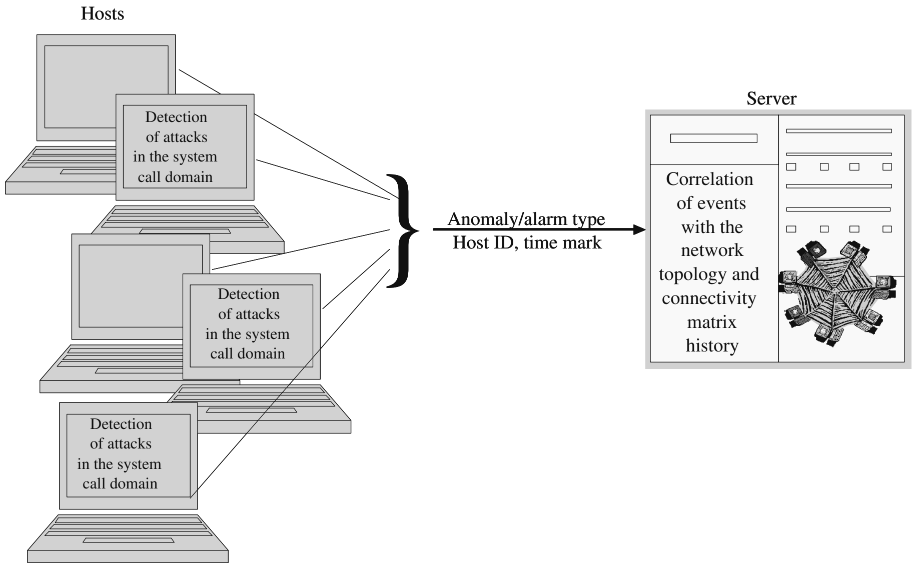

Hosts
Detection of attacks in the system call domain
Detection of attacks in the system call domain
Detection of attacks in the system call domain
Anomaly/alarm type
Host ID, time mark
Server
Correlation of events with the network topology and connectivity matrix history

Fig. 5. IDS operation principle

Anomaly propagation analysis requires distinguishing similar or equivalent abnormalities indicative of the same propagation mechanism. Anomalous segments of system calls detected on the hosts are separated into thread specific sequences, reported to the server, and provided with a time stamp and source host ID. Detectors must then split abnormal segments into thread specific sequences since a worm exploit payload is executed by one thread.

The collection server compiles the anomalies into different groups containing equivalent system call sequences. The distance between two anomalies is determined using the following equation:

$$\underline{d}(S_1, S_2) = 1 - \frac{\max\left(|C(S_1, S_2)| \,|\, (|C_m| > k)\right)}{\min(|S_1|, |S_2|)} \tag{1}$$

where $|C(S_1, S_2)|$ - length of common subsequence of anomalous strings and C_m - minimal common factor of the compact set of factors representing the subsequence. The numerator specifies the value of k as the minimum length of the common segments of two sequences, (S_1, S_2), which are declared to be equivalent.

Initially, we used just the length of the longest common segment, but eventually we realized that this definition would be significant only for propagation engines having a linear execution path. If a propagation code has branches in the execution path, then the common factor may be fragmented and some segments may diverge, causing two anomalies to be recorded as part of different attack instances. The distance (1) tolerates the existence of relatively small call execution branches within the matching segments. In fact, metric (1) is not really a distance, since it does not satisfy triangular inequality, however, it seems to be a suitable measure of anomaly similarity.

An anomaly equivalence measure can be defined with respect to distance (1) as well as another metric:

$$\overline{d}(S_1, S_2) = 1 - \frac{\max\left(|C(S_1, S_2)| \,|\, (|C_m| > k)\right)}{\max(|S_1|, |S_2|)} \tag{2}$$

One can see that (1) and (2) compute the percentage of the non-common part for shorter and longer system call strings respectively. Distances (1) and (2) provide the basis for the establishment of specific groups of equivalent anomalies as defined in terms of system calls.

During the execution, a reported anomalous string (sequence) would first be checked against the existing groups and then be added to the appropriate group as an equivalent to the anomalies contained in the group. Formally, anomaly S is added to the group G that contains representing string R if the following equivalency relation φ holds:

$$S\varphi S_i \in G \;\; iff \;\; (\underline{d}(S,R) < \tau_1)\,\&\, \left(\overline{d}(S,R) < \tau_2\right) \tag{3}$$

One can observe that the equivalency relationship (3) is reflexive, commutative, and transitive for group members. Hence, group members are formally equivalent according to the relationship (3).

If none of the groups is equivalent to the new anomaly, the system will continue comparing the anomaly with the anomalies already contained in the pool to find a candidate for a new group. We should point out that an anomaly cannot be added to the group if the group already contains an anomaly reported by the same host. To avoid pool inflation, close anomalies from the same host are represented by the single longest anomaly reported by the host.

The system tracks group size (number of members) and if the size exceeds a certain threshold, the system analyzes the pattern of propagation of anomalies in the group to expose the replication. Anomaly propagation is considered to have a replication pattern if it is consistent with the connectivity graph G in both topological and time sense. In other words, if an anomaly is replicating, it must propagate according to the following simple rules:

1. Each new instance of anomaly (except the first one) must occur in the process, which has recently been created by another suspicious process (which was already reported as an anomaly).

2. The time elapsed from the last interaction to the current anomaly occurrence must be less than the prescribed threshold (active window).

For the multipartite attack (coordinated multi source malicious activity), the first rule must allow for several sources simultaneously.

The system computes the replication score in the following way: $\max(|V'|)\big/|V|$, where V' is a node set of the graph $G'(E', V')$ being a connected sub-graph of the connectivity graph G. The score is compared to the threshold to decide if anomaly propagation is indeed a replication-related activity. The score takes into account the relative number of instances matched to replication pattern and shows the extent to which the alarm propagation is consistent with the replication activity.

5 The Implementation Aspects and Results

Based on the models presented in the previous sections, the authors have implemented an IDS operating in a MS Windows environment. The IDS consists of a client part and server part. Client agents, installed on every host monitor system calls invoked by selected processes to reveal anomalies. The server application receives abnormal sequences from the clients and performs anomaly propagation analysis. In order to reduce the overhead, the IDS client does not monitor and analyze system calls of the selected process before the process interacts with a remote host. After detecting such an interaction, the IDS client analyzes system calls to establish the fact of normal process operation or abnormal activity. Currently, we use the WinPCAP library to detect 3-way handshake of a TCP session. It should be pointed out that the system currently analyzes only SYN packets captured and reported by the WinPCAP driver working in kernel-level packet filtering mode to minimize the overhead for the user-level packet processing.

The authors do realize that the anomaly propagation pattern could be specifically factorized if the worm were to cyclically use a different payload at different generation steps. To address this problem we propose extending propagation analysis by combining graphs and then by further analyzing the resultant general graph. This graph will have nodes belonging to one of the combined groups. Thus, if the worm propagates using different exploit payloads, the combined graph will preserve propagation structure, demonstrating alarm propagation that may reflect several different anomalies.

The presented IDS technology has been tested with respect to computer worm attacks. Experiments were performed in a network testbed using 200 virtual machines with vulnerable versions of Windows XP. We experimented with the bind shell propagation engine that is common for a class of worms including: W32.Sasser, W32.Blaster, W32.Duster, W32.Welchia, W32.Reatle, W32.Cycle, etc.

For our experiment we selected the W32.Welchia worm as a class member, which according to Symantec.com, is notable for its high geographical distribution and traffic load. In order to use this worm in our testbed, we had to reverse-engineer and modify the original worm to make if more effective.In order to make its propagation faster we removed the ISS exploit, date checking, and

Blaster worm subverting engine. Additionally, we restricted the victim scan space. The modified version exploits the RPC DCOM vulnerability and only scans vulnerable and reachable hosts, resulting in an increased propagation rate.

First, we checked the predicting capability of non-stationary models versus stationary models for several legitimate processes (services) including: Svchost (RPC DCOM), Internet Explorer, LSASS, CCAPP, etc. Model generation from the system call sequences was performed offline using MATLAB. In all cases the non-stationary models showed better performance than the stationary models. Figure 6 below depicts system call prediction performance for Svchost (RPC DCOM) process based on the stationary Markov model versus the non-stationary model. The non-stationary model contains two second order Markov chains. The graph shows the chi-square log-likelihood ratio statistic that formally reflects the prediction performance (the lower the statistic, the better the prediction). Examining the curves one can see that the non-stationary Markov model (solid line) totally outperforms the stationary model (dashed line) in predicting system calls.

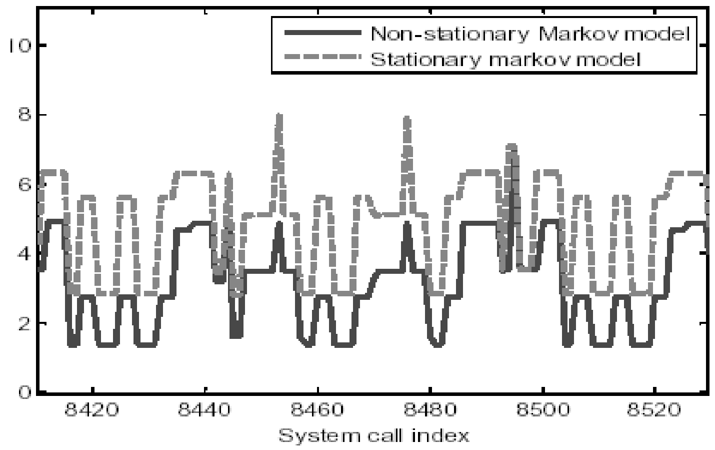


Fig. 6. Performance prediction of Svchost based on stationary and non-stationary Markov models

In the second experiment, we launched the W32.Welchia worm 20 times with arbitrarily chosen attack deployment nodes. We set the threshold for the replication score to 0.75 and the minimum size of the group to 5 hosts. In Figure 7 the solid line (left axis) shows the number of infected hosts before anomaly propagation detection. The dashed line (right axis) represents the mean distance between the string of the group that raised an alarm, and the members of that group. In every attack instance the system detected anomaly propagation after 5 to 13 host infections and on average after 8 host infections. The mean distance fluctuates from 0.05 to 0.18, demonstrating high member similarity. We did not observe any false positives (group spawned by non-worm sequences

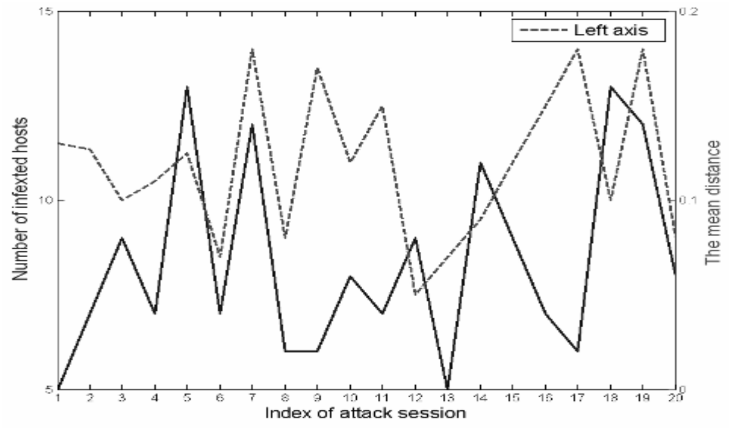


Fig. 7. Number of infected hosts and anomaly group score

being attributed to propagation), demonstrating the high performance of this IDS.

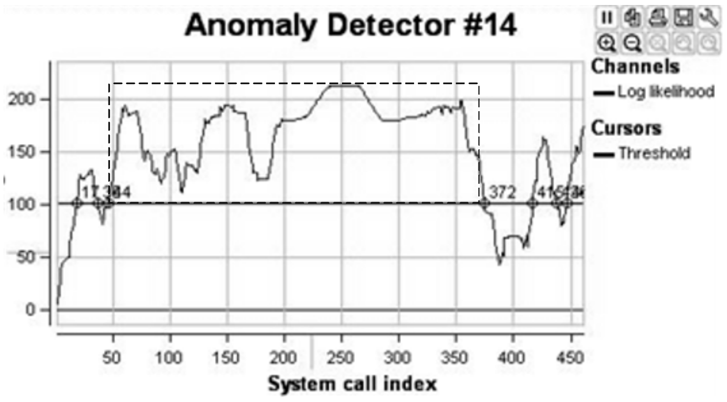


Fig. 8. Attack detection at the arbitrary chosen host

For the arbitrary chosen infected host #14, Figure 8 shows the likelihood function (LF) recorded during the attack. One can see that the attack is detected reliably (wide spike) and the segment having a LF higher than 100 was treated as an anomaly and reported to the server.

The authors are aware of the possibility of obstructing propagation detection through random insertion of dummy system calls into the propagation engine and will address this issue in the future research.

Acknowledgment. The authors are grateful to the Air Force Office of Scientific Research (USA) and the Kazakh National University (Kazakhstan) for supporting this research and to Alex Volynkin and Stephen Mazur of Binghamton University for their valuable suggestions and comments.

References

1. Bernaschi, M., Grabrielli, E., Mancini, L.: Operating System Enhancements to Prevent the Misuse of System Calls. In: Proceedings of the ACM Conference on Computer and Communications Security (June 2000)
2. Bowen, T., Segal, M., Sekar, R.: On preventing intrusions by process behavior monitoring. In: Proceedings of the Workshop on Intrusion Detection and Network Monitoring (May 1999)
3. Durante, A., Di Pietro, R., Mancini, L.V.: Formal Specification for Fast Automatic IDS Training. In: Abdallah, A.E., Ryan, P.Y.A., Schneider, S. (eds.) FASec 2002. LNCS, vol. 2629, pp. 191–204. Springer, Heidelberg (2003)
4. Gottman, J.M., Kumar, R.A.: Sequential analysis. A guide for behavioral researchers. Cambridge University Press, Cambridge (1990)
5. Hofmeyr, S.A., Forrest, S., Somayaji, A.: Intrusion detection using sequences of system calls. Journal of Computer Security 6, 151–180 (1998)
6. Kang, D., Fuller, D., Honavar, V.: Learning classifiers for misuse and anomaly detection using a bag of system calls representation. In: Proceedings of 6th IEEE Systems Man and Cybernetics Information Assurance Workshop (June 2005)
7. Krügel, C., Mutz, K., Valeur, K., Vigna, G.: On the Detection of Anomalous System Call Arguments. In: Snekkenes, E., Gollmann, D. (eds.) ESORICS 2003. LNCS, vol. 2808, Springer, Heidelberg (2003)
8. Liu, A., Cheryl, M.: A Comparison of System Call Feature Representations for Insider Threat Detection. In: Proceedings of the 6th IEEE Information Assurance Workshop (May 2005)
9. Skormin, V., Volynkin, A., Summerville, D., Moronski, J.: Run-Time Detection of Malicious Self-Replication in Binary Executables. Journal of Computer Security 15 (2007)
10. Stolfo, S., Lee, W., Eskin, E.: Modeling system calls for ID with Dynamic Window Sizes. Proceedings of the DISCEX II (June 2001)
11. Tandon, G., Chan, P.K.: Learning Useful System Call Attributes for Anomaly Detection. In: Proceedings of the FLAIRS Conference (June 2005)
12. Tokhtabayev, A.G., Skormin, A.G.: Non-Stationary Markov Models and Anomaly Propagation Analysis in IDS. In: Proceedings of IAS'07 (to appear)
13. Xu, M., Chen, C., Ying, J.: Anomaly detection based on system call classification. Journal of Software 15, 391–403 (2004)

Investigation of the Effectiveness of Alert Correlation Methods in a Policy-Based Security Framework

Bartłomiej Balcerek[1], Piotr Dragan[1], Bogdan Trawiński[2], and Marcin Wojtkiewicz[1]

[1] Wrocław Centre for Networking and Supercomputing, Plac Grunwaldzki 9, 50-377 Wrocław, Poland
[2] Wrocław University of Technology, Institute of Applied Informatics, Wybrzeże S. Wyspiańskiego 27, 50-370 Wrocław, Poland
{bartlomiej.balcerek,piotr.dragan,bogdan.trawinski, marcin.wojtkiewicz}@pwr.wroc.pl

Abstract. The investigation of the effectiveness of alert correlation methods implemented in the Alert Correlation Module was presented in the paper. The module was developed within the POSITIF project (Policy-based Security Tools and Framework) funded by the European Commission. Three effectiveness metrics were applied: reduction coefficient, average ancestor count in alert trees and percentage of meta-alerts in outgoing alerts. Research was conducted using a test environment comprising 14 computers with IDS installed. Network traffic was monitored for 40 days, and during this time the IDSs generated 211 251 alerts. The module correlated and recognized as dangerous 6994 of them therefore achieving a level of reduction equal to 96.69 percent, which could be regarded as a good result.

Keywords: intrusion detection, alert correlation, alert merging.

1 Introduction

Alert correlation is an essential process in a network security system because it recognizes similar alerts and merges them assuring the work of network administrators will be more effective. Several alert correlation approaches were proposed. Cuppens and Miege [1] distinguished three phases in this process: alert clustering, alert merging and alert correlation. IDMEF (Intrusion Detection Message Exchange Format) was used as the format of input data and the attacks were described by means of pre- and post-conditions using the LAMBDA language. Valdes and Skiner [10] proposed a mathematical approach determining minimal values of feature similarities of alerts to be merged. Xu and Ning [12] introduced the notion of the "triggering event" focusing first of all on the causes generating alerts. Our implementation has been inspired by the approach proposed by Valeur et al. [11], and similar methods were described by Debar and Wespi [4]. All of the above mentioned methods are based on the similarity between alert attributes and our approach also belongs to this class. Research reported in the present paper was carried out within the POSITIF (Policy-based Security Tools and Framework) project [5,8] funded by the European Commission.

2 Outline of Alert Correlation Methods

The alert correlation module is composed of a set of procedures which can be arranged in many ways. Some procedures process data of an alert and the others implement correlation methods within individual filters. Each filter is characterized by a time window i.e. maximal time of alert correlation and a reduction flag determining whether the ancestors of alerts being correlated should be deleted when the time window for a given alert expires. All alert correlation algorithms use a time window to determine the maximal time an alert can stay in a queue of a given filter. But not all alerts wait in queues for a whole window length. If a new alert occurs in a queue with a time tag of window length greater than other alerts then the latter is moved to the queue of the next filter. Such a mechanism assures better effectiveness and lower memory consumption, which is especially important during archival file processing. Five types of filters were implemented: Fusion, One2One, Network-Host, One2Many and Many2One. The Fusion filter is a simple and very useful one. Its task is to merge two or more alerts referring to the same event but generated by different IDSs. It should be located at the beginning of the correlation process. The One2One filter associates alerts from the same source and targeted to the same machine. The main task of the Network-Host filter is to correlate alerts generated by the network IDSs with alerts produced by the host IDSs concerning the same network node. In turn One2Many and Many2One filters are designed for detecting repeated alerts which are similar with respect to source or destination address.

3 Investigation of Alert Correlation Effectiveness

3.1 The Architecture of the Testbed

To conduct experiments a test bed was constructed comprising 14 computers. Four of them were unshielded and the other ten created a local area network and were protected from the Internet by means of a firewall. Each of the first four computers was equipped with a Snort functioning as IDS, hepenthes - honeypot, and the Prelude LML log analyzer. This group of computers logged alerts to a Prelude system which could communicate with the correlation module over a BEEP protocol. Other computers operating within the local network had no IDS, but the network traffic from the switch binding them was redirected to a computer with a Snort and second Prelude system, which was also connected to the correlation module. The test bed was connected to the Internet and exposed to actual network traffic. The traffic was monitored for 40 days from January 23th through March 3rd 2007. During this time all IDSs generated a total of 211 251 alerts.

3.2 Metrics of Alert Correlation Effectiveness Employed

During the experiment the following basic parameter values were collected for each filter:

- IAC – Incoming Alert Count,
- OAC – Outgoing Alert Count,
- MC – Merge Count,
- MAC – Meta-Alert Count,
- AC – Ancestor Count.

The relationships between these parameters can be expressed by means of the following formulas:

$$IAC = MC + OAC. \tag{1}$$

$$IAC = MC + MAC. \tag{2}$$

Using the above mentioned parameters three alert correlation metrics were calculated. The first and fundamental one is the Reduction Coefficient (RCF) that can be expressed as a percentage ratio of merge count to incoming alert count using the following formula:

$$RCF = \frac{MC}{IAC} * 100\%. \tag{3}$$

The second metrics is the Average Ancestor Count (AAC) occurring in alert trees. It characterizes the complexity of correlated alerts and can be calculated using the following formula:

$$AAC = \frac{AC}{MAC}. \tag{4}$$

The third metrics is the Meta-Alert Coefficient (MACF) that can be expressed as a percentage ratio of meta-alert count to outgoing alert count using the following formula:

$$MACF = \frac{MAC}{OAC} * 100\%. \tag{5}$$

3.3 Results of the Experiment

In Fig. 1 the cumulative values of effectiveness metrics versus the number of alerts in the first Fusion Filter are shown.

At the beginning of processing, for the first 30 000 alerts both coefficients had values equal 0, because our test bed was being rebuilt during data collection. Within the first 10 days only one IDS (Snort) was installed on the computers and due to filter characteristics alert correlation was not possible. Afterwards the nepenthes was installed, which enabled the correlation to be performed.

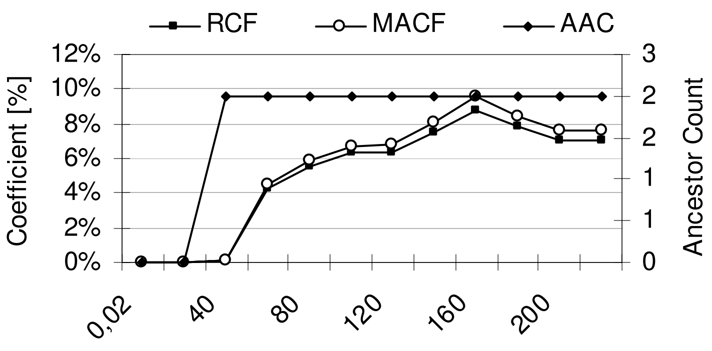


Fig. 1. Effectiveness of the Fusion filter (cumulative)

This caused a growth of coefficient values of about 160 thousand alerts. The average number of ancestors was equal to 2 during the whole process because only two IDSs (Snort and nepenthes) were applied. The reduction and meta-alert coefficients achieved similar values because with a constant number of ancestors the number of merges and meta-alerts is the same and at the relatively low reduction rate the numbers of incoming and outgoing alerts differed slightly. In the One2One filter the reduction coefficient gained high values at all times (see Fig. 2). At the very beginning the meta-alert coefficient achieved high values and simultaneously the average number of ancestors was low. During this time many different meta-alerts composed of no more than a few alerts were created. Afterwards the number of ancestors increased while the percentage of outgoing meta-alerts decreased. The analysis of data collected during the process revealed that there was a great number of brute-force attacks in that period. Within 100 seconds of the One2One filter window the structures were being created where the number of ancestors reached even 350.

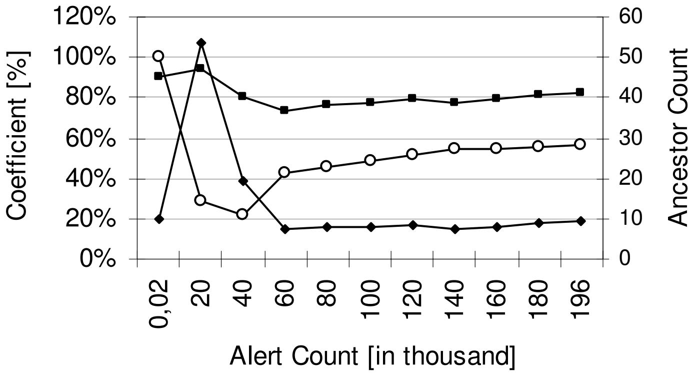


Fig. 2. Effectiveness of the One2One filter (cumulative)

The analysis of the cases detected by the Network-Host filter revealed that the majority of alerts were false despite the low reduction rate. In order to increase the effectiveness it is necessary to apply mechanisms mapping services into ports and verify incoming alerts. It turned out the One2Many and Many2One filters gained a reduction rate lower than we expected. This was caused partly by shorter time windows of 60 and 30 seconds respectively. On the other hand we wanted to protect against generating huge structures in the case of DDoS attack, but no such attack occurred. The remaining 30 082 alerts were finally analyzed with respect to their maliciousness. After reaching the maximum value the reduction rate started to decrease, which can be explained by the fact that the attacks on our network became more and more dangerous day by day. At the end the value of the reduction coefficient dropped to 76.75 percent. The most essential feature of the whole correlation process is the reduction coefficient. The global value of the reduction coefficient is the resultant value of reduction accomplished in all filters. Of 211 251 input alerts the module correlated and recognized 6 994 of them thus achieving the reduction rate of 96.69 percent. This means that the module warned the network administrator on average 174.85 times daily within the analyzed period of 40 days. This is a relatively high value. Table 1 summarizes the results obtained in successive filters of the correlation module.

Table 1. Summary of alert correlation in individual filters

Metrics	Fusion	One-to-One	Network-Host	One-to-Many	Many-to-One	Severity Assessment
RCF	7.07%	80.09%	0.88%	6.68%	7.44%	76.75%
MACF	7.61%	56.08%	0.89%	4.85%	6.25%	
AAC	2.00	9.17	2.00	2.48	2.29	
IAC	211251	196281	35148	34830	32513	30082
OAC	196281	35148	34830	32513	30082	6994

4 Conclusions

The global reduction rate produced by our correlation process equals 96.69 percent and that can be regarded as a relatively good result. However it gives an average number of 175 alerts per day and this figure needs to be improved. In comparison with the results reported in [11] we obtained a reduction coefficient lower than 2.51 percent per honeypot. However the efficiency of our method in terms of reduction rate is much better than those achieved by some archival files (MIT 1998 - 80,87%, MIT1999 - 53,00%, CTV - 93,15%, Rome AFRL - 91,21%) [11]. Of course it should be taken into account that each test file could have different features.

In line with our expectations the best reduction was provided by the One2One filter. The question is what outcome would be obtained if the filters in the module were arranged in another way. However it seems we used the most appropriate sequence of filters in our model.

References

1. Cuppens, F., Miege, A.: Alert Correlation in a Cooperative Intrusion Detection Framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, pp. 202–215. IEEE Computer Society Press, Los Alamitos (2002)
2. Danyliv, R., Meijer, J., Demchenko, Y.: The Incident Object Description Exchange Format (2006), http://www.ietf.org/internet-drafts/draft- ietf-inch-iodef-10.txt
3. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (2006), http://tools.ietf.org/html/draft-ietf-idwg-idmef-xml-15
4. Debar, H., Wespi, A.: Aggregation and Correlation of Intrusion-Detection Alerts. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 85–103. Springer, Heidelberg (2001)
5. Dörges, T., Kossakowski, K.P.: Proactive Security Monitoring in a Policy Managed Network. In: 18th Annual FIRST Conference (2006)
6. Lioy, A.: Positif (Policy-based Security Tools and Framework) - Annex (2003)
7. POSITIF Project Website (2003), http://www.positif.org
8. Prelude-IDS Project Website: http://www.prelude-ids.org
9. Rose, M.: The Blocks Extensible Exchange Protocol Core (2001), http://www.rfc-editor.org/rfc/rfc3080.txt
10. Valdes, A., Skinner, K.: Probabilistic Alert Correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
11. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Transactions On Dependable and Secure Computing 1(3), 146–169 (2004)
12. Xu, D., Ning, P.: Alert Correlation through Triggering Events and Common Resources. In: Yew, P.-C., Xue, J. (eds.) ACSAC 2004. LNCS, vol. 3189, pp. 360–369. Springer, Heidelberg (2004)

Host-Based Intrusion Detection System: Model and Design Features

Pyotr Zegzhda and Semyon Kort

St. Petersburg State Polytechnical University
zeg@ssl.stu.neva.ru, sam@ssl.stu.neva.ru, LNCS@Springer.com

Abstract. This article reports on a model of a host-based intrusion detection system. Using a model of a state machine possible mechanisms of security violations in a computer system are analyzed. Thereafter principles are suggested for building an analysis module based on a model of dynamic monitoring of system statuses. The article concludes with a number of approaches for developing a data acquisition module for a host-based intrusion detection system.

Keywords: intrusion detection, host-based intrusion detection system, attack, API Intercept.

1 Introduction

Talking about host-based IDS we should focus on its two main modules: data collection module and analysis module. Putting together they define features of whole IDS. In the first chapter we discuss analysis module. Nice review of previous works which are devoted to design of the analysis module can be found in [1], [2]. Work [3] is devoted to “practical” and not just not to ”scientific” approach to IDS design. The proposed in this article model of an IDS determines an indefinite-measure multitude of intrusions and a denumerable set of their description models. The set of intrusion description models is used by the evaluation module and determines the likelihood of undesired actions the system is capable of detecting. The multitude of attacks not described by models determines the completeness of detection by the IDS linked to the ability of detecting security violations. Proposed approach seems us promising for design of practical host-based IDS. In the second chapter discuss design principles of design of host-based data collection module. Base works related to building host-based IDS data collection module can be found in [4], [5] for Windows and in [6] for Linux. We systematize offered methods of design and analyze their characteristics.

2 Development of the Analysis Module

The basic approach used for describing the system model for examining its security in the course of engineering is a system status analysis. This article offers the description of an approach to status security analysis aimed at detecting information security violations in the course of computer system operation. The online system status monitoring subsystem should detect system transfers to unsafe statuses due to invalid operations performed by the user in the system, or intrusions into the system.

Hence, the task of online system status security monitoring consists in:

1) detecting the statuses that contradict to the security policy determined in the system i.e. unsafe statuses;
2) identifying the reasons that caused an unsafe status of the system;
3) evaluating the security of the system being intruded.

The approach proposed should take into account security system violations caused by both incompliance with the security policy and attacks on the system resulting in the need to describe two models. However both models need to be described based on unified mathematical formalism which will provide for subsequent merger of the models. The above formalism in this paper presented as finite state machine describing the behavior of a system subject.

2.1 System Model Identifying Security Policy Violations

Let us introduce definitions of the concepts to be used in further reasoning. {S} – a multitude of subjects; {O} – a multitude of objects; {Op} – a multitude of operations; {Prg} – a multitude of services used by the subject (programs or program interfaces). The introduction of this multitude into the system model description is due to the fact that system subject operations over objects are implemented using services. Hence the matrix of subject to object access in the system may be defined as follows: M' (s, prg, o) – access matrix for programs used on behalf of subjects to perform operations with system objects.

Then machine $A = \{\sigma, t, Out, \sigma_0, \delta, \lambda\}$ which represents the user performance with respect to the security policy determined in the system, may be described as follows:

$\sigma = \{op_1(prog_1, 0_1), \ldots, op_i(prog_i, 0_i)\}$ – a machine state describing operations performed by a system subject over objects; the multitude of statuses is partially rank-ordered.

$t \in Op(prg_i, o_j)$ – controlling machine symbols corresponding to the operations performed by the subject over system objects using programs.

A safe status is a status describing operations performed by the subject that do not contradict to the security policy. Thus the status security evaluation describes the machine exit as Out = {Sec, UnSec}. The machine completes its operation if it goes to an unsafe status. Then the transfer function δ may be described as follows:

$\forall\, t = op_i\ \exists\ op_i(prog_i, 0_i) \in \sigma_i = t \rightarrow \sigma_{i+1} = \sigma_i$
$\forall\, t = op_i\ !\exists\ op_i(prog_i, 0_i) \in \sigma_i = t \rightarrow \sigma_{i+1} = \sigma_i \cup t.$

The exit function of machine λ may be presented as follows:

$\forall\, t = op_i\ (op_i(prog_i, 0_i) \in \sigma_i) \vee (op_i(prog_i, 0_i) \in M'(s, prg, o)) \rightarrow Out = Sec$
$\forall\, t = op_i\ (op_i(prog_i, 0_i) \in \sigma_i) \vee (op_i(prog_i, 0_i) \notin M'(s, prg, o)) \rightarrow Out = UnSec$

2.2 Intrusion Detection Model

After describing a model for the check of system functional security for compliance with the security policy let us discuss a model describing likely system attacks. System attacks are identified with the use of attack signatures $\{Sgt_m\}_{m=1}^{M}$. The multitude of signatures describing attacks may be grouped into submultitudes according to their PROPERTY. The PROPERTY reflects the multitude of attack signatures into multitude prp_n $n \in 1: N$ that describes the attack objectives. Each element of multitude prp_n reflects the objectives of an attack involving signatures. Multitude $\{prp_n\}_{n=1}^{N}$ is partially rank-ordered.

It is important that the intruder performing intrusion advances in its actions by means of launching (successfully or not) various attacks on the system. Then the multitude of signatures may be rank-ordered in accordance with the intrusion stage as follows: $\{Sgt_{i_1}\}_{i_1=1}^{M1} ... \{Sgt_{i_k}\}_{i_k=1}^{Mk}$ while $\sum_{j=1}^{K} m_j = M$.

At that, the scenarios of security violation (intrusion) may be described as Scen = $(Sgt_0, Sgt_1, \ldots, Sgt_k)$ $k \leq M$ provided that:

1) $\forall i, j \in 1{:}k, i \neq j \rightarrow Sgt_i \neq Sgt_j$
2) $\forall i, j \in 1{:}k, i \leq j \rightarrow prp(Sgt_i) \leq prp(Sgt_j)$

The machine describing system security violations may be introduced using the following definitions:

$\sigma = \{Sgt_m\}$ – machine statuses described by a signature corresponding to the most advanced intrusion phase reached by the intruder.

$t \in Sgt_i$ – machine control symbols.

$prp(\sigma_i)$ for the current machine status – machine exit

$\sigma_0 \in \sigma$ – initial status in which the subject starts interacting with the system.

Machine transfer function δ may be described as follows:

$\forall\, t = Sgt_j \quad \exists\, Sgt_i \in \sigma_i : prp(Sgt_j) \leq prp(Sgt_i) \rightarrow \sigma_{i+1} = \sigma_i$
$\forall\, t = Sgt_j \quad !\exists\, Sgt_i \in \sigma_i : prp(Sgt_j) \leq prp(Sgt_i), \rightarrow \sigma_{i+1} = \sigma_i \cup Sgt_i$

Exit function λ is described as follows:

$\forall t = Sgt_i \quad Out = prp(\delta(\sigma_i, t)).$

2.3 Unified Model

The components describing the system model may be described using a unified structure. In the above structure machine statuses will be described as follows: $\sigma = \{\{op_1(prog_1, 0_1), \ldots, op_i(prog_i, 0_i)\}, Sgt_m(Scen_l)\}$. The entry of the unified machine is a user-performed operation using a service, or a user-performed system attack using a service. Hence, $t \in (Op(prg_i, o_j) \vee Sgt_m)$ – machine control symbols. Transfer function δ of the unified machine is described as follows:

$\forall\, t = Sgt_j \quad \exists Sgt_i \in \sigma_i : prp(Sgt_j) \leq prp(Sgt_i) \rightarrow \sigma_{i+1} = \sigma_i$

$\forall\ t = Sgt_j\ \ !\exists Sgt_i\ \in\ \sigma_i : prp\ (Sgt_j) \le prp(Sgt_i) \rightarrow \sigma_{i+1} = \sigma_i \cup Sgt_i$
$\forall\ t = op_i\ \ \ \exists op_i(prog_i, 0_i) \in\ \sigma_i = t \rightarrow \sigma_{i+1} = \sigma_i$
$\forall\ t = op_i\ \ !\exists op_i(prog_i, 0_i) \in\ \sigma_i = t \rightarrow \sigma_{i+1} =\ \sigma_i \cup t.$

The machine exit is the unified machine status profile. In accordance with the above definitions the statuses may be: safe, unsafe and attack condition statuses. The exit function of the unified machine λ is described as follows:

$\forall t = Sgt_i\ Out = prp\ (\delta\ (\sigma_i, t))$
$\forall\ t = op_i\ (op_i(prog_i, 0_i) \in\ \sigma_i) \vee (op_i(prog_i, 0_i) \in\ M'\ (s, prg, o)) \rightarrow Out = Sec$
$\forall\ t = op_i\ (op_i(prog_i, 0_i) \in\ \sigma_i) \vee (op_i(prog_i, 0_i) \notin\ M'\ (s, prg, o)) \rightarrow Out = UnSec$

Therefore, this article offers a machine model providing for online monitoring of system status security. Based on online monitoring of system status security various security violations of the computer system may be detected.

3 Features of Development of the Data Acquisition Module

The attacks and intrusions themselves are commonly described as in lower level terms. The bridging of this gap should be facilitated by an adequate data acquisition method with an option to transform the data obtained to higher presentation levels.

Even though for data acquisition in host-based IDS it is possible to use standard tools of operating system audits it is advisable to customize the data acquisition modules due to the fact that standard audit tools frequently acquire information useless for detecting system security violations while vital information is often missing. Thus, to provide for efficacious tracking of intrusions it is necessary to select a system presentation level which would be most appropriate for acquiring the initial data. The level selection will be determined by two contradictory factors – the ease of information acquisition and the unambiguous decision-making process.

Function $Level(M_k) = \{LevMod_m\}$ provides for the return of the level multitude of operation and object descriptions used to describe intrusion model j. Let us assume that a data acquisition module gathers data at level LevData. Let us designate the presentation of system objects and operations at the level of the data acquisition module as $O_{LevData}$, $R_{LevData}$, and at the model level as O_{LevMod}, R_{LevMod}. Thereafter the proposed system will make it possible to describe the basic properties of the IDS determined by the data acquisition module.

1) Validity of the intrusion detection system. The data acquisition module should run at a minimum level of operation and object presentation present in a multitude of models describing attacks $\forall M_j\, LevData \le \min(\min(LevMod(M_j)))$. In the event that the above condition is not met the data acquisition module will not be capable of transferring complete information to the system event analysis module, and the operation of the IDS will be invalid.

2) Compatibility of the modules of the IDS. The data obtained by the data acquisition module of the host-based IDS should be reduced to a single format used by the analysis module of the intrusion detection system $\forall M_j \exists ! F, G : O_{LevMod} = F(O_{LevData})$, $R_{LevMod} = G(R_{LevData})$. The existence of single transformations F and G and their

complexity determine the possibility to identify objects and operations at the level of intrusion model presentation as well as the complexity of development.

3) Compatibility of the IDS with the computer system. Ideally, the intrusion detection system should be transparent for the user.

In terms of completeness of the IDS it is appropriate to use a data acquisition module controlling the behavior of the whole system. However, a module developed after such a pattern may prove inefficient because the volume of data acquired for analysis may be superfluous. The use of data acquired on the events in a particular subsystem may be appropriate from the perspective of detecting subsystem attacks which limits the number of models used and reduces the completeness of the intrusion detection

Table 1. Comparison of Methods Used to Build a Data Acquisition Module for Windows OS Family Host-based IDS

Comparison criteria / Method or tool of data acquisition	Standard audit	Intercept of system calls	OS subsystem drivers	Customized shells	Performance analysis Tools
Efficiency					
High degree of information content	-	+	+	+/-	+/-
Ability to acquire information regarding all processes running on the system	+	+/-	-	+	+
Signature method of attack detection	-	+	+	+	-
Method for anomaly detection	+/-	+	+	+	+
Analysis of object operations in the system	+/-	+	+	+	-
Versatility					
Applicability of a single method for all subsystems	+	+	-	-	-
Ability to use the information acquired for online analysis in the external system	+	+	+	-	-
Compatibility					
Transparency for the user (absence of shell)	+	+	+	-	-
User-free information acquisition	+	+	+	-	-
Does not require any modification of the environment or the software	+	+/-	+/-	-	+
Protectibility					
Difficult to overcome the protection system	-	+/-	+	-	-

system. Thus, a win-win option would provide for tracking the subsystem events that are critical for the system security. In accordance with the requirements the modification of the software environment may impact the compatibility of the data acquisition module with the computer system. As a result, it is recommended to use the intrusion detection system without modifying the software environment or using modificationof environment variables. As shown above, the choice of performance level for the data acquisition module, in its turn, has an impact on the validity of IDS and the compatibility of its modules. At that, it is most appropriate to develop modules for acquisition of the data related to the intercept of system calls and API requests. Table 1 shows a comparison of the most common methods used to build a data acquisition module for Windows OS family host-based IDS.

It should be noted that the values shown in the above table for different comparative features with respect to the method used for building a data acquisition module for host-based IDS, will be typical of not only Windows OS but of other operating systems too. The most promising methods for building a data acquisition module should be the method based on the intercept of system calls (library functions), and the method based on the use of subsystem drivers. The use of methods based on the intercept of system calls is less time-consuming (while the information content is the same) compared to the method based on the use of subsystem drivers. However the use of methods based on the intercept of system calls requires strong efforts aiming at ensuring protectibility compared to the methods based on the use of subsystem drivers. As it follows from Table 1, the methods based on the intercept of system calls may also suffer from problems related to the requirement regarding compatibility.

4 Conclusion

Thus, in this paper we try to find balance between intellectuality and usability of host-based IDS. Our algorithm of detection is effective and expressive and can be used in practical IDS. Proposed model makes it possible to justify the selection of the method for building a data acquisition module of host-based IDS.

References

1. Axelsson, S.: Research in Intrusion-Detection systems: A Survey. Technical Report 98-17, Dept. of Computer Eng. Chalmers Univ. of Tech, SE-412 96 (1998)
2. Axelsson, S.: Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Depart. of Computer Engineering, Chalmers University (2000)
3. Allen, J., Christie, A., Fithen, W., McHugh, J., Pickel, J., Stoner, E.: State of the practice of intrusion detection technologies. Technical Report CMU/SEI-99TR-028, CMU/SEI (2000)
4. Richter, J.: Advanced Windows, 3rd edn. Microsoft Press (1997)
5. Solomon, D., Russinovich, M.: Microsoft Windows Internals, 4 th edn. Microsoft Windows Server(TM) 2003, Windows XP, and Windows 2000. Microsoft Press (2004)
6. Godber, A.: Linux Function Interception (2002) URL http://0xb.org/interception/

Interval Approach to Preserving Privacy in Statistical Databases: Related Challenges and Algorithms of Computational Statistics

Luc Longpré, Gang Xiang, Vladik Kreinovich, and Eric Freudenthal

Department of Computer Science, University of Texas at El Paso
El Paso, TX 79968, USA
vladik@utep.edu

Abstract. In many practical situations, it is important to store large amounts of data and to be able to statistically process the data. A large part of the data is confidential, so while we welcome statistical data processing, we do not want to reveal sensitive individual data. If we allow researchers to ask all kinds of statistical queries, this can lead to violation of people's privacy. A sure-proof way to avoid these privacy violations is to store ranges of values (e.g., between 40 and 50 for age) instead of the actual values. This idea solves the privacy problem, but it leads to a computational challenge: traditional statistical algorithms need exact data, but now we only know data with interval uncertainty. In this paper, we describe new algorithms designed for processing such interval data.

Keywords: privacy, statistical databases, interval uncertainty, computational statistics.

1 Interval Approach to Preserving Privacy in Statistical Databases

Need for statistical databases. In many practical situations, it is very useful to collect large amounts of data.

For example, from the data that we collect during a census, we can extract a lot of information about health, mortality, employment in different regions – for different age ranges, and for people from different genders and of different ethnic groups. By analyzing this statistics, we can reveal troubling spots and allocate (usually limited) resources so that the help goes first to social groups that need it most.

Similarly, by gathering data about people's health in a large medical database, we can extract a lot of useful information on how the geographic location, age, and gender affect a person's health. Thus, we can make measures, which are aimed at improving public health, more focused.

Finally, a large statistical database of purchases can help find out what people are looking for, make shopping easier for customers and at the same time, decrease the stores' expenses related to storing unnecessary items.

Need for privacy. Privacy is an important issue in the statistical analysis of human-related data. For example, to check whether in a certain geographic area, there is a gender-based discrimination, we can use the census data to check, e.g., whether for all people from this area who have the same level of education, there is a correlation between salary and gender. One can think of numerous possible questions of this type related to different sociological, political, medical, economic, and other questions. From this viewpoint, it is desirable to give researchers *ability to perform* whatever *statistical analysis* of this data that is reasonable for their specific research.

On the other hand, we do not want to give them direct access to the raw census data, because a large part of the census data is *confidential.* For example, for most people (those who work in private sector) salary information is confidential. Suppose that a corporation is deciding where to built a new plant and has not yet decided between two possible areas. This corporation would benefit from knowing the average salary of people of needed education level in these two areas, because this information would help them estimate how much it will cost to bring local people on board. However, since salary information is confidential, the company should not be able to know the exact salaries of different potential workers.

The need for privacy is also extremely important for *medical* experiments, where we should be able to make statistical conclusions about, e.g., the efficiency of a new medicine without disclosing any potentially embarrassing details from the individual medical records.

Such databases in which the outside users have cannot access individual records but can solicit statistical information are often called *statistical databases.*

Maintaining privacy is not easy. Maintaining privacy in statistical databases is not easy. Clerks who set up policies on access to statistical databases sometimes erroneously assume that once the records are made anonymous, we have achieved perfect privacy. Alas, the situation is not so easy: even when we keep all the records anonymous, we can still extract confidential information by asking appropriate questions.

Many examples of such extraction can be found in a book by D. Denning [4]. For example, suppose that we are interested in the salary of Dr. X who works for a local company. Dr. X's mailing address can be usually taken from the phone book; from the company's webpage, we can often get his photo and thus find out his race and approximate age. Then, to determine Dr. X's salary, all we need is to ask what is the average salary of all people with a Ph.D. of certain age brackets who live in a small geographical area around his actual home address – if the area is small enough, then Dr. X will be the only person falling under all these categories.

Even if only allow statistical information about salaries $s_1, \ldots, s_q$ when there are at least a certain amount n_0 people within a requested range, we will still be able to reconstruct the exact salaries of all these people. Indeed, for example, we can ask for the number and average salary of all the people who live on Robinson street at houses 1 through 1001, and then we can ask the same question about all the people who live in houses from 1 to 1002. By comparing the two numbers, we get the average salary of the family living at 1002 Robinson – in other words, we gain the private information that we tried to protect.

In general, we can ask for the average $\frac{s_1 + \ldots + s_q}{q}$, and for several moments of salary (variance, third moment, etc): if we know the values v_j at least q different functions $f_j(s_1, \ldots, s_q)$ of s_i, then we can, in general, reconstruct all these values from the corresponding system of q equations with q unknowns: $f_1(s_1 \ldots, s_q) = v_1, \ldots, f_q(s_1, \ldots, s_q) = v_q$.

At first glance, moments are natural and legitimate statistical characteristics, so researchers would be able to request them, but on the other hand, we do not want them to be able to extract the exact up-to-cent salaries of all the folks leaving in a certain geographical area.

What restriction should we impose on possible statistical queries that would guarantee privacy but restrict research in the least possible way?

What is known. These are anecdotal examples of poorly designed privacy and security, but, as we have mentioned, the problem is indeed difficult: Many seemingly well-designed privacy schemes later turn out out to have unexpected privacy and security problem, and it is known that the problem of finding a privacy-preserving scheme is, in general, NP-hard [4].

Different aspects of the problem of privacy in statistical databases, different proposed solution to this problem, and their drawbacks, are described in [2,4,5,6,7,10,11,15,17,18,20,21,22,24] (see also references therein).

Interval approach to privacy protection. A sure-proof way to avoid these privacy violations is to store ranges (intervals) of values instead of the actual values. For example, instead of keeping the exact age, we only record whether the age is between 0 and 10, 10 and 20, 20 and 30, etc.

In this case, no matter what statistics we allow, the worst that can happen is that the corresponding ranges will be disclosed. However, in this situation, we do not disclose the original exact values – since these values are not stored in the database in the first place.

2 Related Challenges and Algorithms of Computational Statistics

Related challenges of computational statistics. This idea of storing intervals solves the privacy problem, but it leads to a computational challenge.

Indeed, suppose that we are interested in the value of a statistical characteristic $C(x_1, \ldots, x_n)$ such as population mean $E = \frac{x_1 + \ldots + x_n}{n}$, (biased) population variance $V = \frac{(x_1 - E)^2 + \ldots + (x_n - E)^2}{n}$, covariance, correlation, etc.

Traditional statistical algorithms for computing these characteristics assume that we know the exact values of the samples x_i, y_i, etc. However, in our case, we do not know these actual values, we only know the *intervals* $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$ of possible values of these characteristics. Since we do not know the actual values x_i, we cannot compute the exact range of the characteristic C, we can only find the *range* of this characteristic:

$$C(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=} \{C(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

So, the challenge is: given the characteristic $C(x_1, \ldots, x_n)$ and the intervals $\mathbf{x}_i$, we must compute the corresponding range.

Important comment: what are the statistical properties of these estimations? What *really* interests the user is not a statistical characteristic like population mean E, but rather the *actual* mean of the actual distribution – of which the database contains only a sample. From this viewpoint, a population mean is interesting because it is a good approximation to the actual mean: when the sample size n increases, then with probability 1 the corresponding statistic $C_n \stackrel{\text{def}}{=} C(x_1, \ldots, x_n)$ converges to the actual value c of the desired characteristics, and the difference $d(C_n, c) \stackrel{\text{def}}{=} |C_n - c|$ tends to 0 fast.

In the case of privacy-related interval uncertainty, for every n, we get an *interval* $\mathbf{C}_n \stackrel{\text{def}}{=} C(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. The quality of this interval approximation can be naturally described by estimating the (Hausdorff) distance $d(c, \mathbf{C}_n) = \min\limits_{C \in \mathbf{C}_n} d(c, C)$ between the actual value c and the interval: e.g., this distance is 0 if and only if the interval $\mathbf{C}_n$ contains the desired value c.

Since each actual (hidden) value x_i belongs to the corresponding interval $\mathbf{x}_i$, we have $C_n = C(x_1, \ldots, x_n) \in \mathbf{C}_n$. The distance $d(c, \mathbf{C}_n)$ is defined as a minimum, hence we have $d(c, \mathbf{C}_n) \le d(c, C_n)$. We can therefore conclude that *the rate of convergence for interval estimates is the same (or better) than for the corresponding point estimates.*

Let us now go back to the computational problem.

The resulting computational problem is known – as interval computations. While privacy-related applications are reasonably novel, the problem of computing the range of a known function $f(x_1, \ldots, x_n)$ under interval uncertainty $x_i \in \mathbf{x}_i$ is a well-known and well-studied problem in applications, known as a problem of *interval computations*; see, e.g., [9] (see also [16]).

Indeed, in many real-life problems, we are interesting in the values of some quantity y which are difficult or impossible to measure directly; example include the amount of oil in a given well or a distance to a star. To estimate the value of this quantity y, we measure the values of easier-to-measure quantities $x_1, \ldots, x_n$ related to y in a known way $y = f(x_1, \ldots, x_n)$, and then use the measured values $\widetilde{x}_i$ of these quantities to estimate y as $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

Measurements are never 100% accurate. As a result, the result $\widetilde{x}$ of the measurement is, in general, different from the (unknown) actual value x of the desired quantity. The difference $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$ between the measured and the actual values is usually called a *measurement error.*

The manufacturers of a measuring device usually provide us with an upper bound Δ for the (absolute value of) possible errors, i.e., with a bound Δ for which we guarantee that $|\Delta x| \le \Delta$. The need for such a bound comes from the very nature of a measurement process: if no such bound is provided, this means that the difference between the (unknown) actual value x and the observed value $\widetilde{x}$ can be as large as possible.

Since the (absolute value of the) measurement error $\Delta x = \tilde{x} - x$ is bounded by the given bound Δ, we can therefore guarantee that the actual (unknown) value of the desired quantity belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.

In many practical situations, we not only know the interval $[-\Delta, \Delta]$ of possible values of the measurement error; we also know the probability of different values Δx within this interval [19].

In practice, we can determine the desired probabilities of different values of Δx by comparing the results of measuring with this instrument with the results of measuring the same quantity by a standard (much more accurate) measuring instrument. Since the standard measuring instrument is much more accurate than the one use, the difference between these two measurement results is practically equal to the measurement error; thus, the empirical distribution of this difference is close to the desired probability distribution for measurement error.

There are two cases, however, when this determination is not done:

– First is the case of cutting-edge measurements, e.g., measurements in fundamental science. When a Hubble telescope detects the light from a distant galaxy, there is no "standard" (much more accurate) telescope floating nearby that we can use to calibrate the Hubble: the Hubble telescope is the best we have.
– The second case is the case of measurements on the shop floor. In this case, in principle, every sensor can be thoroughly calibrated, but sensor calibration is so costly – usually costing ten times more than the sensor itself – that manufacturers rarely do it.

In both cases, we have no information about the probabilities of Δx; the only information we have is the upper bound on the measurement error.

In this case, after performing a measurement and getting a measurement result $\widetilde{x}_i$, the only information that we have about the actual value x_i of the measured quantity is that it belongs to the interval $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. In this situation, for each i, we know the interval $\mathbf{x}_i$ of possible values of x_i, and we need to find the range $\mathbf{y}$ of the function $f(x_1, \ldots, x_n)$ over all possible tuples $x_i \in \mathbf{x}_i$.

Interval computations are sometimes easy. In some cases, it is easy to estimate the desired range. For example, the arithmetic average E is a monotonically increasing function of each of its n variables $x_1, \ldots, x_n$, so its smallest possible value $\underline{E}$ is attained when each value x_i is the smallest possible ($x_i = \underline{x}_i$) and its largest possible value is attained when $x_i = \overline{x}_i$ for all i. In other words, the range

$\mathbf{E}$ of E is equal to $[E(\underline{x}_1, \ldots, x_n), E(\overline{x}_1, \ldots, \overline{x}_n)]$, where, $\underline{E} = \frac{1}{n} \cdot (\underline{x}_1 + \ldots + \underline{x}_n)$ and $\overline{E} = \frac{1}{n} \cdot (\overline{x}_1 + \ldots + \overline{x}_n)$.

Interval computations are, in general, computationally difficult. For more complex functions $C(x_1, \ldots, x_n)$, the problem of computing the range is often more computationally difficult.

For example, it is known that the problem of computing the exact range $\mathbf{V} = [\underline{V}, \overline{V}]$ for the variance V over interval data $x_i \in [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$ is, in general, NP-hard; see, e.g., [8,12,13]. Specifically, there is a polynomial-time algorithm for computing $\underline{V}$, but computing $\overline{V}$ is, in general, NP-hard [8].

Efficient algorithms exist for several practically useful situations. In many practical situations, there are efficient algorithms for computing $\overline{V}$; see, e.g., [12,13].

For example, an $O(n \cdot \log(n))$ time algorithm exists when no two narrowed intervals $[x_i^-, x_i^+]$, where $x_i^- \stackrel{\text{def}}{=} \widetilde{x}_i - \frac{\Delta_i}{n}$ and $x_i^+ \stackrel{\text{def}}{=} \widetilde{x}_i + \frac{\Delta_i}{n}$, are proper subsets of one another, i.e., when $[x_i^-, x_i^+] \not\subseteq (x_j^-, x_j^+)$ for all i and j [3].

Computational problem are usually easier in the privacy case. In the privacy case, intervals correspond to the fixed subdivision of the real line, For such situations, efficient algorithms exist for computing most statistical characteristics; see, e.g., [12,13].

Important comment: there is a strong need to implement interval-related algorithms in Oracle and SQL. In [12,13]:

– we have *theoretically* proven that our new algorithms produce correct results in *reasonable* time (usually linear or quadratic), and
– we have shown, by implementing these algorithms in standard programming languages, that the corresponding computation time is also *practically* reasonable.

It is worth mentioning that in most real-world applications of statistical databases, practitioners do not write new code in high-level programming languages, they use systems like Oracle and SQL. So, to promote the use of interval methods, it is important to implement our algorithms in systems such Oracle and SQL.

We hope that the privacy-enhancing character of interval-related algorithms and their efficiency will inspire database designers to incorporate such algorithms in the future versions of database management systems.

3 New Problem: Hierarchical Statistical Analysis Under Privacy-Related Interval Uncertainty

Need for hierarchical statistical analysis. In the above description, we assumed that we have all the data in one large database, and we process this large statistical database to estimate the desired statistical characteristics.

In reality, the data is often stored hierarchically. For example, it makes sense to store the census results by states, get averages and standard deviations per state, and then combine these characteristics to get nation-wide statisticss; see, e.g., [1].

Formulas behind hierarchical statistical analysis. Let the data values $x_1 \ldots, x_n$ be divided into $m < n$ groups $I_1, \ldots, I_m$. For each group j, we know the frequency p_j of this group (i.e., the number n_j of elements of this group divided by the overall number of records), the average E_j over this group, and the population variance V_j within j-th group.

One can show that in this case, $E = \sum\limits_{j=1}^{m} p_j \cdot E_j$ and $V = V_E + V_\sigma$, where $V_E = \sum\limits_{j=1}^{m} p_j \cdot E_j^2 - E^2$ and $V_\sigma = \sum\limits_{j=1}^{m} p_j \cdot V_j$.

Hierarchical case: situation with interval uncertainty. When we start with values x_i which are only known with interval uncertainty, we end up knowing E_j and V_j also with interval uncertainty. In other words, we only know the intervals $\mathbf{E}_j = [\underline{E}_j, \overline{E}_j]$ and $[\underline{V}_j, \overline{V}_j]$ that contain the actual (unknown) values of E_j and V_j. In such situations, we must find the ranges of the possible values for the population mean E and for the population variance V; see, e.g., [14].

Analysis of the interval problem. The formula that describes the dependence of E on E_j is monotonic in E_j. Thus, we get an explicit formula for the range $[\underline{E}, \overline{E}]$ of the population mean E: $\underline{E} = \sum\limits_{j-1}^{m} p_j \cdot \underline{E}_j$ and $\overline{E} = \sum\limits_{j-1}^{m} p_j \cdot \overline{E}_j$.

Since the terms V_E and V_σ in the expression for V depend on different variables, the range $[\underline{V}, \overline{V}]$ of the population variance V is equal to the sum of the ranges $[\underline{V}_E, \overline{V}_E]$ and $[\underline{V}_\sigma, \overline{V}_\sigma]$ of the corresponding terms: $\underline{V} = \underline{V}_E + \underline{V}_\sigma$ and $\overline{V} = \overline{V}_E + \overline{V}_\sigma$. Due to similar monotonicity, we can find an explicit expression for the range $[\underline{V}_\sigma, \overline{V}_\sigma]$ for V_σ: $\underline{V}_\sigma = \sum\limits_{j=1}^{m} p_j \cdot \underline{V}_j$ and $\overline{V}_\sigma = \sum\limits_{j=1}^{m} p_j \cdot \overline{V}_j$. Thus, to find the range of the population variance V, it is sufficient to find the range of the term V_E. So, we arrive at the following problem:

4 Formulation of the Problem in Precise Terms and Main Result

GIVEN: an integer $m \geq 1$, m numbers $p_j > 0$ for which $\sum\limits_{j=1}^{m} p_j = 1$, and m intervals $\mathbf{E}_j = [\underline{E}_j, \overline{E}_j]$.

COMPUTE the range $\mathbf{V}_E = \{V_E(E_1, \ldots, E_m) \mid E_1 \in \mathbf{E}_1, \ldots, E_m \in \mathbf{E}_m\}$, where

$$V_E \stackrel{\text{def}}{=} \sum_{j=1}^{m} p_j \cdot E_j^2 - E^2; \quad E \stackrel{\text{def}}{=} \sum_{j=1}^{m} p_j \cdot E_j.$$

Main result. Since the function V_E is convex, we can compute its minimum $\underline{V}_E$ on the box $\mathbf{E}_1 \times \ldots \times \mathbf{E}_m$ by using known polynomial-time algorithms for minimizing convex functions over interval domains; see, e.g., [23].

For computing maximum $\overline{V}_E$, even the particular case when all the values p_j are equal $p_1 = \ldots = p_m = 1/m$, is known to be NP-hard; see, e.g., [8]. Thus, the more general problem of computing $\overline{V}_E$ is also NP-hard. Let us show that in a reasonable class of cases, there exists a feasible algorithm for computing $\overline{V}_E$.

For each interval $\mathbf{E}_j$, let us denote its midpoint by $\widetilde{E}_j \stackrel{\text{def}}{=} \dfrac{\underline{E}_j + \overline{E}_j}{2}$, and its half-width by $\Delta_j \stackrel{\text{def}}{=} \dfrac{\overline{E}_j - \underline{E}_j}{2}$. In these terms, the j-th interval $\mathbf{E}_j$ takes the form $[\widetilde{E}_j - \Delta_j, \widetilde{E}_j + \Delta_j]$.

In this paper, we consider narrowed intervals $[E_j^-, E_j^+]$, where $E_j^- \stackrel{\text{def}}{=} \widetilde{E}_j - p_j \cdot \Delta_j$ and $E_j^+ \stackrel{\text{def}}{=} \widetilde{E}_j + p_j \cdot \Delta_j$. We show that there exists an efficient $O(m \cdot \log(m))$ algorithm for computing $\overline{V}_E$ for the case when no two narrowed intervals are proper subsets of each other, i.e., when $[E_j^-, E_j^+] \not\subseteq (E_k^-, E_k^+)$ for all j and k.

Algorithm.

- First, we sort the midpoints $\widetilde{E}_1, \ldots, \widetilde{E}_m$ into an increasing sequence. Without losing generality, we can assume that $\widetilde{E}_1 \le \widetilde{E}_2 \le \ldots \le \widetilde{E}_m$.
- Then, for every k from 0 to m, we compute the value $V_E^{(k)} = M^{(k)} - (E^{(k)})^2$ of the quantity V_E for the vector $E^{(k)} = (\underline{E}_1, \ldots, \underline{E}_k, \overline{E}_{k+1}, \ldots, \overline{E}_m)$.
- Finally, we compute $\overline{V}_E$ as the largest of $m+1$ values $V_E^{(0)}, \ldots, V_E^{(m)}$.

To compute the values $V_E^{(k)}$, first, we explicitly compute $M^{(0)}$, $E^{(0)}$, and $V_E^{(0)} = M^{(0)} - E^{(0)}$. Once we computed the values $M^{(k)}$ and $E^{(k)}$, we can compute

$$M^{(k+1)} = M^{(k)} + p_{k+1} \cdot (\underline{E}_{k+1})^2 - p_{k+1} \cdot (\overline{E}_{k+1})^2 \text{ and}$$

$$E^{(k+1)} = E^{(k)} + p_{k+1} \cdot \underline{E}_{k+1} - p_{k+1} \cdot \overline{E}_{k+1}.$$

5 Proof

Number of computation steps.

- It is well known that sorting requires $O(m \cdot \log(m))$ steps.
- Computing the initial values $M^{(0)}$, $E^{(0)}$, and $V_E^{(0)}$ requires linear time $O(m)$.
- For each k from 0 to $m-1$, we need a constant number $O(1)$ of steps to compute the next values $M^{(k+1)}$, $E^{(k+1)}$, and $V_E^{(k+1)}$.
- Finally, finding the largest of $m+1$ values $V_E^{(k)}$ also requires $O(m)$ steps.

Thus, overall, we need

$$O(m \cdot \log(m)) + O(m) + m \cdot O(1) + O(m) = O(m \cdot \log(m)) \text{ steps.}$$

Proof of correctness. The function V_E is convex. Thus, its maximum $\overline{V}_E$ on the box $\mathbf{E}_1 \times \ldots \times \mathbf{E}_m$ is attained at one of the vertices of this box, i.e., at a vector $(E_1, \ldots, E_m)$ in which each value E_j is equal to either $\underline{E}_j$ or to $\overline{E}_j$.

To justify our algorithm, we need to prove that this maximum is attained at one of the vectors $E^{(k)}$ in which all the lower bounds $\underline{E}_j$ precede all the upper bounds $\overline{E}_j$. We will prove this by reduction to a contradiction. Indeed, let us assume that the maximum is attained at a vector in which one of the lower bounds follows one of the upper bounds. In each such vector, let i be the largest upper bound index followed by the lower bound; then, in the optimal vector $(E_1, \ldots, E_m)$, we have $E_i = \overline{E}_i$ and $E_{i+1} = \underline{E}_{i+1}$.

Since the maximum is attained for $E_i = \overline{E}_i$, replacing it with $\underline{E}_i = \overline{E}_i - 2\Delta_i$ will either decrease the value of V_E or keep it unchanged. Let us describe how V_E changes under this replacement. Since V_E is defined in terms of M and E, let us first describe how E and M change under this replacement. In the sum for M, we replace $(\overline{E}_i)^2$ with

$$(\underline{E}_i)^2 = (\overline{E}_i - 2\Delta_i)^2 = (\overline{E}_i)^2 - 4 \cdot \Delta_i \cdot \overline{E}_i + 4 \cdot \Delta_i^2.$$

Thus, the value M changes into $M + \Delta_i M$, where

$$\Delta_i M = -4 \cdot p_i \cdot \Delta_i \cdot \overline{E}_i + 4 \cdot p_i \cdot \Delta_i^2.$$

The population mean E changes into $E + \Delta_i E$, where $\Delta_i E = -2 \cdot p_i \cdot \Delta_i$. Thus, the value E^2 changes into $(E + \Delta_i E)^2 = E^2 + \Delta_i(E^2)$, where

$$\Delta_i(E^2) = 2 \cdot E \cdot \Delta_i E + (\Delta_i E)^2 = -4 \cdot p_i \cdot E \cdot \Delta_i + 4 \cdot p_i^2 \cdot \Delta_i^2.$$

So, the variance V changes into $V + \Delta_i V$, where

$$\Delta_i V = \Delta_i M - \Delta_i(E^2) = -4 \cdot p_i \cdot \Delta_i \cdot \overline{E}_i + 4 \cdot p_i \cdot \Delta_i^2 + 4 \cdot p_i \cdot E \cdot \Delta_i - 4 \cdot p_i^2 \cdot \Delta_i^2 =$$

$$4 \cdot p_i \cdot \Delta_i \cdot (-\overline{E}_i + \Delta_i + E - p_i \cdot \Delta_i).$$

By definition, $\overline{E}_i = \widetilde{E}_i + \Delta_i$, hence $-\overline{E}_i + \Delta_i = -\widetilde{E}_i$. Thus, we conclude that $\Delta_i V = 4 \cdot p_i \cdot \Delta_i \cdot (-\widetilde{E}_i + E - p_i \cdot \Delta_i)$. So, the fact that $\Delta_i V \le 0$ means that $E \le \widetilde{E}_i + p_i \cdot \Delta_i = E_i^+$.

Similarly, since the maximum of V_E is attained for $E_{i+1} = \underline{E}_{i+1}$, replacing it with $\overline{E}_{i+1} = \underline{E}_{i+1} + 2\Delta_{i+1}$ will either decrease the value of V_E or keep it unchanged. In the sum for M, we replace $(\underline{E}_{i+1})^2$ with

$$(\overline{E}_{i+1})^2 = (\underline{E}_{i+1} + 2\Delta_{i+1})^2 = (\underline{E}_{i+1})^2 + 4 \cdot \Delta_{i+1} \cdot \underline{E}_{i+1} + 4 \cdot \Delta_{i+1}^2.$$

Thus, the value M changes into $M + \Delta_{i+1} M$, where

$$\Delta_{i+1} M = 4 \cdot p_{i+1} \cdot \Delta_{i+1} \cdot \underline{E}_{i+1} + 4 \cdot p_{i+1} \cdot \Delta_{i+1}^2.$$

The population mean E changes into $E + \Delta_{i+1} E$, where $\Delta_{i+1} E = 2 \cdot p_{i+1} \cdot \Delta_{i+1}$. Thus, the value E^2 changes into $E^2 + \Delta_{i+1}(E^2)$, where

$$\Delta_{i+1}(E^2) = 2 \cdot E \cdot \Delta_{i+1} E + (\Delta_{i+1} E)^2 = 4 \cdot p_{i+1} \cdot E \cdot \Delta_{i+1} + 4 \cdot p_{i+1}^2 \cdot \Delta_{i+1}^2.$$

So, the term V_E changes into $V_E + \varDelta_{i+1}V$, where

$$\varDelta_{i+1}V = \varDelta_{i+1}M - \varDelta_{i+1}(E^2) =$$
$$4 \cdot p_{i+1} \cdot \varDelta_{i+1} \cdot \underline{E}_{i+1} + 4 \cdot p_{i+1} \cdot \varDelta_{i+1}^2 - 4 \cdot p_{i+1} \cdot E \cdot \varDelta_{i+1} - 4 \cdot p_{i+1}^2 \cdot \varDelta_{i+1}^2 =$$
$$4 \cdot p_{i+1} \cdot \varDelta_{i+1} \cdot (\underline{E}_{i+1} + \varDelta_{i+1} - E - p_{i+1} \cdot \varDelta_{i+1}).$$

By definition, $\underline{E}_{i+1} = \widetilde{E}_{i+1} - \varDelta_{i+1}$, hence $\underline{E}_{i+1} + \varDelta_{i+1} = \widetilde{E}_{i+1}$. Thus, we conclude that

$$\varDelta_{i+1}V = 4 \cdot p_{i+1} \cdot (\widetilde{E}_{i+1} - E - p_{i+1} \cdot \varDelta_{i+1}).$$

Since V_E attains maximum at $(E_1, \ldots, E_i, E_{i+1}, \ldots, E_m)$, we have $\varDelta_{i+1}V \le 0$, hence $E \ge \widetilde{E}_{i+1} - p_{i+1} \cdot \varDelta_{i+1} = E_{i+1}^-$.

We can also change both E_i and E_{i+1} at the same time. In this case, from the fact that V_E attains maximum, we conclude that $\varDelta V_E \le 0$.

Here, the change $\varDelta M$ in M is simply the sum of the changes coming from E_i and E_{i+1}: $\varDelta M = \varDelta_i M + \varDelta_{i+1}M$, and the change in E is also the sum of the corresponding changes: $\varDelta E = \varDelta_i E + \varDelta_{i+1}E$. So, for $\varDelta V = \varDelta M - \varDelta(E^2)$, we get

$$\varDelta V = \varDelta_i M + \varDelta_{i+1}M - 2 \cdot E \cdot \varDelta_i E - 2 \cdot E \cdot \varDelta_{i+1}E - (\varDelta_i E)^2 - (\varDelta_{i+1}E)^2 -$$
$$2 \cdot \varDelta_i E \cdot \varDelta_{i+1}E.$$

Hence,

$$\varDelta V = (\varDelta_i M - 2 \cdot E_i \cdot \varDelta_i E - (\varDelta_i E)^2) + (\varDelta_{i+1}M - 2 \cdot E_{i+1} \cdot \varDelta_{i+1}E - (\varDelta_{i+1}E)^2) -$$
$$2 \cdot \varDelta E_i \cdot \varDelta E_{i+1},$$

i.e., $\varDelta V = \varDelta_i V + \varDelta_{i+1}V - 2 \cdot \varDelta_i E \cdot \varDelta_{i+1}E$.

We already have expressions for $\varDelta_i V$, $\varDelta_{i+1}V$, $\varDelta_i E$, and $\varDelta_{i+1}E$, and we already know that $E_{i+1}^- \le E \le E_i^+$. Thus, we have $D(E) \le 0$ for some $E \in [E_{i+1}^-, E_i^+]$, where

$$D(E) \stackrel{\text{def}}{=} 4 \cdot p_i \cdot \varDelta_i \cdot (-E_i^+ + E) + 4 \cdot p_{i+1} \cdot \varDelta_{i+1} \cdot (E_{i+1}^- - E) + 8 \cdot p_i \cdot \varDelta_i \cdot p_{i+1} \cdot \varDelta_{i+1}.$$

Since the narrowed intervals are not subsets of each other, we can sort them in lexicographic order; in which order, midpoints are sorted, left endpoints are sorted, and right endpoints are sorted, hence $E_i^- \le E_{i+1}^-$ and $E_i^+ \le E_{i+1}^+$.

For $E = E_{i+1}^-$, we get

$$D(E_{i+1}^-) = 4 \cdot p_i \cdot \varDelta_i \cdot (-E_i^+ + E_{i+1}^-) + 8 \cdot p_i \cdot \varDelta_i \cdot p_{i+1} \cdot \varDelta_{i+1} =$$
$$4 \cdot p_i \cdot \varDelta_i \cdot (-E_i^+ + E_{i+1}^- + 2 \cdot p_{i+1} \cdot \varDelta_{i+1}).$$

By definition, $E_{i+1}^- = E_{i+1} - p_{i+1} \cdot \varDelta_{i+1}$, hence $E_{i+1}^- + 2 \cdot p_{i+1} \cdot \varDelta_{i+1} = E_{i+1}^+$, and $D(E_{i+1}^-) = 4 \cdot p_i \cdot \varDelta_i \cdot (E_{i+1}^+ - E_i^+) \ge 0$. Similarly,

$$D(E_i^+) = 4 \cdot p_{i+1} \cdot \varDelta_{i+1} \cdot (E_{i+1}^- - E_i^+) \ge 0.$$

The only possibility for both values to be 0 is when interval coincide; in this case, we can easily swap them. In all other cases, all intermediate values $D(E)$ are positive, which contradicts to our conclusion that $D(E) \le 0$. The statement is proven.

6 Auxiliary Result: What If the Frequencies Are Also Only Known with Interval Uncertainty?

Reminder: hierarchical statistical data processing. If we know the frequency of the group j, the mean E_j of the group j, and its second moment $M_j = V_j + E_j^2 = \frac{1}{p_j \cdot n} \cdot \sum_{i \in I_j} x_i^2$, then we can compute the overall mean E and the overall variance as $E = \sum_{j=1}^m p_j \cdot E_j$ and $V = \sum_{j=1}^m p_j \cdot M_j - E^2$.

Reminder: hierarchical statistical data processing under interval uncertainty. In the above text, we considered the case when the statistical characteristics E_j and V_j corresponding to different groups are known with interval uncertainty, but the frequencies p_j are known exactly.

New situation. In practice, the frequencies p_j may also only be known with interval uncertainty. This may happen, e.g., if instead of the full census we extrapolate data – or if we have a full census and try to take into account that no matter how thorough the census, a certain portion of the population will be missed.

In practice, the values x_i (age, salary, etc.) are usually non-negative. In this case, $E_j \geq 0$. In this section, we will only consider this non-negative case. Thus, we arrive at the new formulation of the problem:

GIVEN: an integer $m \geq 1$, and for every j from 1 to m, intervals $[\underline{p}_j, \overline{p}_j]$, $[\underline{E}_j, \overline{E}_j]$, and $[\underline{M}_j, \overline{M}_j]$ for which $\underline{p}_j \geq 0$, $\underline{E}_j \geq 0$, and $\underline{M}_j \geq 0$.

COMPUTE the range $\mathbf{E} = [\underline{E}, \overline{E}]$ of $E = \sum_{j=1}^m p_j \cdot E_j$ and the range $\mathbf{M} = [\underline{M}, \overline{M}]$ of $M = \sum_{j=1}^m p_j \cdot M_j - E^2$ under the conditions that $p_j \in [\underline{p}_j, \overline{p}_j]$, $E_j \in [\underline{E}_j, \overline{E}_j]$, $M_j \in [\underline{M}_j, \overline{M}_j]$, and $\sum_{j=1}^m p_j = 1$.

Derivation of an algorithm for computing $\underline{E}$. When the frequencies p_j are known, we can easily compute the bounds for E. In the case when p_j are also known with interval uncertainty, it is no longer easy to compute these bounds.

Since E monotonically depends on E_j, the smallest value $\underline{E}$ of E is attained when $E_j = \underline{E}_j$ for all j, so the only problem is to find the corresponding probabilities p_j. Suppose that $p_1, \ldots, p_n$ are minimizing probabilities, and for two indices j and k, we change p_j to $p_j + \Delta p$ (for some small Δp) and p_k to $p_k - \Delta p$. In this manner, the condition $\sum_{j=1}^m p_j$ is preserved. After this change, E changes to $E + \Delta E$, where $\Delta E = \Delta p \cdot (\underline{E}_j - \underline{E}_k)$.

Since we start with the values at which E attains its minimum, we must have $\Delta E \geq 0$ for all Δp. If both p_j and p_k are strictly inside the corresponding intervals, then we can have Δp of all signs hence we should have $\underline{E}_j = \underline{E}_k$. So, excluding this degenerate case, we should have at most one value p_i strictly inside – others are at one of the endpoints.

If $p_j = \underline{p}_j$ and $p_k = \overline{p}_k$, then we can have $\Delta p > 0$, so $\Delta E \geq 0$ implies $\underline{E}_j \geq \underline{E}_k$. So, the values $\underline{E}_j$ for all j for which $p_j = \underline{p}_j$ should be $\leq$ than all the values $\underline{E}_k$ for which $p_k = \overline{p}_k$. This conclusion can be reformulated as follows: if we sort the groups in the increasing order of $\underline{E}_j$, we should get first $\overline{p}_j$ then all $\underline{p}_k$. Thus, it is sufficient to consider only such arrangements of probabilities for which we have $\overline{p}_1, \ldots \overline{p}_{l_0-1}, p_{l_o}, \underline{p}_{l_0+1}, \ldots \underline{p}_m$. The value l_0 can be uniquely determined from the condition that $\sum\limits_{j=1}^{m} p_j = 1$. Thus, we arrive at the following algorithm:

Algorithm for computing $\underline{E}$. To compute $\underline{E}$, we first sort the values $\underline{E}_j$ in increasing order. Let us assume that the groups are already sorted in this order, i.e., that

$$\underline{E}_1 \leq \underline{E}_2 \leq \ldots \leq \underline{E}_m.$$

For every l from 0 to k, we then compute

$$P_l = \overline{p}_1 + \ldots + \overline{p}_l + \underline{p}_{l+1} + \ldots + \underline{p}_n$$

as follows: we explicitly compute the sum P_0, and then consequently compute P_{l+1} as $P_l + (\overline{p}_{l+1} - \underline{p}_{l+1})$. This sequence is increasing. Then, we find the value l_0 for which $P_{l_0} \leq 1 \leq P_{l_0+1}$, and take

$$\underline{E} = \sum_{j=1}^{l_0-1} \overline{p}_j \cdot \underline{E}_j + p_{l_0} \cdot \underline{E}_{l_0} + \sum_{j=l_0+1}^{m} \underline{p}_j \cdot \underline{E}_j,$$

where $p_{l_0} = 1 - \sum\limits_{j=1}^{l_0-1} \overline{p}_j - \sum\limits_{j=l_0+1}^{m} \underline{p}_j$.

Computation time. We need $O(m \cdot \log(m))$ time to sort, $O(m)$ time to compute P_0, $O(m)$ time to compute all P_l and hence, to find l_0, and $O(m)$ time to compute $\underline{E}$ – to the total of $O(m \cdot \log(m))$.

Algorithm for computing $\overline{E}$. Similarly, we can compute $\overline{E}$ in time $O(m \cdot \log(m))$. We first sort the values $\overline{E}_j$ in increasing order. Let us assume that the groups are already sorted in this order, i.e., that

$$\overline{E}_1 \leq \overline{E}_2 \leq \ldots \leq \overline{E}_m.$$

For every l from 0 to k, we then compute

$$P_l = \underline{p}_1 + \ldots + \underline{p}_l + \overline{p}_{l+1} + \ldots + \overline{p}_n$$

as follows: we explicitly compute the sum P_0, and then consequently compute P_{l+1} as $P_l - (\overline{p}_{l+1} - \underline{p}_{l+1})$. This sequence is decreasing. Then, we find the value l_0 for which $P_{l_0} \geq 1 \geq P_{l_0+1}$, and take

$$\overline{E} = \sum_{j=1}^{l_0-1} \underline{p}_j \cdot \overline{E}_j + p_{l_0} \cdot \overline{E}_{l_0} + \sum_{j=l_0+1}^{m} \overline{p}_j \cdot \overline{E}_j,$$

where $p_{l_0} = 1 - \sum\limits_{j=1}^{l_0-1} \underline{p}_j - \sum\limits_{j=l_0+1}^{m} \overline{p}_j$.

Derivation of an algorithm for computing $\underline{M}$. First, we notice that the minimum is attained when M_j are the smallest ($M_j = \underline{M}_j$) and E_j are the largest ($E_j = \overline{E}_j$). So, the only problem is to find the optimal values of p_j.

Similarly to the case of $\underline{E}$, we add Δp to p_j and subtract Δp from p_k. Since we started with the values at which the minimum is attained we must have $\Delta M \leq 0$, i.e., $\Delta \cdot [\underline{M}_j - \underline{M}_k - 2E \cdot (\overline{E}_j - \overline{E}_k)] \leq 0$. So, at most one value p_j is strictly inside, and if $p_j = \underline{p}_j$ and $p_k = \overline{p}_k$, we must have $\underline{M}_j - \underline{M}_k - 2E \cdot (\overline{E}_j - \overline{E}_k) \leq 0$, i.e., $\underline{M}_j - 2E \cdot \overline{E}_j \leq \underline{M}_k - 2E \cdot \overline{E}_j$.

Once we know E, we can similarly sort and get the optimal p_j. The problem is that we do not know the value E, and for different values E, we have different orders. The solution to this problem comes from the fact that the above inequality is equivalent to comparing $2E$ with the ratio $\dfrac{\underline{M}_j - \underline{M}_k}{\overline{E}_j - \overline{E}_k}$. Thus, if we compute all n^2 such ratios, sort them, then within each zone between the consequent values, the sorting will be the same. Thus, we arrive at the following algorithm.

Algorithm for computing $\underline{M}$. To compute $\underline{M}$, we first compute all the ratios $\dfrac{\underline{M}_j - \underline{M}_k}{\overline{E}_j - \overline{E}_k}$, sort them, and take Es between two consecutive values in this sorting.

For each such E, we sort the groups according to the value of the expression $\underline{M}_j - 2E \cdot \overline{E}_j$. In this sorting, we select the values $p_j = (\overline{p}_1, \ldots, \overline{p}_{l_0-1}, p_{l_0}, \underline{p}_{l_0+1}, \ldots, \underline{p}_m)$ and pick l_0 in the same manner as when we computed $\underline{E}$. For the resulting p_j, we then compute $\underline{M} = \sum\limits_{j=1}^{m} p_j \cdot \underline{M}_j - \left(\sum\limits_{j=1}^{m} p_j \cdot \overline{E}_j \right)^2$.

Computation time. We need $O(m \cdot \log(m))$ steps for each of m^2 zones, to the (still polynomial) total time $O(m^3 \cdot \log(m)))$.

Algorithm for computing $\overline{M}$. A similar polynomial-time algorithm can be used to compute $\overline{M}$. We first compute all the ratios $\dfrac{\overline{M}_j - \overline{M}_k}{\underline{E}_j - \underline{E}_k}$, sort them, and take Es between two consecutive values in this sorting.

For each such E, we sort the groups according to the value of the expression $\overline{M}_j - 2E \cdot \underline{E}_j$. In this sorting, we select the values

$$p_j = (\underline{p}_1, \ldots, \underline{p}_{l_0-1}, p_{l_0}, \overline{p}_{l_0+1}, \ldots, \overline{p}_m)$$

and pick l_0 in the same manner as when we computed $\overline{E}$. For the resulting p_j, we then compute

$$\overline{M} = \sum_{j=1}^{m} p_j \cdot \overline{M}_j - \left(\sum_{j=1}^{m} p_j \cdot \underline{E}_j\right)^2.$$

7 Conclusion

In medicine, in social studies, etc., it is important to perform statistical data analysis. By performing such an analysis, we can find, e.g., the correlation between the age and income, between the gender and side effects of a medicine, etc. People are often willing to supply the needed confidential data for research purposes. However, many of them are worried that it may be possible to extract their confidential data from the results of statistical data processing – and indeed such privacy violations have occurred in the past.

One way to prevent such privacy violations is to replace the original confidential values with ranges. For example, we divide the set of all possible ages into ranges $[0, 10]$, $[10, 20]$, $[20, 30]$, etc. Then, instead of storing the actual age of 26, we only store the range $[20, 30]$ which contains the actual age value.

This approach successfully protects privacy, but it leads to a computational challenge. For example, if we want to estimate the variance, we can no longer simply compute the statistic such as population variance

$$V = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n} \cdot \sum_{i=1}^{n} x_i\right)^2;$$

since we only know the intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of x_i, we can only compute the range $\mathbf{V}$ of possible values of this statistic when $x_i \in \mathbf{x}_i$. In our previous papers, we designed algorithms efficiently computing this range based on the intervals $\mathbf{x}_i$.

In many real-life situations, several research groups independently perform statistical analysis of different data sets. The more data we use for statistical analysis, the better the estimates. So, it is desirable to get estimates based on the data from all the data sets. In principle, we can combine the data sets and re-process the combined data. However, this would require a large amount of data processing. It is known that for many statistics (e.g., for population variance), we can avoid these lengthy computations: the statistic for the combined data can be computed based on the results of processing individual data sets.

In this paper, we show that a similar computational simplification is possible when instead of processing the exact values, we process privacy-related interval ranges for these values.

Acknowledgments. This work was supported in part by NSF grants EAR-0225670 and EIA-0080940, by Texas Department of Transportation grant No. 0-5453, and by the Max Planck Institut für Mathematik. The authors are thankful to the editors and to the anonymous referees for valuable suggestions.

References

1. Cowell, F.A.: Grouping bounds for inequality measures under alternative informational assumptions. J. of Econometrics 48, 1–14 (1991)
2. Dalenius, T.: Finding a needle in a haystack – or identifying anonymous census record. Journal of Official Statistics 2(2), 329–336 (1986)
3. Dantsin, E., Kreinovich, V., Wolpert, A., Xiang, G.: Population variance under interval uncertainty: a new algorithm. Reliable Computing 12(4), 273–280 (2006)
4. Denning, D.: Cryptography and Data Security. Addison-Wesley, Reading, MA (1982)
5. Duncan, G., Lambert, D.: The risk of disclosure for microdata. In: Proc. of the Bureau of the Census Third Annual Research Conference, Bureau of the Census, Washington, DC, pp. 263–274 (1987)
6. Duncan, G., Mukherjee, S.: Microdata disclosure limitation in statistical databases: query size and random sample query control. In: Prof. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May 20–22, 1991 (1991)
7. Fellegi, I.: On the question of statistical confidentiality. Journal of the American Statistical Association, 7–18 (1972)
8. Ferson, S., Ginzburg, L., Kreinovich, V., Longpré, L., Aviles, M.: Computing variance for interval data is NP-hard. ACM SIGACT News 33(2), 108–118 (2002)
9. Jaulin, L., Kieffer, M., Didrit, O., Walter, E.: Applied Interval Analysis. Springer-Verlag, London (2001)
10. Kim, J.: A method for limiting disclosure of microdata based on random noise and transformation. In: Proceedings of the Section on Survey Research Methods of the American Statistical Association, pp. 370–374 (1986)
11. Kirkendall, N., et al.: Report on Statistical Disclosure Limitations Methodology, Office of Management and Budget, Washington, DC, Statistical Policy Working Paper No. 22 (1994)
12. Kreinovich, V., Longpré, L., Starks, S.A., Xiang, G., Beck, J., Kandathi, R., Nayak, A., Ferson, S., Hajagos, J.: Interval versions of statistical techniques, with applications to environmental analysis, bioinformatics, and privacy in statistical databases. Journal of Computational and Applied Mathematics 199(2), 418–423 (2007)
13. Kreinovich, V., Xiang, G., Starks, S.A., Longpré, L., Ceberio, M., Araiza, R., Beck, J., Kandathi, R., Nayak, A., Torres, R., Hajagos, J.: Towards combining probabilistic and interval uncertainty in engineering calculations. Reliable Computing 12(6), 471–501 (2006)
14. Langewisch, A.T., Choobineh, F.F.: Mean and variance bounds and propagation for ill-specified random variables. IEEE Trans. SMC 34(4), 494–506 (2004)
15. Morgenstern, M.: Security and inference in multilevel database and knowledge base systems. In: Proc. of the ACM SIGMOD Conference, pp. 357–373 (1987)
16. Nguyen, H.T., Kreinovich, V., Gorodetski, V.I., Nesterov, V.M., Touloupiev, A.L.: Applications of interval-valued degrees of belief: a survey. In: Touloupiev, A. (ed.) Information Technologies and Intellectual Methods, vol. 3 (IT&IM'3), St. Petersburg Institute for Information and Automation of Russian Academy of Sciences (SPIIRAS), pp. 6–61 (in Russian) (1999)

17. Office of Technology Assessment, Protecting privacy in computerized medical information, US Government Printing Office, Washington, DC (1993)
18. Palley, M., Siminoff, J.: Regression methodology based disclosure of a statistical database. In: Proceedings of the Section on Survey Research Methods of the American Statistical Association, pp. 382–387 (1986)
19. Rabinovich, S.: Measurement Errors and Uncertainties. Springer, N.Y (2005)
20. Su, T., Ozsoyoglu, G.: Controlling FD and MVD inference in multilevel relational database systems. IEEE Transactions on Knowledge and Data Engineering 3, 474–485 (1991)
21. Sweeney, L.: Weaving technology and policy together to maintain confidentiality. Journal of Law, Medicine and Ethics 25, 98–110 (1997)
22. Sweeney, L.: Datafly: a system for providing anonymity in medical data. In: Lin, T.Y., Qian, S. (eds.) Database Security XI: Status and Prospects, Elsevier, Amsterdam (1998)
23. Vavasis, S.A.: Nonlinear Optimization. Oxford University Press, N.Y (1991)
24. Willenborg, L., De Waal, T.: Statistical disclosure control in practice. Springer Verlag, New York (1996)

Fast Service Restoration Under Shared Protection at Lightpath Level in Survivable WDM Mesh Grooming Networks[*]

Jacek Rak and Wojciech Molisz

Gdansk University of Technology, G. Narutowicza 11/12,
Pl-80-952 Gdansk, Poland
{jrak,womol}@eti.pg.gda.pl

Abstract. In this paper, a novel algorithm optimizing the utilization of backup path resources for survivable optical networks, based on graph vertex-coloring approach, is introduced. To the best of our knowledge, this is the first optimization technique, dedicated to WDM grooming networks, such that does not increase the length of backup paths and thus provides fast service restoration. Due to NP-completeness of the proposed Integer Linear Programming model, the respective heuristic algorithm has been developed.

The concept was evaluated for the U.S. Long-Distance Network, European COST 239 Network and Polish PIONIER Network. The results show that with only a little capacity utilization degradation, fast restoration can be achieved and the resource utilization kept at low level. The observed reduction in restoration time values is significant (up to 40%), compared to the typical *a priori* approach.

Keywords: network survivability, WDM mesh grooming networks, routing.

1 Introduction

This paper investigates the issues of service survivability in optical transport networks. The introduction of *wavelength division multiplexing* (WDM) in optical networks provided links consisting of sets of non-overlapping channels (wavelengths), each one capable of transmitting the data independently at peak electronic speed of a few Gbps. A failure of any network element may thus lead to large data and revenue losses [2, 14]. In WDM networks a demand to transmit data between a given pair of end nodes involves a setup of an end-to-end *connection*, originally referred to as a *lightpath*[1]. Each lightpath is established by allocating a sequence of wavelengths at consecutive links. In case of imposed *wavelength continuity constraint*, the lightpath must utilize the same wavelength on each link it uses, otherwise *wavelength conversions* may be performed at lightpath transit nodes.

[*] This work was partially supported by the Ministry of Science and Higher Education, Poland, under grants 3 T11D 001 30 and N517 022 32/3969.

[1] Each lightpath occupies the whole wavelength of each link it traverses.

Survivability of connections, originally defined as *the capability of a system to fulfill its mission in a timely manner, in the presence of attacks, failures or accidents* [4], is achieved by introducing redundancy. It means that for the main path of a connection, referred to as *active path*, there are additional (redundant) paths, called *backup paths*, used to protect the connection in case of a certain failure scenario [7, 9, 12, 19, 20]. Single network element failures are mostly considered. Survivability is based on either dedicating backup resources in advance (*protection* scheme) or on dynamic *restoration* [12, 19, 20]. Regarding the scope of a backup path, *path*, *region* or *link protection/restoration* is typically used [7, 9, 13, 19].

There is a significant difference between the typical bandwidth requirement of the end-user traffic demand and the capacity of a wavelength. Efficient sharing of the bandwidth of a wavelength by multiple traffic flows is necessary in order to avoid large waste of transmission resources. Low speed traffic needs to be multiplexed ("groomed") onto lightpaths. Traffic grooming problem can be formulated as follows: for a given request matrix(es) and network configuration, i.e., physical topology, type of a node, number of transceivers at each node, number of wavelengths at each fibre, the capacity of each wavelength, etc., determine which low speed demands should be multiplexed together on which wavelength. This leads to a multilayer architecture, as shown in Fig. 1. Examples of multilayer network architectures include: *IP over SONET over WDM* or *IP over WDM*.

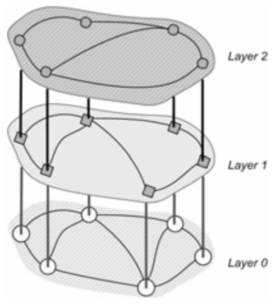


Fig. 1. An example structure of a three-layer network

If the set of connection requests is given in advance – we deal with static grooming, otherwise the grooming problem is dynamic. For static traffic grooming, the objective is to minimize the network cost, e.g. total number of wavelength-links. This is based on the assumption that the network operates in the non-blocking scenario. Otherwise (i.e. for a dynamic grooming) objective is to maximize the network throughput, because not all connections can be set up due to resource limitations. Traffic grooming is usually divided into four subproblems [23], which are not necessarily independent:

1) determining the virtual topology that consists of lightpaths,
2) routing the lightpaths over the physical topology,
3) performing wavelength assignment to the lightpaths,
4) routing the traffic on the virtual topology.

The first two problems are NP-complete, so traffic grooming in a mesh network is also NP-complete [23]. Past research efforts on traffic grooming have mainly focused on SONET/WDM ring networks, with the objective function of either to minimize the number of wavelengths or SONET ADMs. As fiber-optic backbone networks migrated from rings to mesh, traffic grooming on WDM mesh networks became an important area of research [1, 15, 23, 25]. The work in [24] reviews most of recent work on traffic grooming. Majority of papers on optical networks considered various problems separately; either traffic grooming, or protection/restoration. Here, we propose a model for fast service restoration under shared protection at lightpath level in WDM mesh grooming networks, multiplexing low-rate end-user traffic into a single WDM wavelength [1, 11, 15], using *Time Division Multiplexing* (TDM). These streams are then carried jointly by a certain lightpath (i.e. groomed).

In WDM grooming networks, survivability is typically provided at either lightpath or connection level. In *protection-at-lightpath* level (PAL) [15], operating at the aggregate (lightpath) level, survivability is provided on a lightpath end-to-end basis. It means that a single backup lightpath protects all the active paths of connections groomed into a given active lightpath. In PAL (Fig. 2), each connection can be realized by a sequence of active lightpaths, each one protected by a backup lightpath.

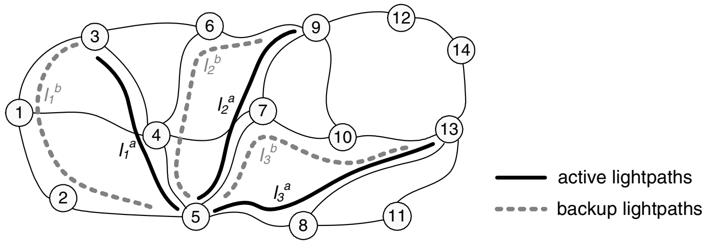


Fig. 2. An example of protection-at-lightpath (*PAL*). There are two connections established between pairs of nodes (3, 9) and (3, 13), respectively. The active paths of connections (3, 9) and (3, 13) utilize lightpaths (l_1^a, l_2^a) and (l_1^a, l_3^a), respectively, meaning that they are groomed onto a common active lightpath l_1^a at links (3, 4) and (4, 5). The backup paths, providing end-to-end protection with regard to active lightpaths, utilize backup lightpaths (l_1^b, l_2^b) and (l_1^b, l_3^b), respectively, meaning that backup lightpath l_1^b provides end-to-end protection with regard to active lightpath l_1^a with two groomed active paths.

In contrast, *protection-at-connection* level (PAC) [15] operates at a per-flow basis and assumes that each active path is protected by a single end-to-end backup. In PAC several backup paths may utilize the same wavelength, but at different time slots.

Assuring survivability of connections by establishing backup paths increases the ratio of link capacity utilization, which in turn limits the number of connections that may be established. However, this ratio can be reduced by applying sharing the link capacities that are reserved for backup paths [6, 9, 19]. Such sharing is possible, if these paths are to protect against different failure scenarios (i.e. if the respective protected parts of active paths are mutually disjoint) [6, 9]. However, typical optimization technique, here called the *a priori* approach, results in increasing the length of backup paths, compared to the "no optimization" case. This is due to performing the optimization when calculating the costs of links to be used in backup paths. Independent of the real link lengths, these link costs are set to 0, if only their capacities may be shared. As a result, the time of connection restoration gets increased, since, due to the three-way handshake protocol [20], the longer the backup paths are, the more time it takes to perform recovery actions.

Although the issues of assuring fast service restoration and the optimization of link capacity utilization have been separately investigated in several papers, e.g. in [6, 16], they still have not been extensively studied jointly. In general it is not possible to achieve the minima of both the objectives. The papers [17] and [18] consider the problem of minimizing both the functions for the case of sharing the backup paths assuming no traffic grooming applied. A slightly different approach is proposed in

[16], where a model to achieve fast connection restoration by minimizing the length of backup paths is introduced. It incorporates a parameter μ to each backup path link cost in a way that the cost reflects both the link length and the ratio of unused capacity that the backup path may use. However, there is no approach introduced to provide fast service restoration under backup path sharing for WDM grooming networks.

This paper is to propose the algorithm that jointly optimizes the ratio of link capacity utilization and provides fast service restoration, dedicated to WDM grooming networks with full wavelength conversion capability and the case of providing survivability at lightpath level (PAL). To the best of our knowledge, this is the first resource utilization optimization approach operating at subwavelength granularity such that does not increase the length of backup paths. The traffic is assumed to be *static* (i.e. the set of connection requests is given in advance). The proposed technique may be also used for the case of *dynamic traffic* where connection requests arrive one at a time. However, in such blocking scenario, different from the one considered here, the objective is to minimize the network resources used for each request, leading to several grooming policies.

The rest of the paper is organized as follows. The next section discusses first the typical *a priori* optimization problems, extended here to the case of protection at lightpath level in WDM grooming networks. In the following section, the ILP optimization problem is stated, which is NP-complete, so the respective heuristic algorithm is proposed. In Section 4, results of modeling, obtained for the U.S. Long-Distance Network, European COST 239 Network and Polish PIONIER Network, are given in detail and discussed. They include the ratio of total link capacity utilization (i.e. for both active and backup lightpaths), the length of backup lightpaths and the values of service restoration time in WDM layer. They show that the proposed solution significantly outperforms the typical resource utilization optimization approach in terms of obtained values of service restoration time for the price of only little degradation in the ratio of link capacity utilization. The results are the original contribution and are published for the first time.

2 Sharing the Backup Path Capacities in WDM Grooming Networks Under PAL

Sharing the backup lightpath capacities is possible, if these lightpaths are to protect against different failure scenarios, i.e. if the respective protected parts of active lightpaths are mutually disjoint. In other words, it must be guaranteed, that under any network element failure scenario there is no need to activate more than one backup lightpath sharing the common bandwidth unit. Considering the optimization strength, we may typically have intra-(inter-)demand sharing between backup lightpaths protecting disjoint parts of the same (different) active lightpaths, respectively.

Typical *a priori* optimization is performed before finding backup lightpaths [6, 9], when calculating the cost of ξ_{ij} of each link l_{ij} to be used in a backup lightpath k, according to Eq. 1. However, as already stated, the found backup lightpaths are not the shortest ones here, since the calculated costs of links are often not equal to the link lengths. Any link l_{ij}, even a long one, may be used in a backup lightpath, since its cost ξ_{ij} equals 0, if only it has enough backup capacity that may be shared (if $m_{ij}^{(k)} \geq r^{(k)}$). Long backups make in turn the process of connection restoration time-consuming.

$$\xi_{ij} = \begin{cases} 0 & \text{if } r^{(k)} \le m_{ij}^{(k)} \\ (r^{(k)} - m_{ij}^{(k)}) \cdot s_{ij} & \text{if } r^{(k)} > m_{ij}^{(k)} \text{ and } f_{ij} \ge r^{(k)} - m_{ij}^{(k)} \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

where:

$r^{(k)}$	-	the requested capacity,
$m_{ij}^{(k)}$	-	the capacity reserved so far at a link l_{ij} (for the backups of the already established active lightpaths) that may be shared,
f_{ij}	-	the unused capacity of a link l_{ij}
s_{ij}	-	the length of a link l_{ij}.

The optimization of resource utilization introduced here, given by Algorithm FSR-SLL, is performed after establishing the connections. Its scope is confined to each single link l_{ij}, so sharing the backup lightpaths is performed within the set of backup lightpaths installed at wavelengths of a given link l_{ij} only. This in turn implies that the backup lightpaths, originally established as the shortest ones, remain unchanged, i.e. they use the same links which they used before applying the optimization. Since the length of the backup lightpaths is not increased, fast service restoration is possible.

The proposed FSR-SLL algorithm, given in Fig. 3, is executed at each network link l_{ij} independently. For each link l_{ij} it tries to reorganize the assignment of backup channels by dividing the set of backup lightpaths B_{ij} into subsets B_{ij}^{s}, such that each subset contains backup lightpaths that protect mutually disjoint active lightpaths and thus may share a common link channel. The number of subsets B_{ij}^{s} must be minimized, since it denotes the number of channels that will become allocated at l_{ij} for backup lightpaths after applying the optimization.

```
INPUT
  Network topology; the sets $B_{ij}$ of backup lightpaths installed on channels of links $l_{ij}$;
  the sets of free channels at links $l_{ij}$; the sets of channels allocated for active
  lightpaths at links $l_{ij}$
OUTPUT
  The sets $B_{ij}^{s}$ that determine the new assignment of link channels to backup lightpaths

For each link $l_{ij}$:
Step 1.   Divide the set $B_{ij}$ of backup lightpaths, installed on channels of a link $l_{ij}$ into
          subsets $B_{ij}^{s}$ such that:
            – each subset contains backup lightpaths that may share capacity one another
            – the number of subsets $B_{ij}^{s}$ is minimized
Step 2.   For each subset $B_{ij}^{s}$:
          Step 2a.  delete original link channel allocations for the backups of $B_{ij}^{s}$
          Step 2b.  apply sharing by allocating one common unit of bandwidth for all
                    the backups of $B_{ij}^{s}$
Computational complexity: NP-complete
```

Fig. 3. FSR-SLL algorithm (optimization of backup path capacity utilization providing Fast Service Restoration for Shared protection at Lightpath Level)

The problem of optimally dividing the set B_{ij} of backup lightpaths installed at each link l_{ij} into subsets B_{ij}^{s} (Step 1 of FSR-SLL algorithm) is NP-complete as it equivalent to the *vertex-coloring* problem of an induced graph of conflicts G_{ij}, which is also NP-complete [10]. Generally, in such a *graph of conflicts* $G = (V, E)$ there is an edge between any two vertices v and w, if and only if there is a conflict between them.

The objective is to find a partition of vertices of G into a minimal number of subsets of mutually compatible vertices, i.e. into subsets of pair-wise non adjacent (i.e. non-conflicting) vertices. As given in [8], a (*proper*) *coloring* of vertices of a graph G is a mapping f: $V \rightarrow C$, where V is a set of vertices of G and C is a finite set of colors, each color being represented by an integer number such that neighboring vertices are assigned different colors (i.e. if an edge $vw \in E$ then $f(v) \neq f(w)$).

For each link l_{ij} an induced graph G_{ij} is constructed, such that:

- its vertices denote backup lightpaths that are installed at channels of a link l_{ij};
- there is an edge between a given pair of vertices u and v in G_{ij}, if and only if there is a conflict between the respective backup lightpaths (i.e. when the protected parts of active lightpaths are not mutually disjoint), implying that the backups must be assigned different units of bandwidth;
- the colors assigned to vertices of G_{ij} define the assignment of backup lightpaths to certain wavelengths of a link l_{ij}.

It turns out that in case of full wavelength conversion capability, no strict assignment of wavelengths to certain backup lightpaths is needed at each link l_{ij} and that it is sufficient only to calculate the minimum number of wavelengths required under backup lightpath capacity sharing.

However, this problem is still NP-complete, since it is equivalent to the issue of finding the chromatic number $\chi(G_{ij})$ [10] being the smallest number of colors required to color the vertices of G_{ij}. One of the known bounds on $\chi(G_{ij})$ are:

$$\omega(G_{ij}) \leq \chi(G_{ij}) \leq \Delta(G_{ij}) + 1 \qquad (2)$$

where:

$\Delta(G_{ij})$ -	the maximum degree[2] of a vertex in G_{ij},
$\omega(G_{ij})$ -	the maximum size of the fully connected subgraph of G_{ij}, referred to as *clique* [10].

Using the upper bound of $\Delta(G_{ij})+1$ to estimate the number of link channels needed at l_{ij} may, however, give the results far greater from the optimal ones. For instance, if G_{ij} has a topology of a *star* [10] with n = 9 vertices, as given in Fig. 4, where $\Delta(G_{ij}) = n$-1 and $\omega(G_{ij})$ = 2 then n units of bandwidth would be used instead of the sufficient two units. This implies that even under full

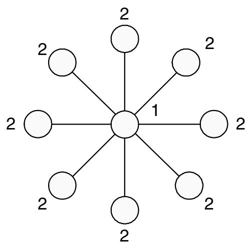


Fig. 4. Example star topology with assigned colors. $\Delta(G_{ij})$=n-1=8; $\omega(G_{ij})$=2 and $\chi(G_{ij})$=2.

[2] Degree of a vertex v in G_{ij}, defined as $deg(v)$ is the number of edges incident to v in G_{ij}.

wavelength conversion capability, knowledge of $\chi(G_{ij})$ is necessary. What is more, in order to obtain the value of $\chi(G_{ij})$, one must always color the vertices of a G_{ij}, meaning that when calculating the value of $\chi(G_{ij})$, a vertex graph-coloring of G_{ij} is also returned.

In Sections 2.1 and 2.2 the algorithms of finding both the optimal as well as approximate values of $\chi(G_{ij})$, respectively, to determine the minimal number backup channels for each link l_{ij} (Step 1 of Algorithm 1) for protection at lightpath level and full wavelength conversion capability, will be introduced. For each link, they utilize a graph vertex-coloring routine and thus apart from returning the value of $\chi(G_{ij})$, they also find the strict wavelength assignment with respect to backup lightpaths of l_{ij}.

2.1 ILP Model of Sharing the Backup Capacities for Traffic Grooming (FSR-SLL-VCO)

variables

$k = 1,2,\ldots, K$	numbers of backup lightpaths in a network
$c = 1,2,\ldots, C$	numbers of colors to be assigned; C - the maximum allowed color - is set to $\Delta(G_{ij})+1$
$x_k^{\ c}$	takes value of 1, if a backup lightpath k is assigned a color c, 0 otherwise
b^c	takes value of 1, if a color c is assigned to any vertex in G_{ij}, 0 otherwise

objective

For a given graph G_{ij} it is to find optimal backup lightpath bandwidth sharing by minimizing the total number of used wavelengths at l_{ij} and thus minimizing the following cost:

$$F = \sum_{c=1}^{C} b^c \tag{3}$$

constraints

(1) on assigning each vertex k of a given graph G_{ij} one color c only

$$\sum_{c=1}^{C} x_k^{\ c} = 1; \qquad k = 1,2,\ldots,K; \tag{4}$$

(2) assuring that neighboring vertices of G_{ij} receive different colors

$$x_k^{\ c} + x_m^{\ c} \le b^c \text{ for each edge } e = (k,m) \text{ in } G_{ij}; \quad c = 1,2,\ldots,C; \tag{5}$$

(3) excluding the link channels allocated for active lightpaths

$$\sum_{k=1}^{K} \sum_{c=1}^{C} x_k^{\ c} = 0 \qquad \text{if } x_k^{\ c} \text{ denotes a link channel } c \text{ allocated at link } l_{ij} \text{ for an active lightpath } k \tag{6}$$

(4) on taking the allowable values

$$\begin{aligned} &x_k^{\,c} \in \{0,1\}; \quad k = 1,2,\ldots,K; \; c = 1,2,\ldots,C; \\ &b^c \in \{0,1\}; \qquad\quad c = 1,2,\ldots,C; \end{aligned} \tag{7}$$

2.2 Heuristic Algorithm of Sharing the Backup Capacities in WDM Grooming Networks (FSR-SLL-VCH)

Since the problem formulated in Section 2.1 is NP-complete, we introduce the following FSR-SLL-VCH heuristic algorithm of polynomial computational complexity (Fig. 5).

INPUT

Network topology; the sets B_{ij} of backup lightpaths installed on channels of links l_{ij}; the sets of free channels at links l_{ij},; the sets of channels originally allocated for active lightpaths at links l_{ij}

OUTPUT

The sets $B_{ij}^{\,s}$ that determine the new assignment of link channels to backup lightpaths

For each link l_{ij}:

Step 1. Create graph of conflicts G_{ij} for the set B_{ij} of backup lightpaths

Step 2. Divide the set B_{ij} of backup lightpaths, installed on channels of a link l_{ij} into subsets $B_{ij}^{\,s}$ by coloring the vertices in G_{ij} using the one of the heuristic algorithms of graph vertex-coloring

Step 3. Delete original link channels allocations for the backup lightpaths of l_{ij} and apply backup bandwidth sharing by allocating one common channel for backup lightpaths from each subset $B_{ij}^{\,s}$, using channels not used by active lightpaths

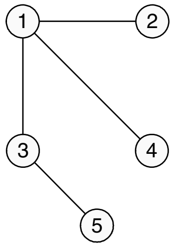
Computational complexity: polynomial (depends on the complexity of graph vertex-coloring routine used in Step 2)

Fig. 5. FSR-SLL-VCH algorithm (computationally effective version of FSR-SLL algorithm using graph Vertex Coloring Heuristics)

As for the graph vertex-coloring heuristic algorithm, to be used in Step 2 of the above algorithm, for instance *Largest First* (LF) or *Saturation Degree Ordering* (DSATUR) algorithm, described in [10] may be used, since either of them provides good results in terms of minimizing the total number of utilized colors for the price of acceptably low polynomial computational complexity ($O(n + m)$ and $O((n + m)\log n)$, where n and m denote the numbers of vertices and edges in G_{ij}, respectively).

In LF, vertices of each G_{ij} are first ordered descending their degree values and then sequentially assigned the lowest possible color based on their position in this ordering, meaning that vertices of higher degree receive their colors first. DSATUR uses the *saturation degree* of each vertex v (instead of a typical vertex degree), defined as the number of already differently colored neighbors of v.

Fig. 6 shows an example vertex-coloring of a graph of conflicts G_{ij}, using the LF algorithm. Suppose that at a link l_{ij} there are five backup lightpaths (occupying five wavelengths) with conflicts as given in Fig. 6a. After applying the LF algorithm, it is clear from (6b) that two colors were sufficient to color the vertices of G_{ij}, implying that two wavelengths will be finally needed at l_{ij} for backup lightpaths (instead of previous five).



descending ordering of G_{ij} vertices	1	3	5	2	4
degrees of G_{ij} vertices	3	2	1	1	1
color assignment	1	2	1	2	2

(a) graph of conflicts G_{ij} (b) vertex-coloring of G_{ij}

Fig. 6. Example vertex-coloring obtained using *LF* algorithm

3 Modeling Assumptions

The modeling was performed for the U.S. Long-Distance Network [22], European COST 239 Network [21] and Polish PIONIER Network [5], presented in Figs. 7-9, respectively. Due to NP-completeness of the original problem, only heuristic algorithms were used in computations. Simulations were performed using our dedicated network simulator implemented in C++. Their goals were to measure the link capacity utilization ratio and the values of connection restoration time in WDM layer for the static traffic case. Time of lightpath restoration was measured according to [12, 20] and comprised: time to detect a failure, link propagation delay, time to configure backup lightpath transit nodes and message processing delay at network nodes (including queuing delay). All the network links were assumed to have 8 channels each with grooming capability $g = 4$ (i.e. offering 4 subchannels each of equal capacity). Channel capacity unit was considered to be the same for all the network links. Network nodes were assumed to have a full channel (wavelength) conversion capability.

For each connection, the following properties were assumed:

- protection against a single link failure (each of the failure states consisted of a failure of one link at a time, the rest being fully operational),
- protection at lightpath level (PAL) implying the use of a single backup lightpath for all the active paths groomed onto a common active lightpath,
- LF graph vertex-coloring algorithm for backup lightpath capacity sharing at each link l_{ij},
- a demand of resource allocation equal to the capacity of one subchannel (equal to the value of one channel capacity divided by g),
- provisioning 100% of the requested bandwidth after a failure of a network element,

– a demand to assure unsplittable flows (both for active and backup lightpaths),
– the distance metrics and Dijkstra's shortest lightpath algorithm [3] in all lightpath computations,
– the three-way handshake protocol of restoring the connections (the exchange of `LINK FAIL`, `SETUP` and `CONFIRM` messages), described in [12, 20].

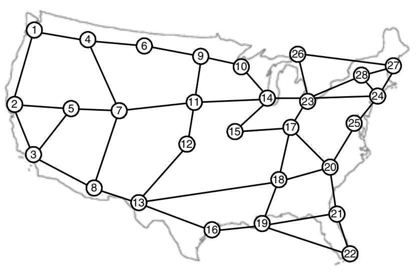


Fig. 7. U.S. Long-Distance Network (28 nodes, 45 bidirectional links)

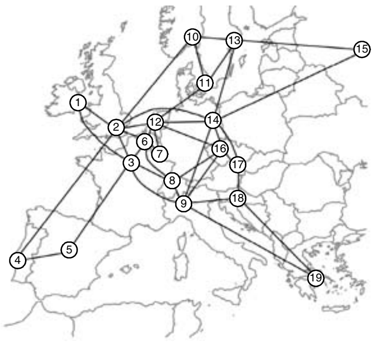


Fig. 8. European COST 239 Network (19 nodes, 37 bidirectional links)

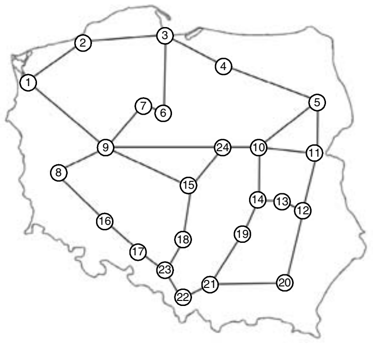


Fig. 9. Polish PIONIER Network (24 nodes, 31 bidirectional links)

The features of the proposed optimization technique were tested using two scenarios of network load. In the first case, the set of connection demands consisted of 10% of all the possible network node pairs, which amounted to 38, 17 and 28 demands for U.S. Long-Distance Network, European COST 239 Network and Polish PIONIER Network, respectively. Using this scenario there were always enough resources to establish all the demanded connections.

The second scenario was intended to test the properties of the proposed optimization technique under varying network load. In this case, only the U.S. Long-Distance Network was examined. The size of connection demand set varied from 10 to 100% of all the possible network node pairs.

During a single modeling, shown below in Fig. 10, one particular variant of optimization strength was provided for all the connections.

Repeat r^* times the following steps:

Step 1. Randomly choose a given number of pairs of source *s* and destination *d* nodes

Step 2. Try to establish the survivable connections utilizing any of the routing algorithms allowing traffic grooming and providing protection at lightpath
level with the respective optimization of backup lightpath capacity utilization**

Step 3. Store the ratio of link capacity utilization and the lengths of the backups

Step 4. t^* times simulate random failures of single links. For each failure state restore connections that were broken and note the values of connection restoration time

* *during each investigation, r = 30 and t = 30 were assumed*
** *any type of sharing is allowed here*

Fig. 10. Research plan

4 Modeling Results

4.1 WDM Layer Link Capacity Utilization Ratio

Figs. 11-13 show the average values of link capacity utilization ratio for all variants of optimization strength in ascending order, while Table 1 gives the lengths of the respective 95% confidence intervals. The obtained results prove that the proposed FSR-SLL-VCH optimization routine remarkably reduces the ratio of link capacity utilization. Under FSR-SLL-VCH, 40% less resources were needed on average to establish survivable connections, compared to the "no optimization case". The best results were obtained for PIONIER Network with parallel intra- and inter-demand sharing, for which FSR-SLL-VCH reduced the link capacity utilization even up to 58%. It's worth mentioning that, although the *a priori* optimization gives slightly better link capacity utilization ratio in all cases, the length of backup lightpath and

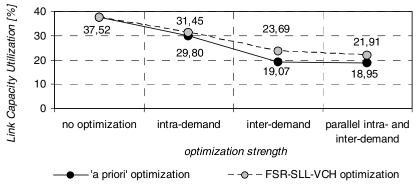


Fig. 11. Average ratio of link capacity utilization as a function of optimization strength for U.S. Long-Distance Network

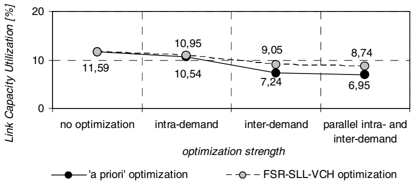


Fig. 12. Average ratio of link capacity utilization as a function of optimization strength for European COST 239 Network

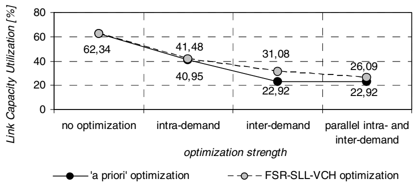


Fig. 13. Average ratio of link capacity utilization as a function of optimization strength for Polish PIONIER Network

Table 1. Lengths of the 95% confidence intervals for the mean values of link capacity utilization [%]

		intra-demand	inter-demand	intra- & inter-demand
U.S. Long-Distance	*a priori* optimization	1,39	0,99	1,02
	FSR-SLL-VCH optimization	1,72	1,21	1,06
European COST 239	*a priori* optimization	0,64	0,56	0,54
	FSR-SLL-VCH optimization	0,73	0,67	0,65
Polish PIONIER	*a priori* optimization	1,91	1,41	1,41
	FSR-SLL-VCH optimization	1,84	1,33	1,18

service restoration times, discussed in next subsections, are much better when our FSR-SLL-VCH algorithm is used.

4.2 Length of a Backup Lightpath

Figs. 14-16 show the average lengths of backup lightpaths for all variants of optimization strength. The results prove that when using our FSR-SLL-VCH optimization, the average length of backup lightpaths is not increased and remains at the same level as in case no optimization is performed. This is true regardless of the optimization strength. Fast restoration is thus possible. They also show that after applying the common *a priori* optimization, the length of backup lightpaths is often far from optimal. Additionally, the stronger the *a priori* optimization is chosen, the longer the backup lightpaths are. For the extreme case, when parallel intra- and inter-demand sharing was used, the average backup lightpath length for the *a priori* optimization was even about 63%, 53% and 33% worse, compared to the results of our FSR-SLL-VCH approach, obtained for U.S. Long-Distance Network, European COST 239 Network and Polish PIONIER Network, respectively.

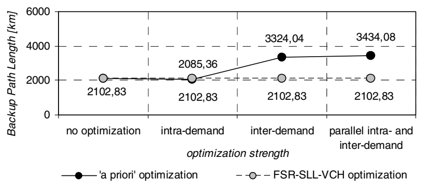


Fig. 14. Average length of backup lightpath as a function of optimization strength for U.S. Long-Distance Network

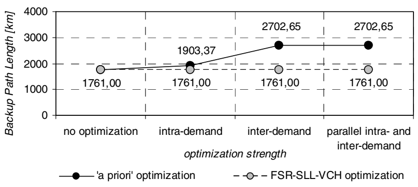


Fig. 15. Average length of backup lightpath as a function of optimization strength for European COST 239 Network

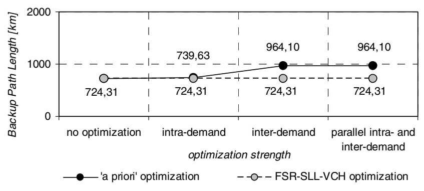


Fig. 16. Average length of backup lightpath as a function of optimization strength for Polish PIONIER Network

4.3 Values of Service Restoration Time at WDM Layer

Figs. 17-19 show the average values of connection restoration time for all variants of optimization strength, while Table 2 gives the lengths of the respective 95% confidence intervals. It is worth mentioning that the obtained mean values of restoration time for our FSR-SLL-VCH optimization were always similar to the shortest ones, achieved when no optimization was performed. When compared to the results of the *a priori* optimization, the difference is remarkable. Under PAL, it took

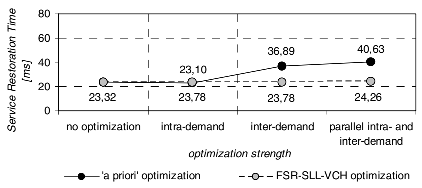


Fig. 17. Average values of service restoration time for various optimization strengths (U.S. Long-Distance Network)

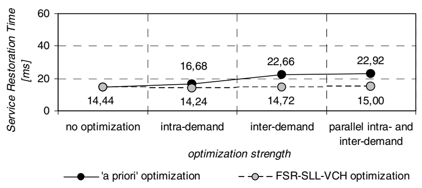


Fig. 18. Average values of service restoration time for various optimization strengths (European COST 239 Network)

Table 2. Lengths of the 95% confidence intervals for the mean values of service restoration time [ms]

		intra-demand	inter-demand	intra- & inter-demand
U.S. Long-Distance	*a priori* optimization	2,40	3,04	2,90
	FSR-SLL-VCH optimization	2,69	2,33	2,49
European COST 239	*a priori* optimization	1,86	2,29	2,38
	FSR-SLL-VCH optimization	2,08	1,38	1,74
Polish PIONIER	*a priori* optimization	0,26	0,67	0,71
	FSR-SLL-VCH optimization	0,52	0,54	0,66

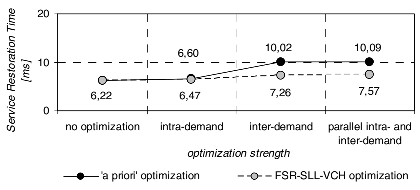


Fig. 19. Average values of service restoration time for various optimization strengths (Polish PIONIER Network)

up to 30% less time on average to restore a connection, when our FSR-SLL-VCH optimization was used. In particular, for the U.S. Long-Distance Network with the parallel intra- and inter-demand sharing applied, it took even about 40% less time (24.26 ms against 40,63 ms) to restore a broken connection.

4.4 Modeling Results for Varying Network Load

The experiment was performed for the U.S. Long-Distance Network. Regarding the strength of optimization, parallel intra- and inter-demand sharing was used. All other modeling assumptions were the same as given in Section 3. The goal of modeling was to test the properties of the proposed FSR-SLL-VCH optimization under different network loads. For that purpose, 10 different sizes of demand sets, given in the horizontal axes of Figs. 20-22, were used, changing from 10 to 100% with the step of 10%. They corresponded to the real average link capacity utilization changing from 15 to 75%. For instance, the demand set size of 30% meant the attempts to establish connections between 30% of all possible pairs of nodes randomly chosen.

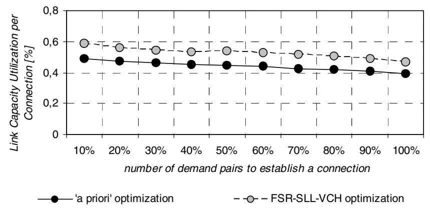


Fig. 20. Average ratio of link capacity utilization as a function of demand set size for U.S. Long-Distance Network

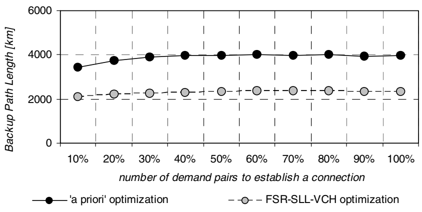


Fig. 21. Average length of backup lightpath as a function of demand set size for U.S. Long-Distance Network

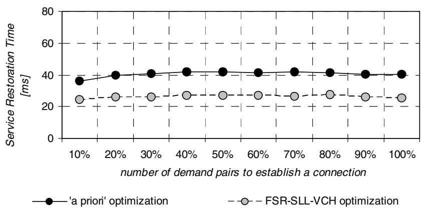


Fig. 22. Average values of service restoration time as a function of demand set size for U.S. Long-Distance Network

The results prove that the properties of the proposed FSR-SLL-VCH optimization scale well with the increase of the network load. Regardless of the network load:

– the efficiency of link capacity utilization was about 19% worse (Fig. 20),
– the length of backup lightpath was always about 40% shorter (Fig. 21),
– the values of connection restoration time were about 40% smaller (Fig. 22),

for our optimization algorithm, compared to the results of the typical *a priori* routine.

The average link capacity utilization per connection decreases for both optimization approaches with the increase of the network load (Fig. 20), since the greater the network load is, the more backup lightpaths are likely to share a common wavelength.

5 Conclusion

Obtained results confirm that one cannot obtain the values of both the connection restoration time and the link capacity utilization at the minimum level. The proposed FSR-SLL-VCH algorithm optimizing the resource utilization, dedicated to protection at lightpath level for survivable WDM grooming networks, significantly reduces the ratio of link capacity utilization while simultaneously providing fast service restoration. It does not increase the lengths of backup lightpaths, which is true regardless of the optimization strength. Typical *a priori* optimization is, however, still more capacity effective, but for the price of much greater restoration time values.

References

1. Cinkler, T.: Traffic and λ Grooming. IEEE Network 17(2), 16–21 (2003)
2. Colle, D., et al.: Data-centric optical networks and their survivability. IEEE J. Select. Areas Communications 20, 6–21 (2002)
3. Dijkstra, E.: A note on two problems in connection with graphs. Numerische Mathematik 1, 269–271 (1959)
4. Ellison, R.J., Fisher, D.A., Linger, R.C., Lipson, H.F., Longstaff, T., Mead, N.R.: Survivable network systems: an emerging discipline, Technical Report CMU/SEI-97-TR-013, Carnegie Mellon University, Software Engineering Institute (1997)
5. http://www.pionier.gov.pl/siec/infrastruktura.htm
6. Ho, P-H., Tapolcai, J., Cinkler, T.: Segment shared protection in mesh communications networks with bandwidth guaranteed tunnels. IEEE/ACM Transactions on Networking 12(6), 1105–1118 (2004)
7. Kawamura, R.: Architectures for ATM network survivability. IEEE Communications Surveys 1(1), 2–11 (1998)
8. Klotz, W.: Graph coloring algorithms (2000), www.math.tuclausthal.de/Arbeitsgruppen/ Diskrete Optimierung/ publications/2002/gca.pdf
9. Kodialam, M., Lakshman, T.V.: Dynamic routing of locally restorable bandwidth guaranteed tunnels using aggregated link usage information. In: Proc. IEEE INFOCOM'01, pp. 376–385 (2001)
10. Hansen, P.: Graph coloring and applications, American Mathematical Society (1999)
11. Modiano, E., Lin, P.J.: Traffic Grooming in WDM Networks. IEEE Communications Magazine 39(7), 124–129 (2001)
12. Molisz, W.: Survivability issues in IP-MPLS networks. Systems Science 31(4), 87–106 (2005)
13. Molisz, W., Rak, J.: Region protection/restoration scheme in survivable networks. In: Gorodetsky, V., Kotenko, I., Skormin, V.A. (eds.) MMM-ACNS 2005. LNCS, vol. 3685, pp. 442–447. Springer, Heidelberg (2005)
14. Mukherjee, B.: WDM Optical Communication Networks: Progress and Challenges. IEEE Journal on Selected Areas in Communications 18(10), 1810–1823 (2000)

15. Ou, C., et al.: Traffic grooming for survivable WDM networks – shared protection. IEEE Journal on Selected Areas in Communications 21(9), 1367–1383 (2003)
16. Qiao, Ch., et al.: Novel models for efficient shared path protection, OFC, pp. 545–547 (2002)
17. Rak, J.: Capacity efficient shared protection and fast restoration scheme in self-configured optical networks. In: Keller, A., Martin-Flatin, J.-P. (eds.) SelfMan 2006. LNCS, vol. 3996, pp. 142–156. Springer, Heidelberg (2006)
18. Rak, J.: Priority-enabled optimization of resource utilization in fault-tolerant optical networks. In: Gerndt, M., Kranzlmüller, D. (eds.) HPCC 2006. LNCS, vol. 4208, pp. 863–873. Springer, Heidelberg (2006)
19. Ramamurthy, S., Mukherjee, B.: Survivable WDM mesh networks, part I – protection. In: Proc. IEEE INFOCOM'99, pp. 744–751 (1999)
20. Ramamurthy, S., Mukherjee, B.: Survivable WDM mesh networks, part II – restoration. In: Proc. IEEE Integrated Circuits Conference'99, pp. 2023–2030 (1999)
21. Wauters, N., Demeester, P.: Design of the optical path layer in multiwavelength cross connected networks. IEEE Journal on Selected Areas in Communications 1(5), 881–892 (1996)
22. Xiong, Y., Mason, L.G.: Restoration strategies and spare capacity requirements in self healing ATM networks. IEEE/ACM Transactions on Networking 7(1), 98–110 (1999)
23. Zhu, K., Mukherjee, B.: Traffic grooming in an optical WDM mesh network. IEEE Journal on Selected Areas in Communications 20, 122–133 (2002)
24. Zhu, K., Mukherjee, B.: A review of traffic grooming in WDM optical networks: Architectures and challenges. SPIE Opt. Networks Mag. 4(2), 55–64 (2003)
25. Zhu, H., Zang, H., Zhu, K., Mukherjee, B.: A novel generic graph model for traffic grooming in heterogeneous WDM mesh networks. IEEE/ACM Transactions on Networking 11(2), 285–299 (2003)

Anycast Communication – A New Approach to Survivability of Connection-Oriented Networks

Krzysztof Walkowiak

Chair of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
Tel.: (+48)713203539, fax. (+48)713202902
Krzysztof.Walkowiak@pwr.wroc.pl

Abstract. In this paper, we tackle the question of how anycast communication can improve survivability of connection-oriented networks. The focus on network survivability arises naturally in the context of growing role of computer networks in almost all aspects of human life. We concentrate on connection-oriented networks e.g. MPLS, ATM for the reason that these techniques can provide effective and reliable services in transport network. We take a detailed view on anycast communication from the perspective of network survivability. A new optimization problem is formulated that is equivalent to the problem of joint unicast and anycast flows restoration in connection-oriented networks. Next, we propose a heuristic algorithm solving that problem. Finally, we present results of exhaustive numerical experiments and a comprehensive discussion on the application of anycast communication in the context of network survivability.

Keywords: survivability, connection-oriented network, anycast.

1 Introduction

In this paper we address the problem of survivability in connection-oriented (c-o) computer networks. Recently, we can observe that c-o networks like MultiProtocol Label Switching (MPLS) gain much attention especially in transport backbone networks. This follows from the fact that c-o transmission enables delivering of traffic engineering capability and QoS performance including reliability issues [4]. Connection-oriented network technologies use quite simple and effective method to enable network survivability. The main idea of this approach is as follows. Each connection, i.e. label switched path (LSP) in MPLS networks, has a working path (route) and a recovery (backup) path (route). The working route is used for transmitting of data in normal, failure-free state of the network. After a failure of the working path, the failed connection is switched to the backup route. In order to provide network survivability two methods can be used: protection and restoration. The distinction between protection and restoration consists in the different time scale in which they operate. Protection needs preallocated network resources while restoration applies dynamic resource establishment. The concept of backup routes can be applied in both protection and restoration methods. For more details on survivability of connection-oriented networks refer to [4].

Most of the interest in the context of network survivability networks has been focused on the unicast traffic [4], [10], [15]. This is obvious, since unicast routing between a specified pair on network nodes is relatively well understood in best effort networks [6]. However, various techniques applying anycast flow paradigm have gained much attention in recent years. Anycast is a *one-to-one-of-many* technique to deliver a packet to one of many hosts. It is assumed that the same information is replicated in many servers located in the network. Therefore, the user can select one of these servers according to some criteria including also QoS parameters. Consequently, anycast transmission can reduce network traffic and avoid congestion causing big delays in data delivery. An additional benefit is that replica servers provide fault-tolerant service, since users can select another server offering the same data, and even a failure of one server does not cause the data to be unreachable. For these two reasons, in this paper we apply the anycast transmission to improve network survivability.

The main novelty of this paper is that we consider anycasting in connection-oriented network. Most of previous research on anycasting focuses on pure IP networks [1], [2], [5], [6], [12], [14]. Recall that connection-oriented networks (e.g. MPLS) offer a number of traffic engineering mechanisms that can significantly improve QoS performance of the network comparing to pure IP networks. There has been a lot of papers addressing the offline unicast routing of c-o networks (see [15] and the references therein).

In this work we consider an existing backbone network. In many cases the network is in an operational phase and augmenting of its resources (links, capacity, replica servers) or changing location of replica servers is not possible in a short time perspective. Moreover, we assume that to provide network survivability the restoration approach is applied. Our main goal is to examine how the application of anycast communication improves the network restoration. Therefore, we formulate a new offline optimization problem of unicast and anycast connections restoration and propose an heuristic algorithm to solve that problem. Similar approach has been proposed in [17], however in this work we apply restoration, while in [17] the protection method was used. Note that in [20] we considered an online version of the considered problem. To verify our approach we report numerous experimental results and present a comprehensive discussion on results.

2 Anycasting in Connection-Oriented Networks

Anycast is a *one-to-one-of-many* technique to deliver a packet to one of many hosts. Comparing to unicast transmission, in anycasting one of the endpoints of the transmission must be selected among many possible nodes. One of the most well-known technologies that applies anycast traffic is Content Delivery Network (CDN). CDN is defined as mechanisms to deliver a range of content to end users on behalf of the origin Web servers. Information is offloaded from source sites to other content servers located in different locations in the network. For each request, CDN tries to find the closest server offering the requested Web page. CDN delivers the content from the origin server to the replicas that are much closer to end-users. The set of content stored in CDNs servers is selected carefully. Thus, the CDNs' servers can approach the hit ratio of 100%. It means that almost all request to servers are satisfied [14]. Other applications of anycast paradigm in computer networks are Domain Name Service, Web Service, Distributed Database System, host auto-configuration, overlay network, peer-to-peer (P2P) systems, sensor networks [1], [2], [5], [6], [7].

Most of previous work on anycasting concentrates on IP networks using connection-less transmission modeled as bifurcated multicommodity flows [5], [6], [12], [14]. Since we consider a connection-oriented network (e.g. MPLS), we model the network flow as non-bifurcated multicommodity flow. Consequently, the anycast request (demand) must consist of two connections: one from the client to the server (upstream) and the second one in the opposite direction (downstream). Upstream connection sends user's requests. Downstream connection carries the requested data. Thus, each anycast demand is defined by a following triple: client node, upstream bandwidth requirement and downstream bandwidth requirement. In contrast, a unicast demand is defined by a following triple: origin node, destination node and bandwidth requirement. To establish a unicast demand a path satisfying requested bandwidth and connecting origin and destination nodes must be found. Optimization of anycast demands is more complex. The first step is the server selection process. Next, when the server node is selected, both paths: upstream and downstream can be calculated analogously to unicast approach. However, the main constraint is that both connections associated with a particular anycast demand must connect the same pair of network nodes and one of these nodes must host a server.

Two approaches proposed for protection of anycast flows [16], [17] can be also applied in the context of restoration. The first method – called backup replica method – assumes that after a failure of working route of anycast connection (downstream or upstream), the anycast demand can be rerouted to another replica server. In the second approach – called backup path – the restoration procedure is the same as in unicast communication. A client is assigned permanently to one replica server and after a failure of the working route of anycast connection, the backup route connecting the same pair of nodes is established.

3 Problem Formulation

In this section we will formulate the offline optimization problem of unicast and anycast connections restoration. Notice that the consider problem is an enhanced version of the UFP (Unsplittable Flow Problem) – well known optimization problem of connection-oriented networks [9], [11]. The UFP is formulated as follows. We are given a directed network with arc capacities and a set of connections (requests) defined by the triple: origin node, destination node and bandwidth requirement. The objective is to find a subset of the connections of maximum total demand with additional constraints: each connection can use only one path and the sum of demands crossing the arc cannot exceed its capacity. The main novelty of our approach is that we consider joint optimization of unicast and anycast flows, while the classical UFP addresses only unicast flows. The anycast version of UFP was formulated in [18]. For the remainder of the paper we will refer to the considered problem as UCFP (Unsplittable uniCast and anyCast Flow Problem). To mathematically represent the problem we introduce the following notations.

Sets:

V - set of vertices representing the network nodes.
A - set of arcs representing network directed links.
P - set of connections in the network. A connection can be of two types: unicast and anycast.
P^{UN} - set of unicast connections in the network defined by a following triple: origin node, destination node and bandwidth requirement.
P^{AN} - set of anycast connections in the network. A connection can be of two types: upstream and downstream. Each anycast connection is defined by a following triple: client node, bandwidth requirement, index of associated connection.
P^{DS} - set of anycast downstream connections in the network.
P^{US} - set of anycast upstream connections in the network.
Π_{p} - the index set of candidate routes (paths) π_p^k for connection p. Route π_p^0 is a "null" route, i.e. it indicates that connection p is not established. If p is a unicast connection, the route π_p^k connects the origin and destination node of p. If p is an anycast upstream connection, route π_p^k connects the client node and the server. Finally, if p is an anycast downstream connection, candidate routes connect the server and the client node.
X - set of variables x_p^k, which are equal to one. X determines the unique set of currently selected routes.

Constants:

δ_{pa}^k - equal to1, if arc a belongs to route k realizing connection p; 0 otherwise
Q_p - volume (estimated bandwidth requirement) of connection p
c_a - capacity of arc a
$\tau(p)$ - index of the connection associated with connection p. If p is a downstream connection $\tau(p)$ must be an upstream connection and vice versa.
$o(\pi)$ - origin node of route π. If π is a "null" route, then $o(\pi)$=0.
$d(\pi)$ - destination node of route π. If π is a "null" route, then $d(\pi)$=0.

Variables:

x_p^k - 1 if route $k \in \Pi_p$ is selected for connection p and 0 otherwise.
f_a - flow of arc a.

The UCFP can be formulated as follows

$$LF = \min_X \sum_{p \in P} x_p^0 Q_p \tag{1}$$

subject to

$$\sum_{k \in \Pi_p} x_p^k = 1 \qquad \forall p \in P \tag{2}$$

$$x_p^k \in \{0,1\} \quad \forall p \in P, \forall k \in \Pi_p \tag{3}$$

$$f_a = \sum_{p \in P} \sum_{k \in \Pi_p} \delta_{pa}^k x_p^k Q_p \quad \forall a \in A \tag{4}$$

$$f_a \le c_a \quad \forall a \in A \tag{5}$$

$$\sum_{k \in \Pi_p} x_p^k d(\pi_p^k) = \sum_{k \in \Pi_{\tau(p)}} x_{\tau(p)}^k o(\pi_{\tau(p)}^k) \qquad \forall p \in P^{AN} \tag{6}$$

$$X = \left(\bigcup_{p,k:x_p^k=1} \left\{ x_p^k \right\} \right) \tag{7}$$

The objective function (1) is a lost flow (*LF*). Function *LF* is as a sum of all demands (connections) that are not established (variable x_p^0 is 1). It should be noted that also an equivalent objective function can be provided, in which we maximize the total volume of established connections. Condition (2) states that the each connection can use only one route or is not established. Therefore, we index the subscript of variable x_p^k starting from 0, i.e. variable x_p^0 indicates whether or not connection *p* is established. If it is established, x_p^0=0 and one of variables x_p^k (*k*>0), which is equal to 1, indicates the selected path. Constraint (3) ensures that decision variables are binary ones. (4) is a definition of an arc flow. Inequality (5) denotes the capacity constraint. Equation (6) guarantees that two routes associated with the same anycast demand connect the same pair of nodes. Notice that constraint (6) must be satisfied only for anycast connections. Moreover, (6) assures that if upstream connection is not established (x_p^0=0 and the “null” route is selected) the downstream connection of the same anycast demand is not established, and vice versa. Finally, (7) is a definition of a set *X* called a selection that includes all variables *x*, which are equal to 1. Each selection denotes for each connection either the selected route or indicates that the particular connection is not established. Note, that we call a connection *p established* in selection *X* if $x_p^0 \notin X$.

4 Algorithm

Since of the objective of network restoration is relatively short decision time, we propose to solve the UCFP problem by a simple *greedy algorithm* (GA). The unicast version of GA proceeds all connections in a one pass and either allocates the processed request to the shortest path or rejects the request if such a feasible path does not exist, i.e. origin and destination node of the connection do not belong to the same component of considered graph [9]. It should be noted that modification of GA called *bounded greedy algorithm* (BGA) and *careful BGA* (cBGA) were proposed in [9], [11]. Online algorithms can also solve the UFP. Several such algorithms were developed in the context of dynamic routing in MPLS networks [3], [10]. MPLS supports the explicit mode, which enables the source node of the LSP to calculate the path. The main goal of dynamic routing is to minimize the number of rejected calls or to minimize the volume of rejected calls. The most common approach to dynamic routing is the shortest path first (SPF) algorithm based on an administrative weight (metric). In [18] we proposed a GA for anycast flows.

Now, we present how to modify unicast greedy algorithm to enable joint optimization of anycast and unicast flows. We refer to this new algorithm as CGA (uniCast and anyCast GA). CGA analyzes all requests (unicast and anycast) in a one pass. Requests can be sorted accordingly to a selected criterion (e.g. bandwidth requirement). We apply some temporary variables in both algorithms. Set H is a selection including decision variables x equal to 1. Set B includes indices of connections. Operator $first(B)$ returns the index of the first connection in set B. Operator $sort(H)$ returns indexes of connections included in H ordered according to their paths' length given by the metric CSPF [3] starting with the longest. We select CSPF metric due to its effectiveness in many dynamic routing problems.

Operator $USP(H,i)$ returns either the index of the shortest path calculated according to selected metric or 0, if a feasible route does not exist for connection i. We apply Dijkstra's algorithm, however any Shortest Path First algorithm can be used. To facilitate the process we construct a residual network, i.e. we remove from the network all arcs that have less residual capacity than volume of connection i. If in the residual network none feasible path exists, the demand is rejected because there are not enough resources of residual capacity to establish connection i.

Operator $ASP(H,i,j)$ returns either the pair of indices of shortest paths for downstream and upstream connection or a pair of zeros, if a pair of feasible routes does not exist for connections i and j associated with the same anycast demand. To find a pair of shortest paths of downstream and upstream connections all replica servers are taken into account. In particular, the residual network is constructed in analogous way as shown above in the context of *USP* operator. Then, for each server node the shortest path of downstream connection and the shortest path of upstream connection are calculated applying Dijkstra's algorithm. Next, the sum of these two paths' lengths is assigned to the considered server node. Finally, we select a server and a pair of paths for which the aggregate length is the smallest.

Algorithm CGA

Step 1. Let H denote an initial solution, in which none connection is established. Let $B:=sort(P^{UN}\cup P^{DS})$.

Step 2. Set $i = first(B)$ and calculate the metric of each arc $a\in A$.

If $i\in P^{UN}$, find the shortest route of connection i according to selected metric $k = USP(H,i)$. Set $H = \left(H - \left\{x_i^0\right\}\right)\cup\left\{x_i^k\right\}$. Go to step 3.

If $i \in P^{AN}$, find the pair of shortest routes of connections i and $\tau(i)$ according to selected metric $\{d,u\} = ASP(H,i,\tau(i))$. Set $H = \left(H - \left\{x_i^0\right\}\right) \cup \left\{x_i^d\right\}$ and $H = \left(H - \left\{x_{\tau(i)}^0\right\}\right) \cup \left\{x_{\tau(i)}^u\right\}$. Go to step 3.

Step 3. Set $B = B - \{i\}$. If $B = \varnothing$, then stop the algorithm. Otherwise go to step 2.

The CGA algorithm is a modification of the classical greedy algorithm developed for unicast flows. Complexity of the algorithm depends on the number of connections. The most time consuming operations is calculation of shortest path in operator *USP* and *ASP*. Therefore, algorithm is relatively simple. This is motivated by the fact that the restoration process must be performed robustly and quickly. Therefore, relatively low complexity of the algorithm can enable the application of CGA algorithm in online restoration.

5 Results

Algorithm CGA was coded in C++. The network on which we conduct our experiment consists of 36 nodes and 144 directed links (Fig. 1). The bold lines represent links of size 96 units while other lines are links of size 48 units. The link capacities were chosen to model capacity ratio of OC-48 circuits. During simulations, the link capacities were scaled by a factor 100 to enable establishing of thousands of connections. Since we consider an existing backbone network, which is in an operational phase and the number of replica servers is constant. However, to verify performance of our approach for various cases we test 12 various servers location scenarios (Table 1). The number of servers varies from 2 to 4. For each of server location we test 13 different demand patterns consisting of 180 anycast demands (360 anycast connections) and 2500 unicast connections. In the simulation we use the anycast ratio (AR) parameter defined as the ratio of the anycast demands' bandwidth requirement to the bandwidth requirement of all demands issued in the network. Demand patterns are generated randomly with various values of AR.

The scenario of numerical experiments is as follows. First we calculate working routes of all connections (anycast and unicast) applying a modified version of algorithm FD_RCL proposed in [19]. The FD_RCL algorithm is based on the flow deviation method applied to many optimization problems [8], [13], [15]. However, it applies a new objective function RCL defined in [19]. According to results presented in [19], the FD_RCL algorithm provides the best assignment of working routes in terms of network survivability comparing to other tested heuristics. The modification of the algorithm consists in joint optimization of unicast and anycast connections similarly to the algorithm published in [17]. Next, we simulate a failure of each link and perform the network restoration process. According to [4] the single link cut is the most probable failure in modern networks. Apparently, also other failure scenarios (node failures, multiple failures) could be considered, however we focus on the most common case. To run the restoration of anycast flows we can use the CGA algorithm i.e. the backup replica method. For comparison we implement backup path restoration method, which applies the greedy algorithm deployed for unicast flows. For easy of reference we will call this algorithm as UGA (Unicast Greedy Algorithm). The UGA algorithm does not use the anycast communication. Unicast flows are restored using algorithm UGA - the traditional greedy algorithm. For each link we repeat the same procedure, i.e. the link is pruned and the restoration of all connection that traversed the failed link is made to obtain the amount of un-restored lost flow. Finally, we compute the aggregate lost flow, which is a sum over all links failures. Note that both kinds of flows: unicast and anycast are considered in the calculation of lost flow. In the case of anycast request, if one of anycast connections (downstream or upstream) is broken and not restored, we add to the lost flow bandwidth requirement of both connections.

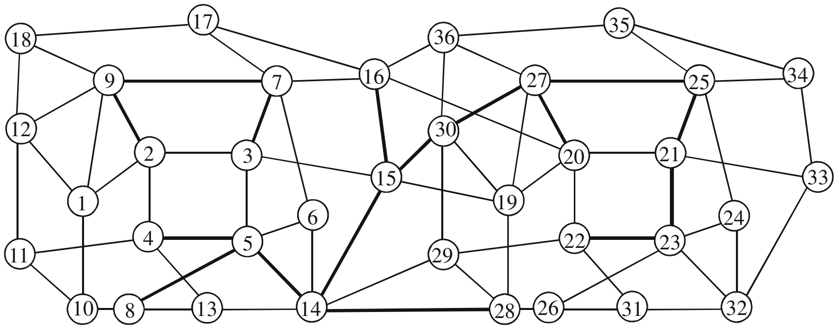


Fig. 1. Topology of test network

Table 1. Location of replica servers – simulation cases

Number of servers	Location of servers	Number of servers	Location of servers	Number of servers	Location of servers
2	5, 23	3	5, 23, 30	4	5, 9, 23, 30
2	5, 25	3	5, 25, 30	4	5, 9, 25, 30
2	7, 23	3	7, 14, 23	4	7, 14, 23, 30
2	7, 25	3	7, 14, 25	4	7, 14, 25, 30

To present the results we define the AVLU (Average Link Utilization) parameter as the average network saturation, i.e. AVLU is the sum of all links flow divided by the sum of all links capacity. The percentage difference between backup replica method and backup path method we calculate in the following way

$$diff = (res\_bp - res\_br) / res\_bp \tag{8}$$

where *res_bp* denotes the aggregate lost flow of backup path restoration and *res_br* denotes aggregate lost flow of backup replica restoration. Notice that the value of *diff* greater than 0 means that the backup replica restoration yielded lower value of lost flow than the backup path restoration.

On Fig 2 we report the average percentage difference between backup path and backup replica restoration as a function of server number. Fig. 3 presents detailed results of the difference between both methods as a function of the anycast ratio.

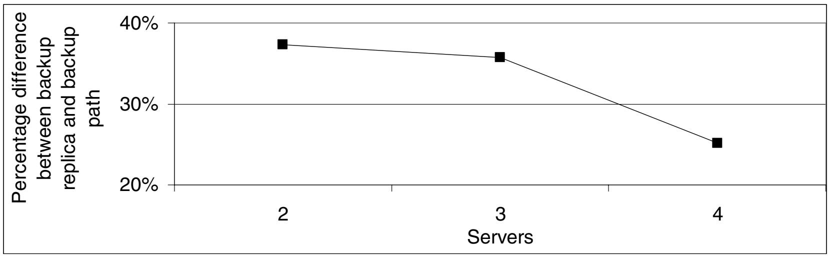


Fig. 2. Average percentage difference between backup replica and backup path restoration methods as a function of server number

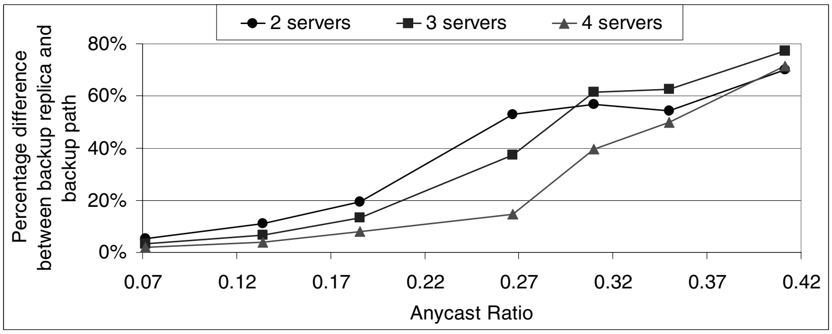


Fig. 3. Average percentage difference between backup replica and backup path restoration methods as a function of anycast ratio for various number of servers

Figs. 4-6 show the percentage difference between backup replica and backup path restoration methods as a function of the anycast ratio for various locations of 2, 3 and 4 servers, respectively.

From Figs. 2-6, we observe that applying the backup replica restoration method yields significant improvement comparing to the backup path restoration. The average gap between two methods overall all 156 tests is almost 33%. In particular, the analysis of results shows that the percentage difference between both methods decreases with the number of servers. Note that less servers in the network means longer – in terms of hops – paths for anycast connections. Moreover, these relatively few servers are more congested. In the case of a link failure, especially close to the server, other links adjacent to the server can become the bottleneck of the restoration. Recall that in the backup path restoration anycast connections are assigned permanently to one replica server, while the backup replica restoration can select another replica server. Consequently, when the number of servers increases, the difference between two restoration approaches decreases.

Another important observation is that the gap between backup replica and backup path methods increases with the anycast ratio. It is evident that more anycast traffic in the network implies more congested links adjacent to replica servers. As noted above, the backup replica outperforms the backup path method particularly in such cases.

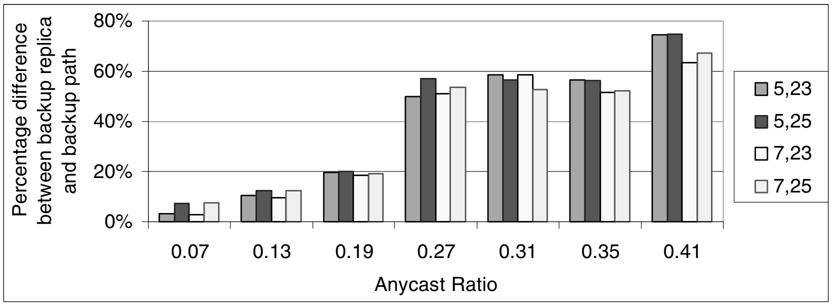


Fig. 4. Percentage difference between backup replica and backup path restoration methods as a function of anycast ratio for various locations of 2 servers

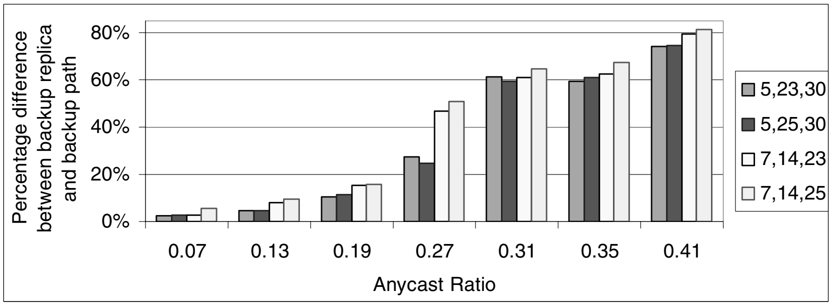


Fig. 5. Percentage difference between backup replica and backup path restoration methods as a function of anycast ratio for various locations of 3 servers

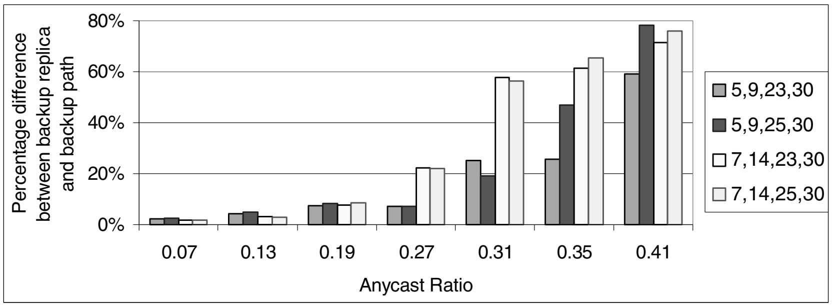


Fig. 6. Percentage difference between backup replica and backup path restoration methods as a function of anycast ratio for various locations of 4 servers

Results presented on Figs. 4-6 suggest additionally that the gap between analyzed methods depends on the location of network servers, however the differences are not significant.

Fig. 7 shows how the number of servers influences the amount of lost flow. Each bar in the graph represents one demand pattern. The x-axis is the anycast ratio and the y-axis is the normalized lost flow calculated in the following way. The same demand pattern was run using the backup replica restoration for 2, 3 and 4 servers in the network. Next, obtained values of lost flow were normalized proportionally to make 100% in sum. We can observe that with the increase of anycast ratio, relatively less flow is lost when the number of servers grows.

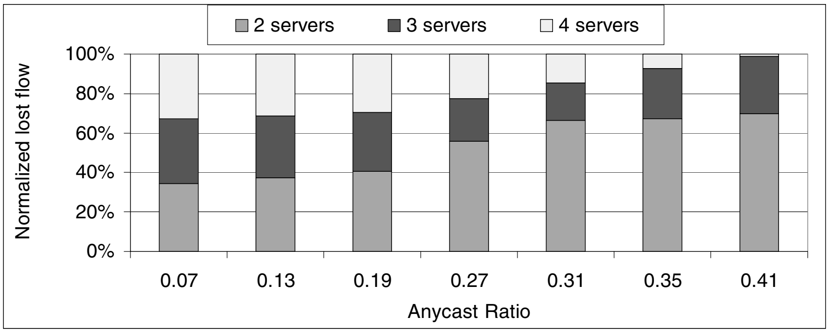


Fig. 7. Normalized lost flow of backup replica as a function of anycast ratio

6 Conclusion

In this paper, we have proposed to apply anycast communication in order to improve network survivability. We have formulated a new optimization problem UCFP, which is equivalent to the joint restoration of unicast and anycast flows in connection-oriented networks. The UCFP problem is motivated by service providers' needs for fast deployment of bandwidth guaranteed services enabling fast and effective distribution of popular content over Internet. A simple heuristic algorithm CGA has been presented to solve the UCFP problem. The CGA algorithm enables a new restoration approach called backup replica. Next, we run extensive simulations to evaluate our approach against traditional unicast restoration. The experiment confirms that the backup replica restoration method is more efficient then the backup path method developed for restoration of unicast flows. The gap between both methods increases with the ratio of anycast traffic in the network. Results of our work can be applied for optimization of Content Delivery Networks located in connection-oriented environment, e.g. MPLS network.

Acknowledgements. This work was supported by a research project of the Polish State Committee for Scientific Research carried out in years 2005-2007.

References

1. Awerbuch, B., Brinkmann, A., Scheideler, C.: Anycasting in adversarial systems: routing and admission control. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1153–1168. Springer, Heidelberg (2003)
2. Ballani, H., Francis, P.: Towards a Global IP Anycast Service. In: Proc. of SIGCOMM'05, Philadelphia, pp. 301–312 (2005)

3. Crawley, E., Nair, R., Jajagopalan, B., Sandick, H.: A Framework for QoS-based Routing in the Internet. RFC2386 (1998)
4. Grover, W.: Mesh-based Survivable Networks: Options and Strategies for Optical, MPLS, SONET and ATM Networking. Prentice Hall PTR, Upper Saddle River, New Jersey (2004)
5. Doi, S., Ata, S., Kitamura, K., Murata, M.: IPv6 anycast for simple and effective service-oriented communications. IEEE Communication Magazine 5, 163–171 (2004)
6. Hao, F., Zegura, E., Ammar, M.: QoS routing for anycast communications: motivation and an architecture for DiffServ networks. IEEE Communication Magazine 6, 48–56 (2002)
7. Hou, Y.T., Shi, Y., Sherali, H.D.: Optimal base station selection for anycast routing in wireless sensor networks. IEEE Transactions on Vehicular Technology 3, 813–821 (2006)
8. Kasprzak, A.: Designing of Wide Area Networks. Wroclaw Univ. of Tech. Press (2001)
9. Kleinberg, J.: Approximation algorithms for disjoint paths problems. PhD thesis, MIT, Cambridge (1996)
10. Kodialam, M., Lakshman, T.: Minimum Interference Routing with Applications to MPLS Traffic Engineering. In: Proc. of INFOCOM, pp. 884–893 (2000)
11. Kolman, P., Scheideler, C.: Improved bounds for the unsplittable flow problem. In: Proc. of the Symposium on Discrete Algorithms, pp. 184–193 (2002)
12. Lin, C., Lo, J., Kuo, S.: Load-Balanced Anycast Routing. In: Proc, of the Parallel and Distributed Systems, Tenth international Conference on (Icpads'04), Washington (2004)
13. Markowski, M., Kasprzak, A.: The web replica allocation and topology assignment problem in wide area networks: algorithms and computational results. In: Gervasi, O., Gavrilova, M., Kumar, V., Laganà, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3483, pp. 772–781. Springer, Heidelberg (2005)
14. Peng, G.: CDN: Content Distribution Network. Technical Report (2003),sunysb.edu/tr/rpe13.ps.gz
15. Pióro, M., Medhi, D.: Routing, Flow, and Capacity Design in Communication and Computer Networks. Morgan Kaufman Publishers, San Francisco (2004)
16. Walkowiak, K.: Designing of survivable web caching. In: Proc. of 8th Polish Teletraffic Symposium, pp. 171–181 (2001)
17. Walkowiak, K.: A Unified Approach to Survivability of Connection-Oriented Networks. In: Yolum, p., Güngör, T., Gürgen, F., Özturan, C. (eds.) ISCIS 2005. LNCS, vol. 3733, pp. 3–12. Springer, Heidelberg (2005)
18. Walkowiak, K.: Unsplittable Anycast Flow Problem Formulation and Algorithms. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J.J. (eds.) ICCS 2006. LNCS, vol. 3991, pp. 626–633. Springer, Heidelberg (2006)
19. Walkowiak, K.: A New Function for Optimization of Working Paths in Survivable MPLS Networks. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 424–433. Springer, Heidelberg (2006)
20. Walkowiak, K.: Survivable Routing of Unicast and Anycast Flows in MPLS Networks. In: Proc. 3rd EURO-NGI Conference on Next Generation Internet Networks, pp. 72–79 (2007)

Privacy Preserving Context Transfer in All-IP Networks

Giorgos Karopoulos, Georgios Kambourakis, and Stefanos Gritzalis

Department of Information and Communication Systems Engineering
University of the Aegean, Karlovassi, GR-83200 Samos, Greece
{gkar,gkamb,sgritz}@aegean.gr

Abstract. In an all-IP environment, the concept of context transfer is used to provide seamless secure handovers between different administrative domains. However, the utilization of context transfer arises some privacy issues concerning the location and movement of users roaming between domains. In this paper we elaborate on these privacy issues and propose an alternative context transfer protocol that protects user's location privacy as well. In addition, assuming that the context carries a user identity in the form of a Network Access Identifier (NAI), we show how the employment of temporary NAIs can further increase the privacy of our scheme.

Keywords: Privacy, Context Transfer, NAI, all-IP networks, secure handover.

1 Introduction

Today, the uninterrupted continuation of the received services during handover between networks with different access technologies still remains an open issue. In order to have fast, secure handovers in such an all-IP terrain new methods were recently proposed, like OIRPMSA [1], MPA [2] and Context Transfer [3]. As discussed in [4], while these methods do succeed in minimizing the disruption caused by security related delays, it seems that little has been done to protect the end users privacy as well.

Whereas a lot of work has been done in privacy and location privacy in general, the authors are not aware of any previous work preserving location privacy in methods offering fast secure handovers in all-IP based networks. In this work we focus on the Context Transfer solution. We discuss and highlight the privacy issues arising from the employment of the Context Transfer Protocol (CTP) [3] and propose a solution towards solving these problems. We further extent our solution based on the observation that the NAI [5] is a suitable type of identity for networks that span across multiple administration domains. Since this applies to our case we use temporary NAIs as context's identity in order to increase the level of user's privacy. The result of our work is that the decision for user's identity and location disclosure is no longer left to the good will and intensions of the visiting networks and the user is not forced to trust the foreign domains but only his home domain with which he has signed a contract. The rest of this paper is structured as follows. In Section 2, some privacy issues are pointed out from the current functioning of the CTP. Section 3 presents the proposed solution to these privacy issues based on two concepts: Mobile Node (MN) submitted context and frequent NAI change. Section 4 provides a discussion about prerequisites and deployment issues for our protocol. Last section offers concluding thoughts and future directions for this work.

2 The Problem: Privacy Issues in Context Transfer Protocol

The way the CTP operates, as defined in the RFC 4067, arises some privacy issues. These issues concern primarily the end user and more specifically his location and movement between different administrative domains. The first observation has to do with the inner workings of the protocol itself. Every time a handover occurs, the previous Access Router (pAR) uses the CTP to send various context data blocks to the new Access Router (nAR). That is, for every handover the pAR and the nAR know where the user came from and where he is going. When these two ARs belong to the same administrative domain it goes without saying that the domain is aware of the movement of the MN inside its own network. However, when the two ARs belong to different administrative domains there is no reason for the pAR to know which the nAR is and the opposite. To sum up, with the use of the CTP for seamless handovers, every administrative domain is aware of the previous and the next administrative domain of the MN, without excluding itself. This means that every domain can track a part of the user's movement. Continuing from the last conclusion, the user's movement can be completely tracked, given that some administrative domains collude. Note, that this does not imply that all administrative domains in the path of the user movement are required to collude for such an attack, but every second domain in that path.

Another aspect of the location privacy problem when the CTP is in place is the type of the identifier used by the user/MN during the protocol negotiation to authenticate to the new administrative domain. The utilization of a static identifier like the MAC address of the MN or a globally used username of the user simplifies the work of a malicious passive observer. An obvious choice for all-IP networks that belong to different administrative domains is the use of a NAI. However, if the administrative domains collude, they can track the whole movement of the user only by the observation of the use of this static NAI. Furthermore, even when administrative domains do not collude there can be a location privacy breach, since every single domain can recognize an old user that returns to it. It is thus, more than obvious, that systems' logistic files can be anytime processed to disclose information about the whole history of movements of a specific user.

3 The Proposed Solution

The proposed solution protects the location privacy of users roaming between different administrative domains utilising the CTP to receive uninterruptible services during handover. Our solution is twofold and it is proposed that: (a) the context should be submitted by the MN, and (b) there should be a frequent NAI change. The basic idea behind our scheme is that the user's sensitive information should only be known to the user himself and his home domain and no-one else, including the visiting domains. This is very important since the user has agreed and signed only one subscription contract; with his home domain.

3.1 Mobile Node Submitted Context

As it is stated in RFC 4067, the context is transferred between layer-3 entities from the old network domain to the new network domain. This way, a part of the MN user's route can be tracked. As already stated this is the case of a single domain tracking the movement of the user; if domains collude, then the full movement of the user can be tracked simply by using the information revealed by the CTP. One possible solution to avoid such problems is to have the MN submitting its own context to the network it is moving to. The complete abstract protocol steps are as follows: *Step 1* - The MN establishes a secure session with the AR of the new domain. This secure session must have the following properties: (a) it must be encrypted and (b) the AR must be authenticated to the MN. *Step 2* - The MN sends the context over the previously established protected channel. *Step 3* - The AR authenticates the MN and re-establishes the services based on the context. It is also assumed that the current domain has established some kind of trust relationships beforehand with the home domain. This way the authentication is processed locally based on an authentication token located in the context, which is digitally signed by the home domain.

The above procedure is the equivalent of a PEAP [6] or an EAP-TTLS [7] authentication and key establishment method using the context as user authentication means. The first phase of the PEAP or EAP-TTLS method is followed as is, e.g. a secure session is established with the use of the digital certificate of the AR. In the second stage the authentication of the user is taking place with the utilization of the credentials contained in the context. The key establishment phase could also be benefited by the context transfer since the context can contain security parameters i.e. cryptographic keys, supported suites, tokens, etc. The proposed method can be used in either a reactive or proactive scenario. In cases where a high QoS must be preserved, the aforementioned procedure could be executed proactively, that is before the MN actually moves to the new administrative domain. This situation is comparable to the pre-authentication procedure exercised in IEEE 802.11 or 802.16 networks. An example of a context transmitted by the MN is shown in Fig. 1. When the MN moves towards P3 the handover procedure starts. The MN establishes a secure channel with the nAR and through this channel transfers the context. As it can be easily noticed, the ARs do not play any role in the context transfer procedure and there is no communication between them. Also, they are not aware of each other in any way. One potential drawback of our method is the possible degradation of service during the handover process; however, this is left to be proved in a future work. The factors that lead to this are the use of asymmetric cryptography and the increased number of messages during the whole procedure.

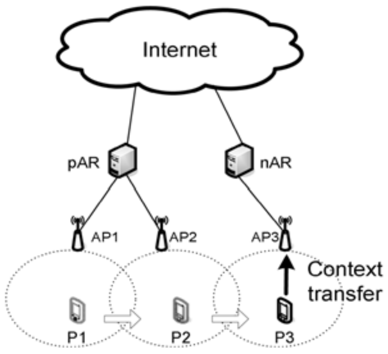


Fig. 1. MN submitted context

3.2 Frequent NAI Change

In this heterogeneous environment one way to identify the users is the use of NAI. Of course, the NAI can also be utilised in conjunction with the CTP. When the NAI concept is employed in the proposed way (MN submits the context) then the current domain or some colluding domains still can track the location of the user simply by observing the transmission of NAIs. More specifically, the current domain can always be aware when a single user was present in its network or when a user returns to it. When the domains collude things get worse since they can observe the exact route of a single user. The solution is based on the use of temporary NAIs and the frequent change of them. Thus: (a) The home domain is the only one that has the correspondence between the true identity of the user and the NAI assigned to him, (b) when a context is created for the user, it contains a temporary NAI. This temporary NAI uses as user_id a random unused string, which the home domain connects with the true identity of the user, and as domain_id the assigned domain_id. Each temporary user_id is used once for every single domain by one user at a time. When the user handovers to another domain (either new or previously visited) he must use a different user_id. The reuse of a temporary user_id by another user is not forbidden since the home domain is also aware of the date and time each user is using it. Therefore, the only sensitive information about the user that is revealed to foreign domains is the home domain of the user, and (c) after the completion of the handover of the MN to a new domain, the MN is using a secure channel (like a TTLS session) to contact its home domain and obtain a new temporary NAI. This way, when the user returns to a previous visited domain, the domain cannot recognize him.

Even if the correspondence between the true identity of the user and his NAI or any temporary NAI is revealed by accident or other reason, the user's past routes cannot be revealed without the help of his home domain. The obvious drawback of this method is the increase in the signaling between the domains. However, this is done after the completion of the handover and therefore has no real effect in the QoS perceived by the user during the handover. In Fig. 2 a message sequence diagram of the overall proposed solution is presented. The MN has an existing session with the pAR; when it wants to handover to the nAR it first establishes (proactively or reactively) a secure session with it. Then, through this secure session, it transfers the context that will allow the MN to authenticate, establish session keys and re-establish the services it already uses. When the handover procedure is finished, the MN should contact its home domain in order to obtain some new credentials (for example a new temporary NAI) that will be used in its next handover.

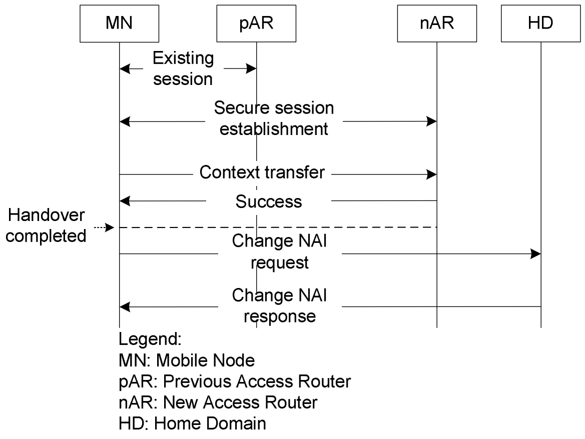


Fig. 2. Message sequence of our solution

4 Discussion

From the trust requirements point of view, the proposed solution has some prerequisites that are analogous to those of CTP. More specifically, CTP requires that trust relationships exist among the ARs and between the MN and each of the ARs (pAR and nAR). In our case, each AR should have trust relationships with the home domain of the roaming MN; since the MN also has trust relationships with its home domain, new trust relationships between the MN and each AR can be established on-the-fly. An important factor concerning the wide deployment of a protocol is the number of changes required in the already installed infrastructure. Taken into account the situation as it is today, our protocol requires a reasonable number of such changes which are comparable to those required for the deployment of the CTP. More specifically, in CTP the ARs should be able to transfer the context among them and interpret the contents of the context; the MN should also implement the CTP in order to be able to request the transfer of the context. In our proposal the ARs should only be able to interpret the contents of the context while the MN should be able to handle the context which it possesses according to the proposed protocol.

Another point of consideration is the protection of the context itself. Since in the proposed protocol the context is carried by the MN, actions must be taken so that the context cannot be altered by the user unnoticed. This implies that there should be a kind of digital signature in place ensuring the integrity of the transmitted context. The encryption of the context while stored in the MN is not a strict requirement since the information contained in it is already known to the user. However, having in mind that the MN is a portable device and thus it is easy to get lost or stolen, some care to prevent tampering, unauthorized use, or fraud could be taken. One final remark about the context is its expiration. The time interval of expiration should be neither too large, containing expired information, nor too small, causing excessive signaling among the administrative domains. What is obvious for our protocol is that when the MN moves to a new domain the context is renewed since a new temporary NAI is requested. In any case, the expiration interval can be set by the network administrators and the current point of attachment (some AR) of the MN can warn it that its context has expired or is about to expire.

5 Conclusions

We have presented a novel solution that preserves user's location privacy when using the CTP which is currently employed by the state of the art methods for seamless secure handovers between different administrative domains. We showed that the standard way the protocol behaves arises some privacy issues and proposed an alternative protocol that alleviates these problems. Moreover, we have proposed how the use of the context in conjunction with a NAI can further enhance user's privacy. Part of our future work is to measure the delays incurred by our protocol. Preliminary analysis discloses that these times are expected to be tolerable with medium-end devices, achieving seamless handovers even to very demanding applications.

References

1. Xu, P., Liao, J., Wen, X., Zhu, X.: Optimized Integrated Registration Procedure of Mobile IP and SIP with AAA Operations. In: Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA), pp. 926–931 (2006)
2. Dutta, A., Fajardo, V., Ohba, Y., Taniuchi, K., Schulzrinne, H.: A Framework of Media-Independent Pre-Authentication (MPA). IETF Internet Draft, draft-ohba-mobopts-mpa-framework-03, work in progress (2006)
3. Loughney, J., Nahkjiri, M., Perkins, C., Koodli, R.: Context Transfer Protocol. RFC 4067 (2005)
4. Karopoulos, G., Kambourakis, G., Gritzalis, S.: Survey of Secure Handoff Optimization Schemes for Multimedia Services Over All-IP Wireless Heterogeneous Networks. IEEE Communications Surveys and Tutorials (to appear)
5. Aboba, B., Beadles, M., Arkko, J., Eronen, P.: The Network Access Identifier. RFC 4282 (2005)
6. Palekar, A., Simon, D., Salowey, J., Zhou, H., Zorn, G., Josefsson, S.: Protected EAP Protocol (PEAP) Version 2. IETF Internet Draft, draft-josefsson-pppext-eap-tls-eap-10, expired (2004)
7. Funk, P., Blake-Wilson, S.: EAP Tunneled TLS Authentication Protocol (EAP-TTLS). IETF Internet Draft, draft-ietf-pppext-eap-ttls-01, expired (2002)

Environment-Aware Trusted Data Delivery in Multipath Wireless Protocols*

Mohit Virendra[1], Arunn Krishnamurthy[1], Krishnan Narayanan[1], Shambhu Upadhyaya[1], and Kevin Kwiat[2]

[1] Computer Science & Eng., State University of New York at Buffalo, Buffalo, NY USA
{virendra,ack6,kn38,shambhu}@cse.buffalo.edu
[2] US Air Force Research Laboratory, 525 Brooks Road, Rome, NY
kwiatk@rl.af.mil

Abstract. Current multipath protocols for Multi-Hop Wireless Networks (MWNs) use hop-count as the default route selection criteria. Route selection should also consider network and link conditions. We propose a network-environment-aware trust-based route selection framework for MWNs that makes informed and adaptive route-selection decisions. A node quantifies trust values for its neighboring nodes and for the routes that pass through it. The trust metric adjusts to varying network conditions and quick convergence of the protocol implies it works well in mobility scenarios. Glomosim simulations demonstrate throughput improvement over conventional multipath protocols under congestion, link failure and route unreliability scenarios.

Keywords: AOMDV, Multipath, Routing, Security, Trust.

1 Introduction

Ad-hoc On-demand Multipath Distance Vector (AOMDV) [1] and AODVM [3] routing are multipath variants of the well-known AODV protocol [2] for Mobile Multi-Hop Wireless Networks (MWNs). Route selection may be affected by congestion, presence of selfish (or malicious) nodes, and other adverse network or physical conditions. Additional route information may enhance probability of packets reaching destination. In existing multipath protocols, if nodes could evaluate confidence on available routes (estimate route reliability due to physical and network conditions), make trusted route selection decisions, and dynamically switch traffic across different available routes, data delivery robustness would be enhanced.

This paper presents a framework for MWN nodes to evaluate route conditions and provides metrics to make informed multipath routing decisions. Our goal is to quickly detect any *effects* on data transfer, attributable to malicious nodes or other adverse conditions in a route, and to take corrective actions (e.g., use alternate routes). In our model, a node quantifies trust values for its neighboring nodes and for the routes that pass through it. The trust metrics adjust to varying network conditions; quick protocol convergence implies it works well in mobility scenarios.

* Research supported in part by U.S. Air Force Research Laboratory Grant No. 200821J.

Recent trust models for node dependability, reliability and security in P2P systems are summarized by Li et al. [5]. Expensive peer-node promiscuous monitoring for behavior assessment, significant in all existing models [9, 6, 7, 8], is minimized in our framework by enhancing node accountability for data forwarding cooperation. Overall route-performance rather than individual node-misbehavior detection is our focus. No extra control overhead for trust computation is introduced; convergence time for our framework is the same as that of AOMDV. Since it assumes presence of cryptographic protocols, information will not be compromised when our protocol is executing in the presence of malicious nodes. We do not distinguish between packets dropped due to malicious or selfish behavior and due to congestion. All these are inhibitive towards efficient data transfer and worthy of loss of trust. Nodes have unique non-forgeable IDs. Links are bidirectional; link costs/capacities maybe directionally different and are estimated through well known techniques (e.g., [10]). Trust is non-transitive, i.e., $Txy \neq Tyx$. Downstream is towards destination and upstream is towards source. Source and destination nodes are assumed to be non-malicious. Trust is computed on a continuous scale of 0 to 1.

2 Technique

In AODV, the source node broadcasts a *Route Request* (RREQ) packet which is in turn re-broadcasted by the nodes' neighbors until the sought route is discovered. Upon receiving an RREQ, the destination node or an intermediate node with a 'fresh enough' route to the destination, unicasts a *Route Reply* (RREP) packet back to the source node. AODV also uses *Route Error* (RERR) and *Route Reply Acknowledgement* (RREP-ACK) control packets for route management. AOMDV discovers link-disjoint or node-disjoint multi-paths between pairs of nodes.

In our model a node maintains two trust values, one for routes passing through it and another for its one-hop neighbors – *Route Trust:* measure of reliability of packets reaching the destination if forwarded on a particular route, computed by each node for all routes in routing table; *Node Trust:* measure of confidence on one-hop neighbors that they accurately assess and report downstream route conditions. Node trust is initialized at 1 for destination and 0.5 for all other participating nodes. Initial route trust is formalized by Effective Link Capacity (ELC) and Effective Route Capacity (ERC) values. ELC is an indicator of the traffic that can be scheduled on the link for a particular route/flow. ERC is the *effective* capacity of the route from an intermediate node to the destination. It factors in the ELCs computed at each intermediate node. An ERC value corresponding to a route at a node can thus be analogous to that node's trust on that route downstream. Subsequent updates to node and route trusts are interdependent. Route trust is recursively computed by each node starting at the destination and moving upstream, taking care of route divergences and convergences. At each hop, a node's assessment of its downstream reporting neighbor (i.e., node trust) is factored into the route trust. In turn, the difference between predicted route trust and the eventual route performance governs an upstream node's trust (node trust) on its downstream neighbor. Thus nodes are accountable for providing an accurate assessment of route conditions.

Additions/modifications to AOMDV are made for piggy-backing ELC and ERC values in the RREP packets upstream and maintaining trust details for all routes in each node's routing table. The details are as follows:

- Each node maintains an additional data structure called the Neighbors' Trust Table. It contains neighboring node IDs, and corresponding node trust values.
- RREP packets have an additional route trust field and routing tables have a route trust entry for every destination as well. Upon receiving an RREP, a node caches the route trust sent by the downstream node. The node then reevaluates its own trust on the route downstream, updates the corresponding route trust entry in its routing table, updates the route trust field in the RREP packet and forwards it upstream.
- A *Type* field in the packet header (values 0-3) in AOMDV identifies the control packet type (RREQ, RREP, RERR and RREP-ACK). We introduce two new control packets, the *Query* (QRY: *Type* value 4) and the *Query-Acknowledgement* (QRY-ACK: *Type* value 5) packets. Route tables have a *Query Flag* bit set when a QRY-ACK is expected in response to a QRY. These packets contain encrypted checksum, computed over the entire packet by the packet creator, ensuring tamper-detection.

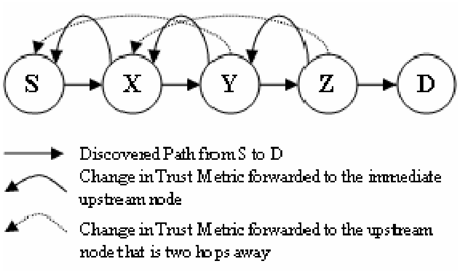

S
X
Y
Z
D
Discovered Path from S to D
Change in Trust Metric forwarded to the immediate upstream node
Change in Trust Metric forwarded to the upstream node that is two hops away

Fig. 1. Reporting of Route Trust Values

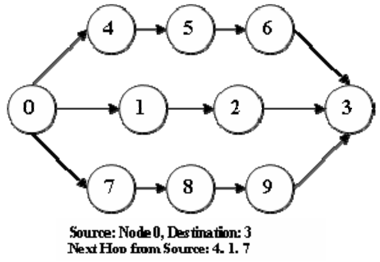


Fig. 2. Simulation Scenario

A node wanting to reassess its route trust whenever appropriate sends a QRY to the destination. The destination sends back a QRY-ACK containing the received packet count since transmission of the last QRY-ACK. The QRY initiator and the intermediate nodes forwarding the QRY-ACK re-compute route trusts using their ERC estimates and the ratio of data packets reaching the destination to data packets forwarded by them.

If multiple QRY or QRY-ACK packets are lost along a route, then the route trust would automatically decrease. We evaluate this in Sec. 3 through simulations. The node trust on the immediate downstream node is computed using the ratio of actual data rate achieved to the data rate promised by the downstream node.

Nodes recursively inform upstream neighbors of any changes in route trusts downstream, plugging in their own assessment of downstream-route-trust at each hop. Accuracy of such updates factors in re-evaluating node trusts on downstream neighbors in turn. For example, precise reporting of decreased route trusts due to congestion does not reduce node trust on the reporting downstream node. A Two-hop Reporting scheme employing AODV's *Localized Repair* feature is used for detecting bogus congestion reporting and silent discarding of QRY-ACK packets by malicious/selfish nodes. Assume that node *Y* (in Fig.1) is malicious. In the two-hop reporting scheme, *Z* sends

QRY-ACK to both *Y* and *X* using the localized repair feature. Node *X* should thus receive two copies of the packet which it can compare.

If node *Y* claims route-congestion for a time more than a threshold or when there is ambiguity between reports sent by the two downstream nodes, then all the routes with next hop Y are invalidated and purged from the routing table and RERR messages are sent to the destination. The node trust on *Y* would be made 0.

3 Performance Evaluation and Discussion

Packet Delivery Ratio and Trust Convergence latency were evaluated through Glomosim-2.02 simulations using the topology of Fig. 2 over a field size of 100m X 100m. This is an enhanced version of the network topologies that were used by Das et al. [1] and Yuan et al. [11]. The simulation was run for 150 seconds. Each node has a transmission range of 30 Meters using a Free-Space propagation-path loss model. Constant Bit Rate (CBR) traffic at 512 Bytes per second (bps) with an inter-departure time of 5 ms was injected between the source node 0 and the destination node 3 from the beginning till the end of the simulation.

To simulate general congestion in the network, we introduced an additional 1024 bps CBR traffic at the links [2, 3], [7, 8] and [4, 5] during the interval of 10-30S. Further, localized congestion was created through 2048 bps CBR traffic across the links [2, 3] and [4, 5] during 30-40S; across [2, 3] during 60-70S; across [7, 8] during 75-85S; and across [4, 5] during 85-99S. This emulated a variety of scenarios: simultaneous congestion on two routes, congestion on one route, different times the congestion eases, etc. Finally, nodes were failed during the following time intervals: Node 1:100-125S, Node 7:115-125S, Node 4:130-140S. This was done to study the adaptability of our protocol and achieve a fine grained comparison with AOMDV.

Route selection was weighted round robin. Source node reevaluated trust metrics by sending QRY packets at time intervals dictated by the already computed route trust values. For trust between 0.5-0.8, querying frequency was every 100 packets; for trust > 0.8, it was every 200 packets. The destination node sent back QRY-ACK packets to all the upstream nodes. Results were compared with native AOMDV. Trust values and Packet Delivery Ratio for each path (via nodes 1, 7 and 4) were evaluated. Results for path via Node 1 are reported; similar results were obtained for other paths.

As seen from the Figures 3a and 4, during the initial interval, 10-30S, local congestion simulated in all the available paths considerably affected the overall packet delivery ratio. As a consequence the route trust fluctuated during this time frame. Since there was no alternate path with better route trust, data packets were sent over all the paths and hence the overall protocol suffered due to this congestion. During the interval 60-70S, only the route via node 1 suffered congestion resulting in packet loss. The trust metric re-computation latency (time interval between the onset of congestion and the time at which the source obtains the QRY reply) was approximately 1.5 sec. This number was an averaged output of several test runs. Since the route trust on route via node 1 was greater than 0.8 before 60S, the QRY packets were sent out only after 200 data packets, and hence the trust convergence interval was large as indicated by pointer 1 in Fig. 3b. Once the congestion was realized at the source, the route trust on node 1 was decreased and the traffic was diverted through alternate paths via nodes 4 and 7. Thus, the route trust follows the packet delivery ratio computed using the QRY-ACK packet(s) from the destination. During this 60-70S interval, 5 data packets were sent through the congested route periodically to check if the congestion got cleared. Thus when the localized congestion between nodes 2 and 3 subsided after 70S, the source was able to reassess the trust within the next 0.5 sec. This was because of the reduced QRY request frequency that was set to be every 20 data packets. Thus, the trust convergence interval was less as indicated by pointer 2 in Fig. 3b. In this duration, traffic was redirected through alternate paths and hence the overall packet delivery ratio was not affected as can be seen from the overall packet delivery ratio in Fig. 4. Likewise, when the trust metrics were maintained in between 0.5-0.8, the trust convergence interval was approximately 1 sec. This could be visualized during the 30-31st sec in Fig. 3b. Similar localized congestion, reduction in route trust and diversion of data through highly trusted routes were monitored during 75-85S and 85-99S for the alternate paths and were found to show strict resemblance to the trust convergence latencies observed in Fig. 3b.

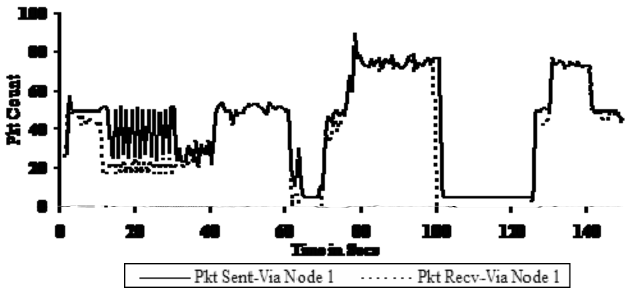


Fig. 3a. Packets Sent/Received Vs Time (For Next Hop Node 1)

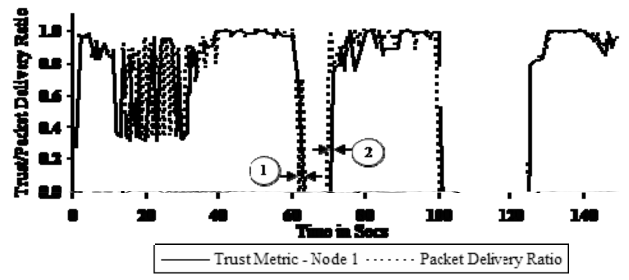


Fig. 3b. Trust & Packet Delivery Ratio Vs Time (For Next Hop Node 1)

The same simulation setup was also used to run two variants of AOMDV: round robin route selection and using a single route. Comparison of our scheme's overall packet delivery ratio with AOMDV variants (Fig. 4) shows that AOMDV (round robin) suffered approximately 50% throughput decline with downstream route congestion; single route AOMDV was even worse. Additionally, our protocol quickly sensed node failures and diverted traffic via alternate paths as against AOMDV that kept attempting to send traffic via routes with failed nodes. The results assure the

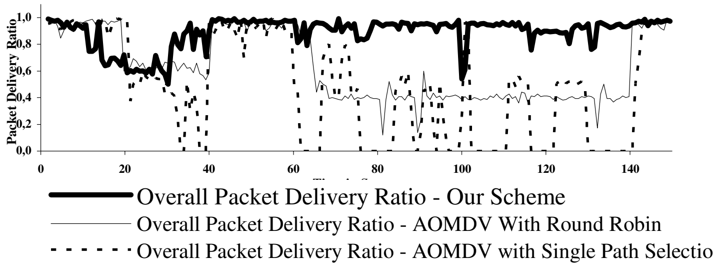


Fig. 4. Throughput Comparison

effectiveness of our proposal when adapted to multipath protocols. It is a self learning scheme which adapts to environment conditions.

References

1. Marina, M.K., Das, S.R.: Ad-hoc On-demand Multipath Distance Vector Routing. In: Proceedings of the IEEE International Conference on Network Protocols (ICNP), pp. 14–23. IEEE Computer Society Press, Los Alamitos (2001)
2. Perkins, C., Royer, E., Das, S.: Ad hoc On-Demand Distance Vector Routing. RFC-3651 (2003)
3. Ye, Z., Krishnamurthy, S., Tripathi, S.: A Framework for Reliable Routing in Mobile Ad Hoc Networks. In: Proceedings of the IEEE Conference on Computer Communications (INFOCOM), IEEE Computer Society Press, Los Alamitos (2003)
4. Zeng, X., Bagrodia, R., Gerla, M.: GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks. In: Proceedings of the 12th Workshop on Parallel and Distributed Simulations (1998)
5. Li, H., Singhal, M.: Trust Management in Distributed Systems. Computer 40, 45–53 (2007)
6. Sun, Y., Han, Z., Yu, W., Liu, K.J.: A Trust Evaluation Framework in Distributed Networks: Vulnerability Analysis and Defense against Attacks. In: Proceedings of the IEEE Conference on Computer Communications INFOCOM, IEEE Computer Society Press, Los Alamitos (2006)
7. Li, X., Lyu, M.R., Liu, J.: A Trust Model Based Routing Protocol for Secure Ad Hoc Networks. In: Proceedings of the IEEE Aerospace Conference (IEEEAC), pp. 1286–1295. IEEE Computer Society Press, Los Alamitos (2004)
8. Zouridaki, C., Mark, B., Hejmo, M., Thomas, R.: A Quantitative Trust Establishment Framework or Reliable Data Packet Delivery in MANETs. In: Proceedings of Workshop on Security of Ad-Hoc and Sensor Networks (SASN), Alexandria, VA (2005)
9. Marti, S., Giuli, T., Lai, K., Baker, M.: Mitigating Routing Misbehavior in Mobile Ad Hoc Network. In: Proceedings of 6th Annual Conference on Mobile Computing and Networking, pp. 255–265 (2000)
10. Li, J., Blake, C., De-Couto, D., Lee, H., Morris, R.: Capacity of Ad Hoc Wireless Networks. In: Proceedings of International Conference on Mobile Computing and Networking (MOBICOM) (2001)
11. Yuan, Y., Chen, H., Jia, M.: An Optimized Ad-hoc On-demand Multipath Distance Vector (AOMDV) Routing Protocol. In: Proceeding of the Asia-Pacific Conference on Communications, Australia (2005)

A Spatial Watermarking Algorithm for Video Images

Dumitru Dan Burdescu, Liana Stanescu, Anca Ion, and Cristian Marian Mihaescu

University of Craiova, Software Engineering Department,
Bvd. Decebal, 107, Craiova, Romania
{burdescu,stanescu_liana,soimu_anca,mihaescu}@software.ucv.ro

Abstract. The lack of control inherent to digital content has been put on the spotlight by copyright infringement coupled with massive content online distribution (e.g., Peer-to-Peer). Digital Rights Management seems to be the solution to counter this problem advocating the use of cryptography and other related security mechanisms to protect digital content and to associate rights with it which determine how, when and by whom it can be consumed. The rapid growth of digital multimedia technologies brings tremendous attention to the field of digital watermarking. Watermarking embeds a secret message into a cover multimedia data. In media watermarking the secret is usually a copyright notice and the cover a digital image. In digital watermarking, robustness is still a challenging problem if different sets of attacks needed to be tolerated simultaneously. In this paper we present an original spatial watermarking technique for video and images. Our approach modifies blocks of the image or frames by a spatial watermark insertion. Spatial mask of suitable size is used to hide data with less visual impairments. Watermark insertion process exploits average color of the homogeneity regions of the cover image. We took a frame-based approach to video watermarking. From video we extract a certain number of key-frames: the first, the middle, and last key-frame. The first step is decoding: transformation of mpeg to jpeg sequences, after that we select three frames that will be process by applying the watermark mask. In the reverse process of encoding we take the marked frames.

Keywords: video watermarking, robustness, multimedia, JPEG, algorithm.

1 Introduction

The growing proliferation of multimedia digital content throughout, virtually, every digital platform and system available today has produced a profound impact. Digital Rights Management (DRM) solutions aim to enable content providers to assign and oversee usage permissions or rights for multimedia content upon purchase/distribution with the aid of cryptographic mechanisms. Nowadays, the rapid and extensive growth in Internet technology is creating a pressing need to develop several newer techniques to protect copyright, ownership and content integrity of digital media. This necessity arises because the digital representation of media possesses inherent advantages of portability, efficiency and accuracy of information content. On the other hand, this representation also puts a serious threat of easy, accurate and illegal perfect copies of unlimited number. Digital watermarking is now considered an efficient technology for copyright protection. Several image watermark schemes have been developed in the past few years, both spatial and frequency domains are used for watermark embedding [1], [2], [3], [4]. The requirements of watermarking techniques, in general, need to posses the following characteristics: (a) imperceptibility for hidden information, (b) redundancy in distribution of the hidden information inside the cover image to satisfy robustness in watermark extraction process even from the cropped watermarked image and (c) possible use of one ore more keys to achieve cryptographic security of hidden content [5].

Video watermarking involves embedding cryptographic information derived from frames of digital video into the video itself. Ideally, a user viewing the video cannot perceive a difference between the original, unmarked video and the marked video, but a watermark extraction application can read the watermark and obtain the embedded information. The features of the video watermarking algorithm are: video and audio watermark are combined; it is robust against the attack of frame dropping, averaging and statistical analysis; it allows blind retrieval of embedded watermark which does not need the original video; the watermark is perceptually invisible; it is resistant to loss compression. While spatial domain watermarking, in general, is easy to implement on computational point of view but too fragile to withstand large varieties of external attacks. On the other hand, frequency or transformed domain approach offers robust watermarking but in most cases implementation need higher computational complexity. Moreover the transform domain technique is global in nature and cannot restrict visual degradation of the cover image. But in the spatial domain scheme, degradation in image quality due to watermarking could be controlled locally leaving the region of interest unaffected.

The present paper describes a computationally efficient block based spatial domain watermarking technique for a one level watermark symbol. The selection of the required pixels is based on variance of the block and watermark insertion exploits average color of the blocks. The proposed algorithms were tested on the most usual transformations of images and the obtained results showed that the proposed method is efficient.

2 Watermarking Algorithms

All watermarking methods share the same building block – an embedding system and the watermark extraction or recovery system [6]. Any generic embedding system should have as inputs: a cover data/image (I), a watermark symbol (W) and a key (k) to enforce security. The output of the embedding process is always the watermarked data/image (I'). The generic watermark recovery process needs the watermarked data, the secret key and depending on the method, the original data and/or the original watermark as input while the output is recovered watermark W with some kind of confidence measure for the given watermark symbol or an indication about the presence of watermark in the cover image under inspection. The original cover image I is a standard image of size NxN where N = 2p with a 24 bit RGB format. In the proposed work a binary image of size 256x256 or 512x512 is considered. It is marked each image with a watermark coefficient. That means, for each pixel, it is changed the value of the pixel given the following formula:

$$D(i,j) = C(i,j) + a*M*W$$

where C(i,j) is the original value of a pixel at position (i,j); D(i,j) is the watermarked value of the same pixel; 'a' is a scalar factor (here a is chosen constant, but can be a variable of the position to improve the invisibility of the watermark and its detection); M is the mean of the block; and W is the watermark coefficient to be embedded. In our work, W could take the values +1 or –1 (one can easily extend the implementation to M).

Our method has a simpler implementation and less complex computations than the general method.

The proposed watermarking scheme is secure because any degree of degradation we apply the watermark remain in the virtual graph nodes.

We took a frame-based approach to video watermarking.

From video we extract a certain number of key-frames: the first, the middle, and last key-frame. The first step is decoding: - transformation of **mpeg** to **jpeg** sequences, after that we select three frames that will be process by applying the watermark mask. In the reverse process of encoding we take the marked frames.

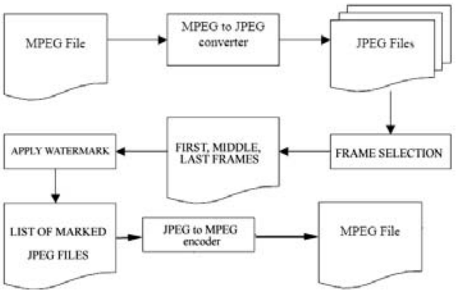


Fig. 1. Proposed watermark processing

The pixels of each image key-frame are arranged into hexagons like in the Fig. 2. Then the image key-frame is viewed as a graph not as a pixel matrix. The vertices represent the pixels and the edges represents neighborhood between pixels. The algorithm for this operation is as following:

```
procedure construct_graph_from_keyframe (Keyframe K,
edge ) is:
  for i=0, width/edge - 3*edge do
    for j=0; height/3 do
      if i modulo 3=0 then
          if j modulo 2=0 then
         K[i][j]=K[edge*i][edge*j+edge-1];
          if j modulo 2 =1 then
         K[i][j]=K[edge*i][edge*j+edge+2];
      if i modulo 3=1 then
          if j modulo 2 =0 then
         K[i][j]=K[edge*i-1][edge*j-edge];
          if j modulo 2 =1 then
```

```
            K[i][j]=K[edge*i-1][edge*j+edge*2];
        if I modulo3 =2 then
             if j modulo 2 =0  then
            K[i][j]=K[edge*i-2][edge*j+edge-1];
             if j modulo 2 =1 then
            K[i][j]=K[edge*i-2][edge*j+edge+2];
   *output the graph
 End
```

The total running time of a call of the presented procedure is O(m*n), where "m" is the width and "n" is the height of image.

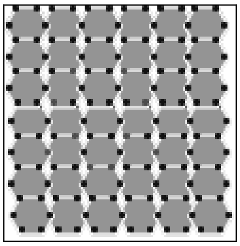


Fig. 2. Example of key-frame block watermarking

For reconstructing the marked image, there will verify only the graph's nodes corresponding to the selected key. If the marked pixel has the color in the interval [min, max] with respect to the color of the bottom pixel, then it will consider that this pixel was marked in conformity with the given algorithm. The values 'min, max' resulted from many experiments.

3 Experiments

There are a lot of transformation that can be done on images [7], [8]: rotation, re-dimension, compression (transforming the image to JPEG), cropping and of course the case in which the image is not changed. Because it is not known what transformation the user done, all these transformations are verified one –by – one and the percent of similitude between the original image and the verified one is returned.

The detection algorithm detects the marked video in percent of 100%. Also this algorithm may be applied if the image is cropped. In this case it may be possible to lose some marked pixels depends on the position where the image was cropped.

The detection algorithm detects this cropped video in percent of 91.3%.

In the case of image rotation (angle of 90, 180 and arbitrary), there are verified all the key-frame nodes because the nodes' position is changed. We search the nodes for which all of the three color' channels have the values like the color's channels of the bottom pixel minus one (or a certain constant). Before verifying these nodes, the image is re-dimensioned to the initial dimensions at which the image is marked.

By experiments, there resulted that in the case of rotation by 90 and 180 degree, the constant is zero, but in the case of rotation by arbitrary angle the constant is very great (100).

Table 1. The percents of recognition obtained between the un-rotated and rotated marked key-frames, for different rotation angles

Rotation Angle	Recognition Percent
30	55.1%
45	55.1%
60	55.1%
90	100%
180	100%

In the case when we want to detect an image that was enlarged the results are weaker. From experiments resulted the following percents of recognition between the marked image and the marked enlarged image, as in the following table. The greater are the enlarge percents the weaker are the recognition percents.

Table 2. The percents of recognition obtained between the marked and enlarged marked key-frames, for different enlarged percents

Rotation Angle	Recognition Percent
20	89%
40	75%
60	74.22%
80	72.3%
100	55.6%

Another possibility is the compression of JPEG key-frames at a different quality level.

The implemented algorithm entirely detects a key-frame that was compressed to JPEG, a quality level of 100% and 80%. The color of pixels arranged into nodes selected by us for marking the image is changed because of transformations supported by the image. Then the color of these pixels is searched into a certain interval.

From experiments results that a good value for this constant is 30. Using different degree of image compression for JPEG key-frames, the following percents of recognition between the marked image and the JPEG marked image are resulted, as in the bellow table:

Table 3. The percents of recognition obtained between the marked and compressed marked key-frames, for different quality level

Quality	Recognition Percent
100	100%
80	100%
60	33.33%
50	33.33%
30	15.4%

4 Conclusions

The paper proposed a watermark technique for video with the following properties:

- Invisibility
- Blind or Informed Detection
- Capacity (Number of bits that can be hidden)
- Robustness – Key-frame watermarking (filtering, resizing, contrast enhancement, cropping, rotation, etc)

Our method has a simpler implementation and less complex computations than the general method. The proposed watermarking scheme is secure because any degree of degradation we apply, the watermark remains in the virtual graph nodes.

The method developed above satisfies the necessary requests for the watermarking technique and the series of presented transformations accounts for the fact that it resists possible attacks. The method is easy to implement and the experimentally determined robustness shows that it can be used without fear of being detected or changed.

References

1. Hsu, C.T., Wu, J.-L.: Image Watermarking by Wavelet Decomposition. Academy of Information and Management Sciences Journal 3(1), 70–86 (2000)
2. Suhail, M., Obaidat, M.S.: Digital Watermarking-Based DCT and JPEG Model. IEEE Transactions on Instrumentation and Measurement 52(5), 1640–1647 (2003)
3. Wong, P.H.: A data hiding technique in JPEG domain compressed domain. In: Proceeding of SPIE Conf. of Security and Watermarking Multimedia Contents, vol. 4314, pp. 1820–1827 (2001)
4. Holliman, N.: Adaptive public watermarking of DCT-based compressed images. In: Proc. Of SPIE-The international society for optional engineering, pp. 284–295 (1998)
5. Katzenbesser, S., Petitcolas, F.A.P.: Information Hidden Techniques for Steganography and Digital Watermarking, Artech House, Boston, MA, pp. 1512–1519 (2000)
6. Burdescu, D.D., Stanescu, L.: An Algorithm for Authentication of Digital Images. In: Proceedings of International Conference on Security and Cryptography, pp. 303–308 (2006)
7. Lin, C., Wu, M., Lui, Y.M., Bloom, J.A., Miller, M.L., Cox, I.J.: Rotation, Scale and Translation Resilient Public Watermarking for Images. IEEE Transaction on Image Processing 10(5), 767–782 (2001)
8. Fei, C., Kundur, D., Kwong, R.: Analysis and Design of Watermarking Algorithms for Improved Resistance to Compresion. IEEE Transaction on Image Processing 13(2), 126–144 (2004)

Watermarking Software to Signal Copy Protection

Ernő Jeges[1], Zoltán Hornák[1], Gergely Eberhardt[2], and Zoltán Nagy[2]

[1] Budapest University of Technology and Economics,
Department of Measurement and Information Systems, SEARCH Laboratory
{jeges,hornak}@mit.bme.hu
[2] SEARCH-LAB Ltd.
{gergely.eberhardt,zoltan.nagy}@search-lab.hu

Abstract. Enforcement of copyright laws of software products is primarily managed in legal way, as the available technological solutions are not strong enough to prevent illegal distribution and use of software. This situation is particularly dangerous on the growing market of mobile software products: the lack of protection may be the most important setback of the expansion of this market, although we have all the necessary components to assure a trusted environment on handsets. In this paper we present our copy protection scheme called *Swotector* targeting primarily mobile phone applications, which combines techniques of obfuscation and software-watermarking to provide a solution which is purely technical, however still provides the necessary flexibility. Our test results have shown that the solution is efficient enough to overcome current challenges of software copy protection, above all the ability of the operating system to run both protected and non-protected applications.

Keywords: software copy protection, software watermarking, obfuscation, reverse engineering, trusted OS, mobile software.

1 Introduction

According to the statistics of the Business Software Alliance (BSA) the global financial losses due to software piracy were about $30 billion in 2004 [1]. It is a common belief that there is not much to be done against the piracy of software in the PC world, however in the world of mobile phones, where the integrity of the operating system can be trusted, we may still have the opportunity to evolve in such way that the losses due to illegal software distribution could be at least moderated.

In our belief the availability of a strong and flexible copy protection scheme is the most important prerequisite for further expansion of the mobile phone software market. This is why our research targeted embedded systems used in mobile phones, on which some slightly different assumptions can be made than on usually discussed PC platforms, as we can rely on a *trusted OS* ensuring the integrity of the processes.

We propose a scheme, which merges the reliability of the public key infrastructure (PKI) with obfuscation and software watermarking techniques, assuming a trusted and tamperproof OS on mobile phones, resulting in a protection supporting both freely distributable and copy protected software[1].

2 Background

Two main categories of software copy protection mechanisms exist: the autonomous systems and those, which use external collaboration [2].

The protections of autonomous systems are integrated into the software itself, so the security depends only on the used software techniques, including integrity protection, software obfuscation, calculating checksums, encryption, etc. [3] The most common solutions are based on the program checking itself, which are part of the program, so one can reveal them by reverse engineering and can bypass the protection by modifying the code [2]. These techniques are neither theoretically nor – based on our experience – practically secure enough [4].

The other category of protection mechanisms involve some external collaboration: the application uses a tamperproof processor, an operating system or other secure hardware or software solutions. This support can be either on-line or off-line [2].

To link the authorized user to his or her instance of the software, usually the services of a public key infrastructure [5] are used. For a copy protected software a license containing the information about the user, the product issuer or distributor and about the product itself should be attached to the product, so that the OS could check the authorization. The integrity of this license is protected by a digital signature, and the OS should not run copy protected software without the appropriate license.

However, to support multiple use cases and business models, and also to allow comfortable software development, the OS should be capable of running both copy protected and unprotected software, which is one of the biggest challenges in developing a successful copy protection scheme.

As opposed to usual protection models used in multimedia files using watermarking to trace content in order to identify its origin, software can apply watermarking to make possible the indication on the code that it is copy-protected. As the instructions of the code can be obfuscated arbitrary as long as the user-perceptible output remains the same, this enables us to implement more efficacious watermarks to software than to audio/video files.

We can differentiate between two types of *software watermarking* techniques: static and dynamic. In case of static watermarks the information is injected into the application's executable file. The watermark is typically inside the initialized *data, code or text* sections of the executable [3]. As opposed to this, dynamic watermarks are stored in the program's execution state. This means that the presence of a watermark is indicated by some run-time behavior of the executable [6].

To prevent the easy removal of watermarks we can use for example *software obfuscation*, which is a collection of several different code transformations, originally with the common goal to make the reverse engineering more difficult both in case of automatic tools and for the human understanding of the code [7]. The most important

[1] The research and development project was realized by the financial support of the Economic Competitiveness Operative Programme (GVOP 3.1.1/AKF) of the Hungarian Government.

aim of these transformations in our proposed copy protection scheme is to make the code transformations meant to remove the watermark hard to accomplish.

3 The Copy Protection Scheme

The most important building blocks of the proposed copy protection scheme are the above mentioned watermarking and obfuscation techniques and the services of a public key infrastructure (PKI). The integrity of the software is ensured through a digitally signed license, and if the license or its digital signature is invalid, the OS should prevent the application from running. However, if there is no license attached to the application, the OS can start it assuming that it is not protected, but should continuously check for the presence of watermark in it, which would designate that originally it was a protected peace of software, so should have a license file attached. The removal of the watermark is made hard with applying different obfuscation methods, so that it is extremely hard for the attacker to change the code in order to break the protection and make the application run without the license.

License checking, as the most important part of the scheme is shown in Fig. 1:

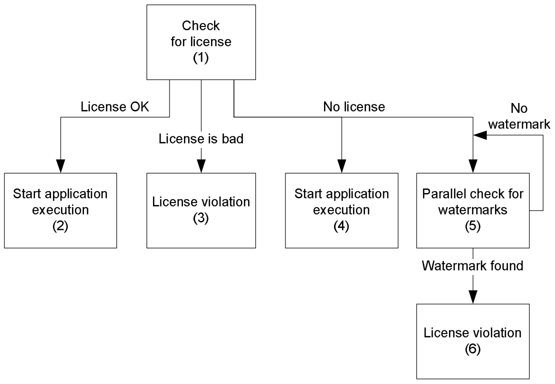


Fig. 1. The license checking algorithm

The major steps of the license checking algorithm are the following:

1. Check if a digitally signed license is attached to the application.
2. If there is a license, and if both the license (i.e. its hash signature) and the digital signature are valid, then the application can be run without any further checks.
3. If the license or its digital signature is not valid, the application should stop immediately as the copy protection is violated.
4. If there is no license attached to the application, it could either be a manipulated copy protected, but also a freely distributable software, so it should start.
5. Parallel to this the OS should start the continuous search for watermarks.
6. If the watermark is found in the application, its running should be stopped immediately, and the appropriate steps should be made, because the presence of the

watermark unambiguously signals that a license should have been attached to the application, without which it is an illegal copy.

In our scheme we have chosen to use dynamic watermarks, because this way the watermark is formed continuously during program execution; this implies that the program should run for a while so that the OS can detect the watermark. The generation of the watermark is inserted into the code of the protected application by several transformations applied to it. These transformations (together with some control and data obfuscation methods) are applied at the assembly code level, and the process is integrated into the compilation process. The whole process consists of four major steps, which are the (1) Preparation, (2) Analysis, (3) Transformation, and (4) Synthesis. As a first step the source code is transformed to assembly, which is then parsed to get an internal representation. Control flow and data analysis is accomplished on this representation as a second step, after which the transformations are accomplished, aiming at both watermarking and obfuscation. Finally, the internal representation is serialized back to assembly code, compiled and linked.

4 The Software Watermarking Technique

The basic idea is that the dynamic watermark signals the fact that the application is protected by statistically boosting the appearance of some specific values, the watermark values in the memory state of the application. A watermark value (*WM*) is derived from a random value appended with its transformation with a function *g*. This function can be even a simple XOR operation; its role is simply to keep the probability of appearing of such a value pair in a non-protected application low. The watermark value can be defined as follows:

$$WM = \left(r; g(r)\right), \tag{1}$$

where $r = RND$, a different random value every time a new WM is needed.

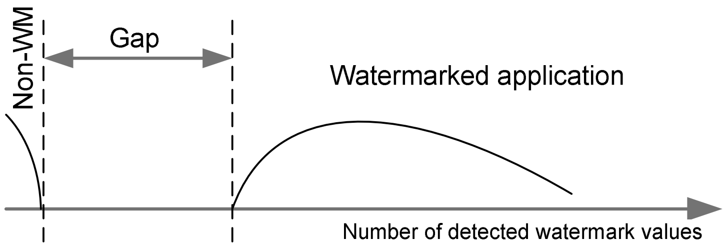


Fig. 2. The gap between non-watermarked and watermarked applications

These different *WM* values should be hidden in the memory of the process as frequently and for as long as possible, which means that these values should appear in the state of the program frequently enough to allow their detection and statistical evaluation of these detections. To achieve this goal we can for example pick the parameters of different data obfuscation transformations in a way that one or more original values of a variable (D_0, e.g. a typical value of a loop control variable) are transformed into watermark values, which are then stored in program state in the transformed domain. The gap between the prevalence of such watermark values in non-watermarked and watermarked applications, as shown in Fig. 2, can allow easy detection of the watermark by statistical means.

The proposed watermarking technique is loosely interwoven with data obfuscation, as the watermarking module calculates the parameters of the obfuscation in such way that the transformed data should signal the presence of watermark through transforming some common original values (D_0) to a watermark value (*WM*).

5 Results

To evaluate the software watermarking we have implemented a full framework system called *Swotector*, a number of control and data obfuscation methods, and a watermarking technique relying on the linear variable encoding data obfuscation [7]. We performed different tests in the framework: the results are summarized in Table 1.

Table 1. Results of test executions of watermarked and obfuscated code

Test case	Obf. Level	WM Level	WM Count	Check Count	Lines	Blocks	Exec. Time
1	None	None	0	261822	282	38	01:43
2	None	Low	6806	360574	536	38	01:36
3	None	Normal	13565	243006	883	38	01:40
4	Low	None	0	476782	517	57	10:26
5	High	None	0	477913	5939	191	10:27

In Table 1 above columns *Obf. Level* and *WM Level* designate the applied obfuscation and watermark levels. In column *WM Count* the number of occurrences of watermark values, while in *Check Count* the overall number of examined variables during the execution are shown. Columns *Lines* and *Blocks* show the number of lines and blocks in the assembly file after the transformations, and finally the last column shows the execution time of the transformed test application on the test platform (*Nokia 6600 – Symbian S60*).

6 Conclusions

From the results shown above we can conclude the followings:

- Watermark is not detected in non-watermarked software (*WM Count* was zero in test cases 1, 4 and 5). This was an important expectation, which is apparently fulfilled by our watermarking technique.
- In the watermarked software the number of detected watermark values is higher by several orders of magnitude (*WM Count* in test cases 2 and 3 has value of several

thousands) than in non-watermarked ones (where it was zero), so this proves the viability of our concept, as the gap shown in Fig. 2 can be detected dependably.

- The implemented watermarking solution does not go with a high overhead in performance (the number of lines and blocks did not increase by different watermarking levels in test cases 1–3).
- The obfuscation transformations increase the complexity of the software, however are also leading to a remarkable overhead in execution time (test cases 4 and 5).

7 Summary

In the above paper we have presented our copy protection scheme combining cryptography (PKI), software watermarking and obfuscation in order to achieve a strong but still flexible technical solution for software copy protection, targeting primarily mobile software development. The main idea is to support the running of both copy protected and freely distributable software by signaling that the application is protected via the presence of a watermark. Based on this scheme we have designed the architecture of a protection tool named *Swotector* that can be integrated into a development environment to provide the proposed protection solution.

Test results have shown that our technique is viable, as the gap of the statistical appearances of watermarked values is several of orders of magnitude wide between the non-watermarked and watermarked applications, thus the detection of the watermark can be done robustly enough.

References

1. First Annual BSA and IDC Global Software Privacy Study, Business Software Alliance and IDC Global Software (2004)
2. Mana, A., Lopez, J., Ortega, J.J., Pimentel, E., Troya, J.M.: A Framework for Secure Execution of Software. International Journal of Information Security 2(4), 99–112 (2004)
3. Hachez, G.: A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards. Ph.D. thesis, Universite Catholique de Louvain, Louvain-la-Neuve (2003)
4. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)Possibility of Obfuscating Programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
5. International Telegraph and Telephone Consultative Committee (CCITT): The Directory – Authentication Framework. Recommendation X.509 (1988)
6. Collberg, C., Thomborson, C., Townsend, G.M.: Dynamic Graph-Based Software Watermarking. Technical Report TR04-08 (2004)
7. Collberg, C., Thomborson, C., Low, D.: A Taxonomy of Obfuscating Transformations. Technical Report 148. Department of Computer Science, The University of Auckland (1997)

Author Index

Printing: Mercedes-Druck, Berlin
Binding: Stein + Lehmann, Berlin