Anirban Chakrabarti

Grid Computing Security

Springer

Grid Computing Security

Anirban Chakrabarti

Grid Computing Security

With 87 Figures and 12 Tables

Springer

Anirban Chakrabarti

Infosys Technologies Limited
Electronic City
Hosur Road
560100 Bangalore
India
Anirban_Chakrabarti@infosys.com

Library of Congress Control Number: 2007922355

ACM Computing Classification (1998): C.2, D.4.6, K.6.5

ISBN 978-3-540-44492-3 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Typeset by the author
Production: Integra Software Services Pvt. Ltd., India
Cover design: KünkelLopka Werbeagentur, Heidelberg

Printed on acid-free paper 42/Integra 5 4 3 2 1 0

Preface

Grid computing is widely regarded as a technology of immense potential in both industry and academia. The evolution pattern of grid technologies is very similar to the growth and evolution of Internet technologies that was witnessed in the early 1990s. Similar to the Internet, the initial grid computing technologies were also developed mostly in the universities and research labs to solve unique research problems and to collaborate between different researchers across the globe. Recently, the high computing industries like finance, life sciences, energy, automobiles, rendering, etc. are showing a great amount of interest in the potential of connecting standalone and silo based clusters into a department and sometimes enterprise-wide grid system. Grid computing is currently in the midst of evolving standards, inheriting and customizing from those developed in the high performance, distributed, and recently from the Web services community. Due to the lack of consistent and widely used standards, several enterprises are concerned about the implementation of an enterprise-level grid system, though the potential of such a system is well understood. Even when the enterprises have considered grid as a solution, several issues have made them reconsider their decisions. Issues related to application engineering, manageability, data management, licensing, security, etc. have prevented them from implementing an enterprise-wide grid solution. As a technology, grid computing has potential beyond the high performance computing industries due to it's inherent collaboration, autonomic, and utility based service behavior. To make this evolution possible all the above-mentioned issues need to be solved. Some of the issues are technical and some of them have business and economic overtones like the issue related to licensing. Each of the issues mentioned above is important and deserves a close look and understanding. In this book we will solely concentrate on the issue related to grid computing security.

As an issue, security is perhaps the most important and needs close understanding as grid computing offers unique security challenges. In this book we look at different security issues pertaining to the grid system; some of them are of immediate concern and some are long term issues. We will also look at security issues in other areas of computer science like networks and operating systems which may affect the design of future grids. We have categorized the issues pertaining to grid computing security into three main buckets *viz.* architecture related issues, infrastructure related issues, and management related issues. Architecture related issues are concerned about the overall architecture of the grid system like the concerns pertaining to information security, concerns about user and resource authorization, and issues pertaining to the overall service offered by the grid system. The infrastructure related issues are concerned about the underlying infrastructure which includes the hosts or the machines, and the network infrastructure. In addition, several management systems need to be in place for an all pervasive enterprise level and secure grid system. There are three main types of management systems which are important from the grid perspective, namely the credential management systems, the trust management systems, and the monitoring systems. All the three issues mentioned above are dealt with in this book, along with existing solutions and potential concerns.

Organization

In this book we have made no assumption about the prerequisites for the readers. We have provided a short background on grid computing, security technologies, and Web service standards for readers who are new to this field. It is to be noted that the background is not extensive and enough references are provided for readers to have a fair understanding about the different background technologies. The book is organized into 13 chapters and an appendix. Chapter 1 looks at the background, benefits, and concerns pertaining to grid systems. Chapter 2 talks about the different security technologies that are available and useful to build a secure grid system. The different security technologies that are covered in this chapter are different authentication/encryption systems, identity protocols and popular technologies like Kerberos, HMAC, SSL/TLS, IPSec, among others. It is to be noted that Web services security standards, which form the backbone of grid security, are not dealt with in this chapter. They are separately listed in the appendix. Chapter 3 provides a taxonomy of the different grid security issues and solutions. We feel that the chapter is important because readers will get a snapshot of different issues, solutions, and concerns in one place and can refer to the detailed discussions in the subsequent chapters, if interested. After the brief overview and background about the different technologies, landscape, and taxonomy we discuss the different issues in detail in Chap. 4 to 11. In Chap. 4, we look at the information security aspects of the grid system. Here we look at the grid standards like GSI, and security implementations of popular grid standard Globus. In Chap. 5, we look at the authorization systems namely the Community Authorization Service (CAS), Virtual Organization Membership Service (VOMS), Akenti, PERMIS, among others. In Chap. 6, we look at the issues pertaining to the grid service *viz.* Denial-of-Service (DoS) attacks and Quality of Service (QoS) violation attacks. Different solutions and concerns are also discussed in this chapter. It is to be noted that since grid systems have relatively limited deployments, most of these attacks and solutions described in the chapter have been borrowed from the domain of the Internet. Many of these solutions would be useful in designing the future secure grid. Chapter 7 looks at the security issues pertaining to the hosts or the machines comprising the grid system. Different solutions like sandboxing, flexible kernels, virtualization, etc. have been discussed in detail in this chapter. Chapter 8 looks at another important infrastructure component namely the network. The immediate issues like integrating firewalls, VPNs, etc. are looked at in this chapter. In addition some long term issues like secure grid multicasting sensor grids, and others are also discussed in this chapter. Chapters 9 - 11 deal with the different management systems. Chapter 9 discusses about credential management systems like MyProxy, Smartcards, etc. and issues pertaining to them. Chapter 10 talks about trust management systems and issues like trust creation, negotiation among others. These issues are important in dynamic systems and have enormous research potential. Chapter 11 talks about the monitoring systems that are currently present. Two grid security case studies are provided in Chap. 12. These case studies should help readers in getting a holistic view of the different concepts, principles, protocols, and technologies mentioned in the book. Finally Chap. 13 concludes the book by looking at a few future technologies and mapping the issues and solutions into immediate, medium term and long term categories.

Acknowledgments

I would like to thank all those people who have contributed to the book. I would like to thank my colleagues, Dr Shubhashis Sengupta, Mr Deependra Moitra, Mr Srikanth Sundarrajan, and Mr Hariprasad Nellitheertha for helping me sort out the administrative issues, and providing me with useful comments and reviews. I would also like to thank Mr Anish Damodaran for providing me with useful inputs whenever needed. Moreover, I would take the opportunity to thank Mr Ralf Gestner and Ms Ulrike Stricker of

Springer for providing me the opportunity and extending support in writing this book. I am also extremely grateful to all the reviewers for their insightful comments. Moreover, I would like to extend my gratitude to the Springer production and copyediting team for their tireless efforts. Finally, special thanks go to my wife Lopamudra and also to my mother for their constant support and encouragement.

Contents

Preface v
Organization vi
Acknowledgments vii

1 Introduction..1
1.1 Background...1
1.2 Grid Computing Overview 3
1.2.1 Evolution of Grid Computing 4
1.2.2 Benefits of Grid Computing...6
1.2.3 Grid Computing Issues and Concerns................................8
1.3 About the Book..11
1.3.1 Target Audience..12
1.3.2 Organization of the Book...12

2 Overview of Security 15
2.1 Introduction 15
2.1.1 Characteristics of Secure System.....................................15
2.1.2 Security Threats 16
2.2 Different Encryption Schemes...17
2.3 Different Authentication Schemes..20
2.3.1 Shared Secret Based Authentication 20
2.3.2 Public Key Based Authentication 21
2.3.3 Third Party Authentication Schemes 21
2.4 Different Integrity Schemes..22
2.4.1 Message Authentication Code (MAC)..............................22
2.4.2 Keyed MAC 23
2.5 Standard Protocols...24
2.5.1 Public Key Infrastructure..24
2.5.2 Secure Socket Layer (SSL) 27
2.5.3 Kerberos..27
2.5.4 IP Security (IPSec)...29
2.6 Chapter Summary 31

3 Taxonomy of Grid Security Issues 33
3.1 Introduction 33
3.1.1 Grid Security Taxonomy 35
3.2 Architecture Related Issues 36
3.2.1 Information Security 36
3.2.2 Authorization 37
3.2.3 Service Security 38
3.3 Infrastructure Related Issues 40
3.3.1 Host Security Issues 40
3.3.2 Network Security Issues 41
3.4 Management Related Issues 42
3.4.1 Credential Management 43
3.4.2 Trust Management 44
3.4.3 Monitoring 45
3.5 Chapter Summary 46

4 Grid Information Security Architecture 49
4.1 Introduction 49
4.2 Grid Security Infrastructure (GSI) 50
4.2.1 Grid Security Model 52
4.3 Authentication in GSI 54
4.3.1 Certificate based Authentication 54
4.3.2 Password based Authentication 57
4.3.3 Integration with Kerberos 59
4.4 Delegation in GSI 59
4.5 An Example: Security in Globus Toolkit 4.0 (GT4) 61
4.5.1 Message Protection in GT4 61
4.5.2 Delegation in GT4 64
4.6 Chapter Summary 65

5 Grid Authorization Systems 67
5.1 Introduction 67
5.1.1 Different Access Control Models 69
5.1.2 Push vs. Pull Authorizations 72
5.2 Characteristics of Grid Authorization Systems 74
5.2.1 Scalability Issues 75
5.2.2 Security Issues 76
5.2.3 Revocation Issues 77
5.2.4 Inter-operability Issues 79
5.2.5 Grid Authorization Systems 79

5.3 VO Level Authorization Systems 80
5.3.1 Community Authorization Service (CAS) 80
5.3.2 Virtual Organization Membership Service (VOMS) 87
5.3.3 Enterprise Authorization and Licensing Service (EALS) 88
5.4 Resource Level Authorization Systems 90
5.4.1 Akenti 90
5.4.2 Privilege and Role Management Infrastructure Standards Validation (PERMIS) Project 94
5.4.3 Authorization Using GridMap 100
5.5 Comparing the Different Authorization Systems 100
5.5.1 Comparison 100
5.5.2 Roadmap to Grid Authorization Systems 103
5.6 Chapter Summary 103

6 Service Level Security in Grid Systems 105
6.1 Introduction 105
6.1.1 Components of Service 106
6.1.2 Service Vulnerabilities 106
6.2 DoS Attacks and Countermeasures 108
6.2.1 Effect of DoS attacks 108
6.2.2 Distributed Denial-of-Service Attacks 111
6.2.3 Existing DoS Countermeasures 118
6.2.4 Preventive DoS Counter-measures 119
6.2.5 Reactive DoS Countermeasures 123
6.2.6 Comparison between DoS Countermeasures 126
6.3 QoS Violation Attacks and Countermeasures 127
6.3.1 Different Types of QoS Violation Attacks 128
6.3.2 Existing Solutions 129
6.4 Chapter Summary 131

7 Host Level Security 133
7.1 Introduction 133
7.2 Data Protection Issue 135
7.2.1 Application Level Sandboxing 136
7.2.2 Virtualization 138
7.2.3 Flexible Kernel Systems 145
7.2.4 Sandboxing 149
7.3 Job Starvation Issue 154
7.3.1 Advanced Reservation Techniques 155
7.3.2 Priority Reduction Techniques 156
7.4 Chapter Summary 157

8 Grid Network Security 159
8.1 Introduction 159
8.1.1 Grid Network Security Issues 159
8.2 Firewalls 161
8.2.1 Different Types of Firewalls 163
8.2.2 Firewalls and Grid – Issues 165
8.2.3 Firewalls and Web Services 167
8.3 Virtual Private Networks (VPN) 168
8.3.1 VPNs and Grid – Types of VPNs 169
8.3.2 VPNs and Grid – Issues 170
8.3.3 VPNs and Grid – Some Solutions 171
8.4 Secure Routing 173
8.4.1 Impacts of Routing Table "Poisoning" 173
8.4.2 Different Routing Protocols 175
8.4.3 Routing Attacks and Countermeasures 175
8.5 Multicasting 178
8.5.1 Secure Multicasting 178
8.6 Sensor Grids 182
8.6.1 Security in Sensor Networks – Issues 183
8.6.2 Existing Solutions 186
8.7 High Performance Interconnects 188
8.7.1 10-Gigabit Ethernet 188
8.7.2 Infiniband Architecture (IBA) 188
8.7.3 Some High Performance Security Solutions 189
8.8 Chapter Summary 190

9 Grid Credential Management Systems 193
9.1 Introduction 193
9.1.1 Types of Credentials 194
9.1.2 Characteristics of Credential Management Systems 195
9.1.3 Different Credential Management Systems 197
9.1.4 Centralized Vs. Federated Credential Management 198
9.2 Credential Repositories 200
9.2.1 Smart Cards 200
9.2.2 Virtual Smart Cards 201
9.2.3 MyProxy Online Credential Repository 202
9.3 Federated Credential Management Systems 205
9.3.1 Virtualized Credential Manager (VCMan) 206
9.3.2 KX.509 208
9.3.3 Liberty Alliance for Federated Identity 209
9.3.4 Shibboleth Identity Federation 210
9.4 Chapter Summary 212

10 Managing Trust in the Grid 215
10.1 Introduction 215
10.1.1 Definition of Trust 215
10.1.2 Reputation and Trust 217
10.1.3 Categories of Trust Functions 218
10.2 Trust Management Systems 221
10.2.1 Life Cycle of Trust Management Systems 223
10.2.2 Characteristics of Trust Management Systems 225
10.3 Reputation-Based Trust Management Systems 228
10.3.1 PeerTrust – A P2P Trust Management System 228
10.3.2 XenoTrust Trust Management System 231
10.3.3 NICE Trust Management System 233
10.3.4 Secure Grid Outsourcing (SeGO) System 236
10.4 Policy-Based Trust Management Systems 238
10.4.1 PeerTrust Trust Negotiation 238
10.4.2 TrustBuilder 240
10.4.3 Trust Negotiation for the Grid 242
10.5 Comparing the Trust Management Systems 243
10.5.1 Generic Understanding of Trust Management Systems 243
10.5.2 Applicability of the Trust Management Systems 245
10.6 Chapter Summary 246

11 Grid Monitoring 247
11.1 Introduction 247
11.1.1 Stages of Monitoring 248
11.1.2 Requirements of Distributed Monitoring System 250
11.2 Grid Monitoring Architecture (GMA) 251
11.3 Different Monitoring Tools/Frameworks 253
11.3.1 Simple Network Management Protocol (SNMP) 254
11.3.2 Different System Monitoring Tools 255
11.3.3 Ganglia 256
11.3.4 Hawkeye Monitoring System 258
11.3.5 Relational GMA (RGMA) 259
11.3.6 Globus Monitoring and Discovery System (MDS) 261
11.3.7 Management of Adaptive Grid Infrastructure (MAGI) 263
11.3.8 GlueDomains 265
11.4 Discussions on the Different Monitoring Systems 266
11.4.1 Comparison 266
11.4.2 Applicability 268
11.5 Chapter Summary 269

12 Putting it All Together 271
12.1 Security in the European Data Grid (EDG) 271
12.1.1 Authentication and Delegation 271
12.1.2 Credential Management 272
12.1.3 Job Execution 272
12.2 An Enterprise Case Study 274
12.2.1 Overview of the Security Architecture 275
12.3 Chapter Summary 278

13 Conclusion 281
13.1 Looking at the Future 281
13.1.1 Identity Based Encryption (IBE) 281
13.1.2 Application Oriented Networking (AON) 282
13.2 Summarizing the Security Issues in Grid 283
13.2.1 Immediate Issues 284
13.2.2 Medium-term Issues 285
13.2.3 Long-term Issues 287
13.3 Summarizing the Security Solutions in the Grid 289
13.3.1 Solutions to Immediate Issues 289
13.3.2 Solutions to Medium Term Issues 290
13.3.3 Solutions to Long Term Issues 291

Appendix 293
A.1 Web Services 293
A.1.1 Components of Web Services 294
A.2 Web Services Security 296
A.2.1 WS-Security 299
A.2.2 WS-Policy* 301
A.2.3 WS-SecureConversation 302
A.2.4 Security Assertions Markup Language (SAML) 304
A.2.5 eXtensible Access Control Markup Language 306
A.3 Open Grid Services Architecture (OGSA) 307
A.3.1 Open Grid Services Infrastructure (OGSI) 308
A.3.2 Web Services Critique of OGSI 309
A.3.2 Web Services Resource Framework (WSRF) 310

Bibliography 313

Index 329

1 Introduction

1.1 Background

When we watch the recent spectacular science fiction and fantasy thrillers, we remain totally under the spell of the amazing scenes and the underlying special effects that are being displayed. We watch the movies, applaud the actions, and come home totally enchanted and wanting for more. Most of the time we forget about the enormous amount of effort that is required to produce such a spectacle. Even if we acknowledge that, we tend to ignore the proverbial "work horses" or computers that generate such remarkable special effects. We have got used to the special effects so much that we take the computing power required to provide such visual spectacles for granted. It takes billions and trillions of CPU cycles to create special effects like those in *Spiderman* [1], *Shrek* [2], and other such visual treats. To satisfy the ever increasing hunger for better special effects among movie-goers, more and more complex animations are being developed which are continuously raising the bar for computing power required.

Requirements of huge amounts of computing power are not only limited to the field of rendering and animation. Scientists are analyzing terabytes and petabytes of data to provide better weather forecasting [3], develop more efficient models for detecting natural disasters, high energy physics [4], and so on. By virtue of the hugely popular SETI@Home [5] project, most of us are aware of the enormous computing power required for searching extraterrestrial intelligence. The project allowed people to download the SETI@Home software in their own machines and run the program in the screen saver mode. Apart from fundamental research, computing power is also required in huge quantities in the life sciences industry for drug discovery [6]. Financial industries require huge amounts of processing power to do risk calculations, credit analysis, and so on [7]. Manufacturing industries are not very far behind. Simulations of automobiles based on complicated mathematical models [8] require enormous computing power. Similarly, EDA and Oil & Gas explorations also require computing power to do more computations in a shorter time to satisfy the ever increasing demands of the market.

Therefore huge computing power is required in several industries. Now if we look at the computing resources available, we will find that the laptops of today are perhaps as powerful as servers a decade ago. Moore's law, which states that computing power doubles every eighteen months, is valid even today and will probably be true for the next five to six years. With the advancements in the field of multi-core technologies, this growth can be extended further [9]. Therefore computing power is increasing and so is the demand. In this rat race, researchers have found an able ally in the form of networking. Between 2001 and 2010, while processing power is supposed to increase 60 times, networking capabilities is supposed to increase by 4000 times. This means that at the same cost 4000 times the same bandwidth will be available in 2010 as compared to 2001 [10]. Therefore the computing architectures developed a decade back would probably require a rethink based on the technological progress in the fields of computers and networks. Last decade saw the development of a field called *cluster computing* [11] where the different computing resources are connected together using a very high speed network like the Gigabit Ethernet or more recently Infiniband [12].

In addition to the technological progress and the huge requirement of computing power, enterprises have also undergone a radical shift in Information Technology (IT) operations in the last few years. Enterprises are now witnessing increasing collaboration and data sharing among the different participating entities, resulting in the need and use of distributed resources and computing. Another important element that has increased the complexity of IT operations is the need for integration of different applications, middleware developed in different platforms and by different vendors. We are also seeing a spurt of mergers and acquisitions which require integration of technologies across enterprises. Moreover, the enterprises are outsourcing the nonessential elements of the IT infrastructure. The dual pull of requiring more computing power and the integration of heterogeneous components into the IT infrastructure has led to the development of *grid technologies*. The technology is seeing a classical evolution pattern. Initiated and started from the academic and the research community to fulfill their needs, it is slowly being adopted by the enterprises, especially those who have high computing needs like the life sciences, finance, and manufacturing industries. However, the promise of grid computing goes beyond that and the next few years should see a gradual adoption of the grid as a natural choice among the other enterprises. However, the widespread adoption of grid computing as an automatic choice in enterprises depends upon the ability of the researchers and practitioners in reducing the pitfalls that lie on the way. One such pitfall is *security* which is the focus of the book as a whole. In this chapter we will briefly look at the evolution of grid computing, its benefits, and concerns.

1.2 Grid Computing Overview

One of the earliest proponents of grid technology is Ian Foster of the Argonne National Laboratory and professor at the University of Chicago. In 1998, in a book called *The Grid: Blueprint for a New Computing Infrastructure* co-authored with Carl Kesselman, Foster defined the grid as "*A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities*" [13]. Over the years even Foster's definition of a computational grid has evolved, by his own admission. In a subsequent article, *The Anatomy of the Grid*, co-authored with Steve Tuecke in 2000, he changed the definition to include some element of social and policy issues, stating that grid computing is concerned with, "*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*" [14].

The major applications and systems vendors who have a grid strategy also have their own definitions. Oracle® described its vision of the grid as an adaptive software infrastructure which is able to balance resources efficiently through the usage of low cost servers and storage [15]. Sun® Microsystems, meanwhile, breaks the grid down into three levels: cluster grids, enterprise grids, and global grids. While cluster grids are the simplest form of grid where the resources within a local area network are shared, the enterprise grid takes a broader picture, where the resources within an enterprise are shared. Global grids, on the other hand, talk about a grid across enterprises sharing resources [16]. HP® tends to talk more about utility computing – its own take on the grid concept – while IBM® talks about grid technology in the same breath as its vision for what it calls autonomic computing [17]. According to IBM's® definition of grid, they are a collection of resources which create dynamic virtual organization [18]. While the academics' and vendors' definitions of grid vary somewhat, there are some consistent themes: at a basic level they are talking about some sort of network of computing resources – not necessarily inside an organization's own walls – that can be given jobs to do, and that with only a limited amount of user intervention, will get them done quickly, reliably, and cheaply.

In a more generic sense, a grid is *a hardware and software infrastructure that allows service oriented, flexible, and seamless sharing of heterogeneous network of resources for compute and data intensive tasks and provides faster throughput and scalability at lower costs.*

1.2.1 Evolution of Grid Computing

Though grid computing has become the buzzword in both industry and academic communities, it is not a technology which has been developed from scratch. Rather, it is a conglomeration of different existing technologies like cluster computing, peer-to-peer (P2P), and Web services technologies.

During the last decade different technology elements like cluster computing and peer-to-peer computing (P2P) have evolved from the distributed and high performance computing communities respectively. In cluster computing, different computing resources like machines, servers, etc. are connected together by high-speed inter-connects like Infiniband, Gigabit Ethernet, etc. to provide high performance. Computing paradigms like Message Passing Interface (MPI) [19] and Parallel Virtual Machines (PVM) [20] allow programmers to write parallel programs for clusters. Peer-to-Peer system, on the other hand, allows peers or computers to share resources. They are suitable for storing files or information either in an unstructured or a structured P2P mode. Gnutella [21] is a classic example of unstructured P2P where users store the files and a particular request is processed in a heartbeat manner. Structured P2P, on the other hand, uses structures like mesh or ring, more generically called the Distributed Hash Table (DHT), so that the search time for information retrieval is bounded. CHORD [22] and CAN [23] are examples of structured peer to peer systems which are based on the principles of the distributed hash table.

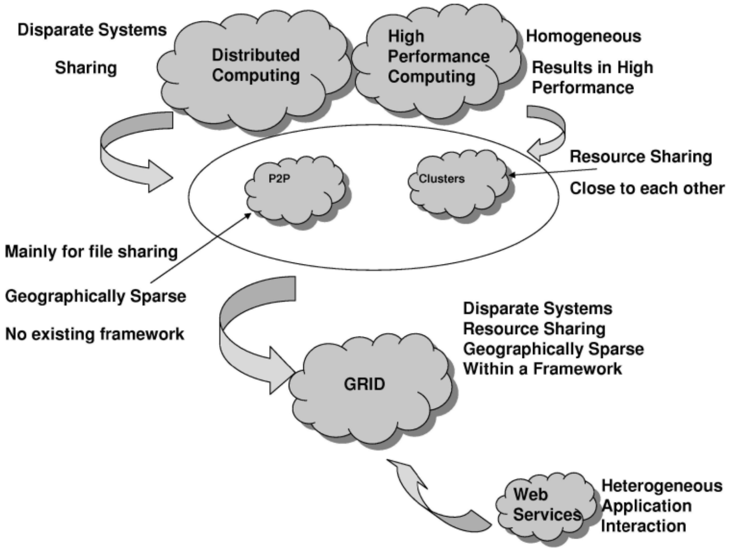


Fig. 1.1. Evolution of grid computing

It would be unfair to say that the high performance community solely contributed to the development of clusters and distributed community resulted in the development and later flourishing of the P2P systems. There was a fair amount of technical interaction between these two different communities resulting in the final evolution of P2P and clusters. Similarly, these two different technologies contributed a lot to the eventual acceptance of grid computing as a promising IT virtualization technology. In terms of concepts, grid computing combines the unique points of both P2P and clusters. Recently a new Web technology, mainly driven by the industry leaders like Microsoft®, IBM® etc., called Web services is making waves in the application inter-operability area. Figure 1.1 shows an abstract evolution of the grid computing technology from the P2P and clusters and the possible marriage of the grid with the Web services technologies. Since understanding the basics of Web services is important in the grid context, we have provided a brief summary of the Web services technologies and relevant Web services standards in the appendix.

1.2.2 Benefits of Grid Computing

In recent years, the IT infrastructure of most enterprises is facing a huge amount of stress due to the significant increase in transaction volumes. In addition, there are requirements in terms of collaboration and virtualization of resources and policies. All these business requirements drive the need and deployment of the grid in enterprises. There are mainly four distinct benefits of using grids *viz. performance and scalability*, *resource utilization*, *management and reliability*, and *virtualization.*

Performance and Scalability

Many pharmaceutical and financial enterprises are on a constant lookout for solutions which can reduce their time to market. In some cases, even a 5 - 10% improvement results in huge cost savings. Grid computing solutions of having a shared infrastructure provide more computational capabilities and increase scalability of the IT infrastructure. Most of the enterprises are therefore currently looking at the grid as a more flexible and scalable versions of their cluster infrastructure. As a result, most of the applications running on the grid infrastructure are compute intensive or batch-type applications.

Resource Utilization

Another pertinent grid imperative is the need to utilize the IT resources more efficiently. It has been found that most of the IT resources in medium to large scale enterprises are grossly underutilized. The fact is quite evident from IBM's® case study in [24] which talks about average utilization as low as 5 - 10% for PCs and around 30 - 35% for servers. Though the study had been carried out in 2000, the observations are true even today. Grid computing offers a mechanism to utilize the resources more efficiently through the process of resource sharing. A typical grid advantage of resource sharing is shown in Fig. 1.2. Let there be three clusters in an organization in three different departments as illustrated in the figure. In the absence of the grid middleware, clusters would have to be provisioned according to peak utilization. However, the loads across the clusters are not uniform and hence resource utilization can be very low. Grid middleware, on the other hand, allows the clusters to be shared and hence higher utilization can be achieved. What makes the grid really attractive for the enterprise is its ability to share resources across geography. Organizations having departments in India, Europe, and United States, can share resources as the loads across the clusters vary. A grid can harness the idle processing cycles that are available in desktop PCs located in various locations across multiple time zones. For example, PCs that would typically remain idle overnight at a company's Mumbai manufacturing plant could be utilized during the day by its North American operations.

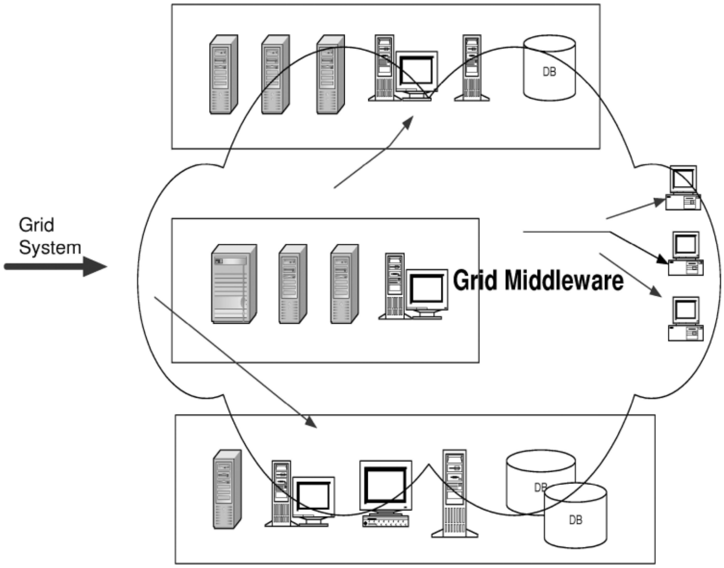


Fig. 1.2. Sharing of resources using the grid

Management and Reliability

As the IT infrastructure grows, the systems become more and more complex and heterogeneous. Therefore, the issue of management becomes extremely critical. Grid computing provides a single interface for managing the heterogeneous resources. The complexity of managing the heterogeneous resources separately is greatly reduced in such an integrated management environment. Another benefit of grid computing is that it can create a more robust and resilient IT infrastructure through the use of decentralization, fail-over and fault tolerance to make the infrastructure better suited to respond to minor or major disasters.

Virtualization

With the growth of mergers and acquisitions in the enterprise world, heterogeneity is inevitable. Heterogeneity exists in the type of hardware, storage, operating systems, and policies within the enterprises. The grid provides virtualization of heterogeneous resources resulting in better management of the resources. It is to be noted that the problem is not entirely solved. As one will find out in the subsequent chapters of the book, managing heterogeneous security policy is still a challenge and requires research attention.

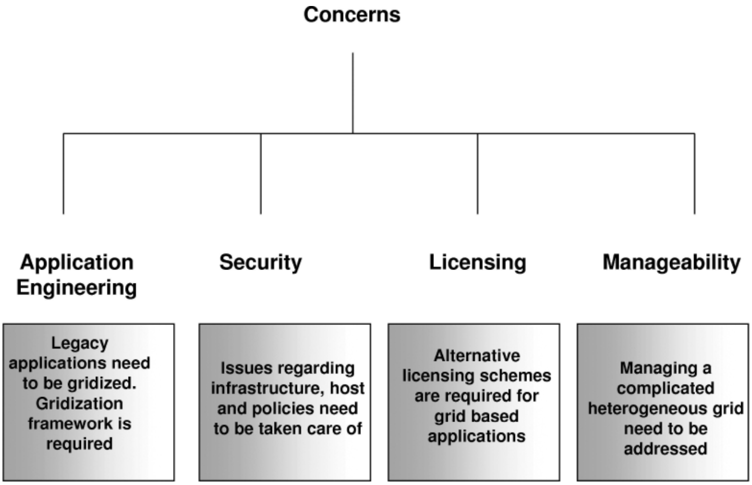

Concerns
Application Engineering
Security
Licensing
Manageability
Legacy applications need to be gridized. Gridization framework is required
Issues regarding infrastructure, host and policies need to be taken care of
Alternative licensing schemes are required for grid based applications
Managing a complicated heterogeneous grid need to be addressed

Fig. 1.3. Grid computing concern areas

1.2.3 Grid Computing Issues and Concerns

As can be inferred from the previous subsection, grid computing is a technology of enormous promise. There are a large number of technology providers, users, and academicians who are working at different levels of the grid computing stack to make the technology usable and ubiquitous. If we look back and take the position of an observer we will find that there is quite a lot of similarity between the evolution of grid technology and another widely used distributed technology called the Internet. Both Internet and grid computing were first researched on by the academic community, later handed over to the research labs and finally industry has taken over in the use and adoption of the technology. Will grid computing become as ubiquitous and widely adopted as the Internet technologies? The answer to the question lies in answering a few concerns that clog the mind of the users of grid technologies. As shown in Fig. 1.3, the concerns are: *application and data engineering*, *security, manageability,* and *licensing* issues. The adoption of grid computing in a big way depends on how the researchers and practitioners answer these critical concerns.

Application and Data Engineering

Though grid computing is more than just a technology to abet high performance computing, most of the early adopters of the grid are users in the areas where there are huge amounts of data and computation involved like life sciences, finance, automotive and aerospace, energy etc. Most of the users have been using applications in high performance clusters or in some cases SMPs and find grid computing an excellent opportunity to move their applications to a cluster of clusters or a combination of cluster and desktops or PCs available in the organization to get the performance benefits without putting in too much investment. The questions that immediately come to their mind are: “Will my Cobol application run on the grid?” “Will I get performance benefits?” “What is my return on investments?” There is a significant dearth of tools, frameworks, platforms, analyses which can immediately answer these questions or in other words there are no tools, frameworks, or platforms to help users gridize their applications. Gridization has two aspects: (i) Data can be manipulated, striped across the grid for enhanced performance. There are applications in life sciences like BLAST [25] and in other domains where this type of technique will be useful. We call this *data engineering*. (ii) Another aspect involves manipulating the applications themselves so that they are able to extract maximum benefit out of the grid computing infrastructure. We call this *application engineering*.

In the data engineering space, where application data is split across the grid for enhanced performance, there are tools and technologies [26,27] which partially achieve this. However most of these solutions work on flat data files where there is no/minimal interaction between the different data components. The application engineering space, on the other hand, is relatively bare. Some of the well-known work includes Parallax [28], PYRROS [29], P-Grade [30], CASCH [31], etc. These tools do provide some mechanisms for writing parallel applications or analysis of applications. However, most of these tools and techniques are insufficient for enterprise needs. Typically, enterprise applications are complex and have several business level, application level, and temporal dependencies, which these tools do not handle very well. Therefore, significant research and development efforts need to be undertaken in this direction to develop tools for enterprise applications. Even tools which would be able to analyze and provide hints whether the application(s) are gridizable would be greatly appreciated by the enterprise grid community. Some efforts are undertaken in this direction by ASPEED® and Cornell Theory Center (CTC) [32,33].

Grid Manageability

From its inception, IT systems were besotted with problems like scheduling, management, security, and other challenges. To solve these problems, substantial work has been carried out at different levels, for example in the form of infrastructure management systems, job schedulers, mechanisms for implementing security, etc. One class of such systems is being used in industry and it consists primarily of proprietary solutions which combine technological elements like Web services, J2EE, distributed computing, and others to solve specific business problems. Another class is that of initiatives in the academic community – mainly through joint efforts of universities around the world which take elements of freely available grid middleware and build upon them to fulfill their specific needs. All this has happened in the midst of evolving standards for grid computing, for example the development of OGSA and the ongoing evolution of the WS set of standards, including WS-RF and WS-Agreement.

It is becoming clear that one key concern which this evolutionary growth of the technology has resulted in is complexity, and the ensuing problem of *manageability*. Manageability is often cited as one of the key issues in any real world grid implementation [34]. The problem of manageability is closely related to that of integration. Since grids bring together software components, frameworks, middleware and hardware elements, integrating them together often entails gluing together systems which may not be designed and developed with that in mind.

Grid Licensing

Large scale information technology systems are undergoing transformational changes in the wake of technological developments and their adoption in scientific and business applications. Software-intensive systems are increasingly being developed using service orientation and virtualization of resources, as evident in the growth and adoption of Web services and grid computing technologies. While software architectures and products are evolving rapidly to realize these visions, the very way in which software is priced and licensed is not aligned with this new reality. The grid allows the sharing of resources across different systems. For custom defined applications this vision holds good. However, for vendor applications the gain in sharing of resources is offset by the licensing needs of the applications sharing the resources.

The intriguing part about the whole pricing and licensing equation in the grid is the technology and business nature of it. The technology challenge is there to develop suitable pricing and licensing infrastructure. Extensive research is being carried out to develop such systems. However, the challenge also lies in the business nature of the problems and hence application providers like Microsoft®, Oracle®, Sun®, IBM®, and others have to develop models so that the grid vision can be realized more effectively. Slowly but surely, we are seeing the adoption of a more flexible model for the grid system. Sun® Microsystems have come forward with their vision of charging the users 1$ for using the grid resources. With the growth of Grid systems, we would probably see the emergence of a flexible licensing and pricing model in the line of transport business.

Grid Security

Lastly, we come to the issue of security. In addition to the typical security challenges like authentication, confidentiality, and integrity, the grid offers several other unique security challenges. Policy integration, authorization, credential management related issues are unique mainly due to the typical heterogeneous nature of the grid systems. Additionally, the integration issues and the evolving standards make the problem challenging for researchers. Keeping in mind the importance of the security issues in the grid, the whole book is dedicated to provide insights into grid security issues, challenges, and solutions.

1.3 About the Book

This book aims at generating awareness regarding the different issues of security among the researchers and practitioners of grid computing. The book does not assume that the reader is an expert in security or grid computing technologies. However, some prior knowledge about general security principles and/or grid computing technologies will be required to understand the chapters covering advanced security issues in grid computing. The book has a brief and concise background on grid computing security which would be used in explaining the specific issues of grid computing security in later chapters. A small primer on Web services security and grid computing standards like OGSA is also provided in the appendix for readers who are new to this field.

1.3.1 Target Audience

The book aims at covering most of the important issues, challenges, research and deployed solutions, and research roadmaps that are needed for taking the grid to the enterprises. The book is aimed at providing benefits to researchers as well as professionals working on different aspects of grid computing through the insights about the solutions and the issues. Through this book professionals working on grid computing would be made aware of the security requirements. It would also enlighten them about the security features about some existing open source as well as some proprietary products. The book therefore would be able to provide them with information which would be useful for making important business decisions. Graduate students working on grid computing security would be able to get all the information about the different aspects in a single place. The book will also provide pointers for further research directions which will benefit students who are looking for potential research areas. Experienced researchers in the field of grid computing will be able to get a comprehensive overview of different security issues in grid computing. The book aims at providing insights into the different solutions and issues which would help experienced researchers take important research decisions.

1.3.2 Organization of the Book

The book is organized as follows: Chap. 2 provides an overview of generic security technologies which are useful in the area of grid computing also. Chapter 3 provides a taxonomy of all the issues and a solution sketch. The chapter gives readers a high-level view of the grid security area and the scope therein. Chapters 4 to 11 talk about issues and solutions pertaining to the different components of grid computing security. Grid computing security issues can be broadly categorized into three main types: architecture related issues, infrastructure related issues, and management related issues. Information security, authorization, and service security are discussed in Chap. 4, 5, and 6, respectively. The different components of the infrastructure issues are host and network related issues which are dealt with in Chap. 7 and 8 respectively. Credential management, trust management, and monitoring issues form the general security management issues, which are discussed in Chap. 9, 10, and 11, respectively. Chapter 12 provides a couple of case studies related to grid security, and Chap. 13 summarizes the book with a brief description about future technologies.

2 Overview of Security

2.1 Introduction

In the previous chapter we have provided an overview of grid computing. In this chapter we will provide an overview of the computer security technologies, protocols, and principles. Since computer security particularly cryptography is a discipline by itself, in this chapter we will only concentrate on protocols, principles, and technologies which are applicable to the grid computing area. The motivation of this chapter is to provide readers with a "one stop shop" on security which would be referred to later during the description of grid computing security. For more inquisitive readers substantial references will be provided.

2.1.1 Characteristics of Secure System

Let us assume that Alice is in the Human Resources (HR) department of an organization and Bob is an employee of the same organization. Alice is writing an official letter informing Bob about his promotion. What are the aspects Alice would be concerned about? She would be concerned that some unauthorized adversary would be able to access the contents of the letter. We describe this as *confidentiality* of the secure system. Alice achieves confidentiality by putting the letter in Bob's mailbox. Since only Bob has the key to the mailbox, nobody has access to the letter. As a receiver of the letter, Bob is concerned about three aspects. Firstly, the letter has actually been written by Alice. Bob assures himself by checking Alice's signature. We call this the *authentication* characteristic of the secure system. Secondly, if Alice had delegated somebody to put the letter in Bob's mailbox, then Bob is concerned that the other person may have tampered with the letter. To make Bob feel assured, Alice puts a seal on the letter. If anybody opens the letter the seal would be broken and Bob would immediately come to know that somebody had tampered with the letter. This is called *message integrity*. Thirdly, Bob can be concerned that Alice may later say that she had not given him that letter. However, since the letter has been signed, Alice cannot deny writing such a letter. This is called the *nonrepudiation* requirement of a secure system. Nonrepudiation works at different levels. Let us assume Alice delegates the responsibility of dropping the letter to Charlie. However, Charlie loses the letter. If Charlie had signed in a register mentioning that he picked up the letter then he cannot later say that he was not assigned the responsibility. This simple example illustrates that every day we have confidentiality, authentication, integrity, and nonrepudiation embedded into our systems. To make a computer system secure, all the above characteristics need to be followed. The characteristics are achieved in the same way as described in the above example, only the mechanisms are different.

2.1.2 Security Threats

Now let us discuss some common security threats that any system should be concerned about. Let us revert back to the above example where Alice writes a letter of promotion to Bob. Let us also assume that Bob has an adversary Derrick who wants to tamper with the process of Bob getting the promotion letter. What can Derrick do? He can prevent the letter from reaching Bob's mailbox by stealing the letter from Charlie. In this case, he resorts to *interruption*. Though different mechanisms can be employed, interruption is the most difficult threat to prevent both in computer and noncomputer systems. Another malicious activity that Derrick can resort to is to read the contents of the letter. If he is able to do that then he has resorted to *interception*. Interception does not always imply interruption. Derrick may actually read the contents, reseal the letter, and then put the letter in Bob's mailbox. He may be able to do this without raising any alarms. The third type of threat that any system would try to protect itself from is *fabrication.* In this case, Derrick writes a fabricated letter and puts the letter in Bob's mailbox. Derrick may not fabricate a new letter but may *modify* the contents of the old letter. Another type of threat which some systems may be concerned about is called *replication*. In this type of threat, the attacker does not modify or fabricate any message but replays or *replicates* the old message. If the above example, if the letter contains one line "Bob, you have been promoted," signed by Alice without any date, then Derrick can store the previous year's letter and put it in Bob's mailbox this year. One may find this type of letter unreal and the threat impractical. However, this type of threat is really dangerous in many computing system where some amount of synchronization is required based on the received message. One such example is routing updates and creation of routing tables in the Internet scenario.

Table 2.1. Threat characteristics mapping

Security threat	Characteristics violated
Interruption	Availability
Interception	Confidentiality
Modification	Integrity, confidentiality, availability
Fabrication	Availability, authentication
Replication	Authentication, availability

In Table 2.1, mapping between different threats and characteristics is provided. When an adversary interrupts a message then the receiver is not able to receive the message resulting in loss of availability for the receiver. In case of interception, the characteristic violated is confidentiality as the adversary is able to read the contents of the message which may be confidential in nature. When an adversary modifies the contents of a message then the characteristics violated are integrity, confidentiality, and availability. Since the adversary modifies the content of the message integrity of the message is violated, and if the adversary is able to read the contents of the message confidentiality is violated, while availability is violated because the receiver receives the wrong message. Fabrication violates availability similar to modification and authentication as the sender of the message is an adversary who poses as somebody else. Similarly, replication violates authentication and availability.

2.2 Different Encryption Schemes

As mentioned earlier, encryption involves hiding the actual information so that any unintended receiver would not be able to decipher the contents of the information. Traditionally this has been carried out by performing a mathematical operation on the part of the information using a *key* which is a set of characters known only to the sender and the intended receiver. One of the earliest forms of such encryption was developed by the famous Roman emperor Julius Caesar where each character of the information was transformed to another character based on addition operation with a specific key. This simple encryption technique is popularly called the *Caesar Cipher* [35]. Many variations of this simple technique have been developed over the years. Another such technique is called the one time pad where the key length is very large and is equal to the size of the message to be encrypted. This can be implemented by having the characters of any particular book as a key. This technique was employed on a wide scale in World War II. Other than this transformation technique, transposition techniques are also used where the order of the characters are changed to deceive the unintended receivers. One implementation of such a technique would be to arrange the characters in equal sized rows and then interchange the rows with the columns. It can be shown that while transformation techniques aim at creating *confusion*, transposition techniques aim at creating *diffusion*, which means that in transposition techniques two changes in the same message will result in changes in different positions of the message. More recent techniques like the Data Encryption Standard (DES) [36], Advanced Encryption Standard (AES) [37], etc. combine the above mentioned two techniques to generate both confusion and diffusion.

Encryption schemes may be broadly categorized into two main types: *symmetric key* encryption and *asymmetric key* encryption. In the former there is only one key which is shared between the sender and the receiver. In this type of scheme, the sender encrypts using the shared key and the receiver decrypts using the same key. Examples of such encryption schemes are DES and AES where a combination of transformation and transposition using the shared key is applied.

In the second type of encryption scheme or asymmetric key encryption, there is a pair of keys called the public key and the private key. The public and the private keys are generated through a mathematical function having the property such that when information is encrypted by the public key it can only be decrypted by the corresponding private key. Knowing the public key it would be computationally very expensive to generate the private key. So each person or a system using such an encryption scheme has its own set of public and private keys where the private key is stored secretly and the public key is disclosed to the whole world. Let us assume that Alice wants to send information to Bob. Alice should know the public key of Bob and only Bob knows his own private key. Alice encrypts the information with Bob's public key and Bob decrypts it with his private key. Examples of such schemes are RSA [38], El Gamal [39], etc. RSA is most popular among all the different techniques. It uses the principle that, if any one knows the product of two very large primes it would be computationally very expensive to determine any of the prime if the other prime is known. El Gamal, on the other hand, is based on the principle that the discrete logarithm is computationally very difficult. A high level view of the public key encryption is illustrated in Fig. 2.1.

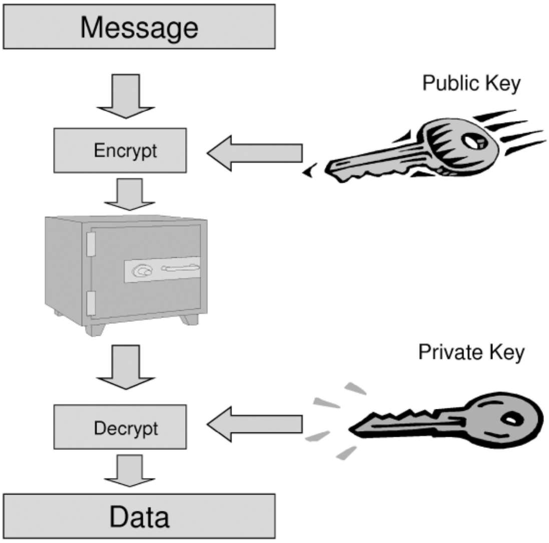


Fig. 2.1. Overview of public key encryption

Now readers may have questions in mind that why do we require two different types of encryption and what are the scenarios where each of them can be applied. Symmetric encryption schemes like the DES are computationally less intensive than the public key schemes like the RSA. However asymmetric schemes are more secure as the key need not be shared between the sender and the receiver. Therefore symmetric key schemes are always vulnerable to adversaries knowing the secret keys as several cryptanalysis techniques like differential and linear cryptanalysis [40] are available which can help someone to ascertain the key based on a sufficient amount of data. To reduce this vulnerability, the shared key needs to be changed periodically to have a secure conversation. In practical settings a combination of symmetric and asymmetric encryption mechanisms are used. Let us again take an example of Alice who wants a secure conversation with Bob. Alice creates a session key (shared key) and sends it to Bob encrypted with Bob's public key. After this phase, all conversations taking place between Alice and Bob are encrypted by the session key. This is an example where asymmetric key encryption is used to send the session key and symmetric key is used to encrypt the regular conversation using the session key. A lot of variation of the above mentioned simple technique is generally used in practice. One variation would be to generate the session key using the Diffie-Hellman [41] technique where information from both the parties is used to generate the session key. Another variation would be to use the Key Distribution Center (KDC) which is responsible for generating the session key.

2.3 Different Authentication Schemes

Let us now concentrate on the problem of authenticating a system or a user. This problem is generally handled using three different mechanisms: based on shared secret, based on public key, and based on third party.

2.3.1 Shared Secret Based Authentication

The first mechanism is through sharing a secret. For example, when I call my bank to get the details of my account they ask for a secret PIN number which is supposed to be known only to me. Most of the digital systems also work by the principle of shared secret. One way to implement such a system would be to share a password between the authenticator and the user. In this type of system, the authenticator asks the user for a password which when disclosed will allow the user to enter the system. This type of system is perhaps the most prevalent mechanism of authentication that is used. These systems are simple to implement and are computationally inexpensive because the password supplied by the user is checked with a hash of the password which is stored in some database. The checking process is simple and does not require any extra computation. However this type of system is vulnerable in two ways. Firstly the password is sent unencrypted over the wire which can be easily tapped by a malicious adversary. Therefore the password has to be encrypted by different mechanisms as mentioned earlier. Secondly, choosing the password itself is a difficult problem as automated tools are available which can guess a password with relative ease and accuracy [42]. Therefore a password based system cannot be used where strong authentication is the need of the hour. Another way to implement the shared secret would be through a challenge mechanism. Here the authenticator would challenge the user to encrypt a bit of known information by using the shared key. So the user responds by encrypting the required information and is allowed to access the system once the encrypted information has been validated by the authenticator. In this type of system the shared secret needs to be changed periodically so that the adversary cannot guess the secret. These systems are slightly more expensive than the password based system. Another vulnerability which dogs the challenge based systems is the *man-in-the-middle attack.* Let us assume that Alice is the user and Bob is the authenticator and Charlie is a malicious adversary. When Bob challenges Alice, Charlie grabs the message and sends the same message to Alice. Alice sends the answer to the challenge; Charlie taps those and sends it to Bob. Now Bob will think that it has authenticated Alice and would allow Alice to access the system. In realty it is Charlie who is accessing the systems as he is acting as the man in the middle between Alice and Bob. This vulnerability prevents the challenge based systems to be the sole mechanism of authentication and is generally used in conjunction with the other mechanisms which will be discussed subsequently.

2.3.2 Public Key Based Authentication

Going back to the example of the HR manager sending the letter of promotion to the employee, how does the employee believe the contents of the letter? When the employee looks at the letter he finds that the letter is signed by the HR manager. Since he can always verify the signature of the HR manager he can always determine the authenticity of the letter. Public key based authentication uses the principle described above. In this type of scheme the user has a public and private key pair and the authenticator knows the public key of the user. The user encrypts standard information with his/hers private key. The authenticator can verify the authenticity by decrypting the same information with the user's public key. This type of mechanism is very secure and is generally tamper proof. The biggest problem in adopting the system in wide scale is the scalability of the system. Take a scenario where there are millions of users using a particular computing infrastructure and there is the need for authentication of each and every user. This happens in case of a Website where not only there are millions of users but also the nature of the users is transient. Many of the users may be home users who do not possess a public-private key pair. It is also difficult for the authenticator to maintain the public information of so many users. So in reality a variation of this scheme is used which is called the certificate based system or third party authentication schemes.

2.3.3 Third Party Authentication Schemes

When a person tries to enter a new country, the immigration department of the country mandates that the person possesses valid passport and visa to enter the country. In this case, the immigration department does not know the person entering the country. However the department believes some third party like the person's own country issuing the passport and the consulate issuing the visa. This is a classic case of third party authentication where the authenticator does not know the user, however uses a third party credential (in this case passport/visa) for authentication purposes. In digital systems also this type of authentication is very popular. Here the user gets a digital certificate from a Certificate Authority (CA) which is a known third party. Certificates are nothing but information about the user hashed and then signed by the CA's private key. Since the public key of the CA is widely known therefore the authenticator has no problem in validating the certificate and hence authenticating the user to access the system based on the certificate. However this type of system mandates that each user has a public key which can be validated by the Certificate Authority. This means that there is a need for Public Key Infrastructure (PKI) to make the above scheme work. This may not be feasible always, especially in the Internet scenario. Another mechanism of third party based authentication used in the Kerberos system is to have a key distribution center (KDC) which authenticates the user using a standard mechanism like using a password. The KDC generates a session key for the user to access the system encrypted with the systems public key. More details about the Kerberos system will be provided subsequently.

2.4 Different Integrity Schemes

Let us now look at the problem of message integrity where the contents of the message may be changed by a malicious adversary. It is like sending an open and signed letter which is being tampered by an adversary in the middle where the signature is kept intact. This type of attack would work even if the message is encrypted because in most of the cases the purpose of the attacker is to mislead and confuse the receiver. Therefore changing the contents of the message may confuse the receiver though the attacker may not be able to understand the contents of the message. This problem of message integrity is generally solved using two mechanisms: *Message Authentication Code (MAC)* and *Keyed MAC.*

2.4.1 Message Authentication Code (MAC)

MAC is based on the principle of hash functions. Hash functions are one way functions which when applied to messages will result in a shorter message or hash. Since it is a one way function there is no way in which the main message can be recreated from the hash. When the hash function applied to two messages results in a same hash it is called a collision. Good hash functions should be able to minimize collision as much as possible. This hash or MAC (sometimes referred to as Message Digest) is appended to the message and sent to the receiver. Therefore if anyone tampers with the message the hash sent with the message will not match with the hash of the tampered message. On receiving the message, the receiver would be able to ascertain that the message has been tampered with. Hence the receiver can drop the message or can ask the sender to re-send the message. This scheme is constrained by the fact that anyone can generate the MAC by knowing the message. Therefore the scheme can only be employed if both the message and the hash are encrypted. MD5 [43] and SHA-1 [44] are examples of such a scheme.

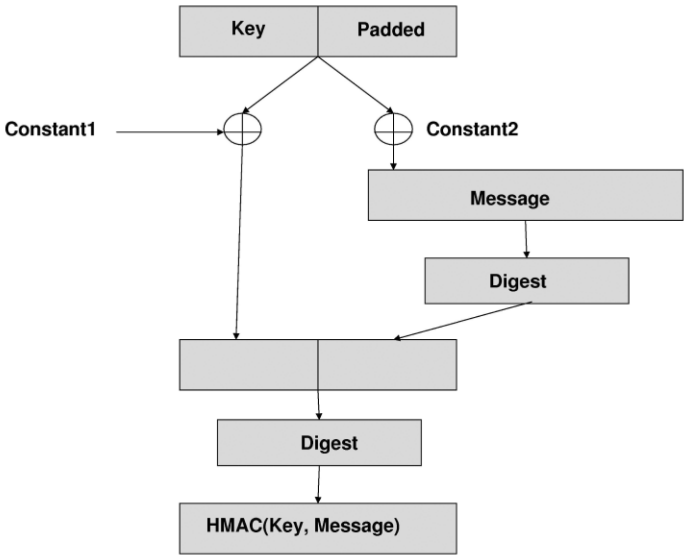


Fig. 2.2. Overview of the HMAC scheme

2.4.2 Keyed MAC

To prevent the problem of the generic MAC, a keyed MAC is generally used. In this case, two stages are employed to compute the final MAC which is not only dependent on the message but also on a key which is known only to the sender and the receiver. The key is first converted to a fixed length string using some simple functions like XOR and the message is hashed using a common hash function. Now the hashed message and the string are combined and hashed again using the hash function. The final hash is the MAC which is dependent on both the message and the key. Different algorithms will use variations of the above scheme and they may also run the steps multiple times to make the final MAC very secure. HMAC [45] (see Fig. 2.2) is an example of Keyed MAC.

2.5 Standard Protocols

In this section we will concentrate on some of the standard protocols and mechanisms used in grid computing systems. We will discuss public key infrastructure, Secure Socket Layer (SSL), Kerberos, and IP security as part of this section.

2.5.1 Public Key Infrastructure

An infrastructure which supports the public key based authentication and encryption is called the Public Key Infrastructure (PKI). As the name suggests, each entity user in the PKI environment possesses a public and a corresponding private key. At the heart of the PKI lies the concept of certificates, which are used to validate the user and the public key associated with the user, and Certification Authority (CA), who issue these certificates. A certificate validation process is illustrated in Fig. 2.3.

Certificates

Certificates are credentials of a specific user containing the user details which is signed by the CA. Different types of certificates are available. The most popular and commonly used certificate format is called X.509 format [46]. A typical X.509 certificate consists of the following information:

- The version of X.509 that has been used.
- The information about the user or the issuing CA.
- The algorithms used to compute the signature of the certificate.
- The subject whose public key is being certified.
- The validity of the certificate which indicates the time for which the certificate is valid.
- The public key information.

- The signature field which is actually a hash of the above information signed by the CA's private key.
- In addition there are some optional fields and extensions so that some customization of the X.509 certificate is possible.

If we take a closer look at the above certificate format, we find that it bears a close resemblance to a driving license or a passport issued to people in the physical world. Since the purpose of both digital and physical certificates is the same therefore this resemblance is not at all surprising.

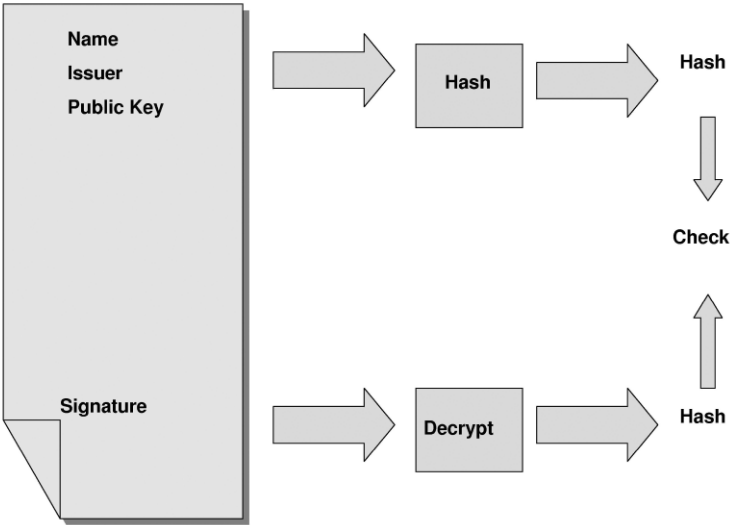


Fig. 2.3. Certificate overview

Certification Authority (CA)

Certification Authorities (CA) [47] are entities which are trusted by different systems. The CAs are responsible for certifying the public keys of different users who subscribe to the CA. There are different models of trust that are available in the PKI system namely the monopoly model, the monopoly plus RA model, and the delegated CA model.

- **Monopoly Model:** In this model, there is only one CA which is trusted by all other entities who get the certificate from the trusted CA. This is a very simple model; however it has scalability problem, especially for large systems.
- **Monopoly plus RA:** This model is similar to the monopoly model,, except that the single CA chooses other organizations (known as Registration Authorities or the RAs) to securely verify the public keys. The RAs communicate this information to the CA.
- **Delegated CA:** In this type of model, the trusted CA (called the trust anchor CA) can issue certificates for other CAs called the delegated CA. The users can then obtain certificates from the delegated CAs rather than from the trust anchor CA. This type of delegation is used in the open source Globus implementation.
- **Oligarchy:** This type of model is commonly used in browsers. Here, the different products come with a single key, configured with many CAs, and the certificate issued by any one of them is accepted.
- **Others:** There are other trust models like *Anarchy*, used in PGP, which is a distributed model where each user is responsible for configuring trust anchors or public keys of some other set of users. There are also concepts like the *Name Constraints* where the trustworthiness of CA is not a binary value, completely trusted or untrusted for everything. Rather, CA should only be trusted for certifying a set of users.

At this point the readers may ask, what happens if the private key of a user is stolen or compromised? To solve this, the CAs issue certificates with specific validity period. However, due to practical reasons the validity periods are typically in months, which is a large time for adversary to use the key for malicious activities. The problem is similar to that faced by the credit card companies. The credit cards also contain an expiration date which is in years. If the card is stolen, the credit card companies publish a list of credit cards which have been compromised so that the merchants can verify that list before accepting any card. This is also the mechanism that is followed in case of certificates. Each CA periodically publishes a list of revoked certificates in the form of a Certificate Revocation List (CRL) which is signed by the CA. Each CRL consists of a list of all unexpired revoked certificates, which is consulted by the authenticating system before accepting the user as a valid user.

2.5.2 Secure Socket Layer (SSL)

One of the most popular protocols to secure the transport layer is called the Secure Socket Layer (SSL), whose newer versions are called Transport Layer Security (TLS) [48]. SSL/TLS works on top of Transport Layer Protocol (TCP), and provides security in managing sessions over the transport channel. The SSL version 2 was deployed with Netscape Navigator 1.1 by Netscape® [49] in 1995. Netscape® came out with version 3 a few years later. The Internet Engineering Task Force (IETF) [50] extended the concept to develop a standard called Transport Layer Security (TLS).

The protocol works as follows:

- The client contacts the server to initiate a SSL/TLS session. In this step the client does not identify itself, however it mentions the set of cryptographic algorithms the client can support. In addition, the client also sends a random number R_C, which will be used to create the session key.
- The server replies by sending its certificate to the client. It also sends a random number R_S which will contribute towards the creation of session key.
- The client then verifies the certificate, extracts the public key of the server, and then selects a random number S. In addition, the client also computes K, which is the master secret computed as a function of R_C, R_S, and S.
- The client sends S and the hash of K encrypted with the server's public key.
- Subsequently, all the data sent over the SSL/TLS channel is encrypted with the session key K.

It is to be noted that the SSL/TLS protocol defined above helped the client to authenticate the server. However, the server cannot authenticate the client. As a protocol, SSL/TLS allows the option for mutual authentication, where the server can authenticate the client if the client possesses the required certificate. However, in most cases, if such an authentication is required the client generally sends its user name and password encrypted with the server's public key.

2.5.3 Kerberos

Kerberos [51] is a secret key based mechanism for providing authentication in the network. It was originally designed at MIT, based on the work of Needham and Schroeder [52]. It is one of the examples of a third party authentication scheme described earlier.

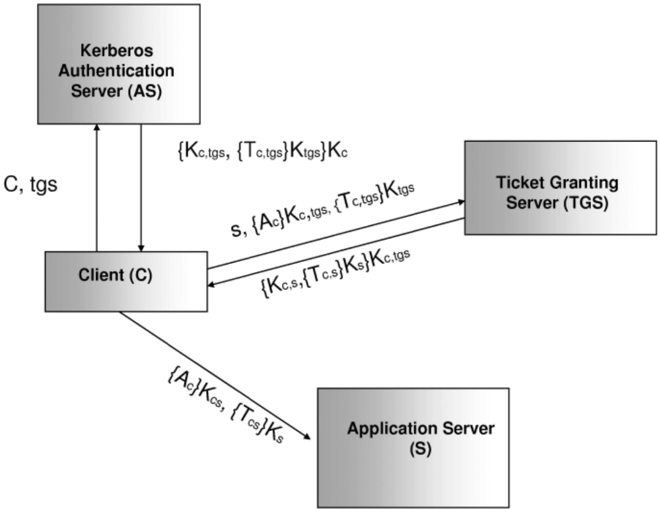

Kerberos Authentication Server (AS)
$\{K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}\}K_c$
C, tgs
Ticket Granting Server (TGS)
s, $\{A_c\}K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}$
Client (C)
$\{K_{c,s},\{T_{c,s}\}K_s\}K_{c,tgs}$
$\{A_c\}K_{cs}, \{T_{cs}\}K_s$
Application Server (S)

Fig. 2.4. Overview of the Kerberos system

Kerberos consists of a Key Distribution Center (KDC) which runs on a secure node, in the network. The KDC is composed of basically two components: the Ticket Granting Server (TGS) and the Authenticating Server (AS). Kerberos was designed to provide authentication between two entities who are trying to communicate on an insecure network. Kerberos is based on three clear design goals:

- **Minimum Exposure:** Kerberos ensures that the client passwords do not flow as a cleartext on the network. It also allows the minimum exposure of client key on workstations.
- **Containment:** This indicates that if there is a compromise, only one client or server is affected. This is also ensured through limited authentication lifetime (which is typically 8 hours to 24 hours).
- **Transparency:** Kerberos ensures that the old applications can transparently work in the new setup. Applications need to provide login facilities, which is the only change required.

Figure 2.4 shows the overview of the operation of the Kerberos system. The following steps are performed:

1. The client logs in to the Authentication Server (AS) mentioning that it wants to access the Ticket Granting Server (TGS).
2. The AS replies by providing the TGS ticket encrypted with the TGS's public key and the session key for the client to access the TGS. The whole information is encrypted by the client's public key, which can be the client's password.
3. The client sends the ticket information and the authentication information (which is typically time, client information, etc.). The client encrypts the information by the TGS's public key.
4. TGS replies with the ticket to access the server.

2.5.4 IP Security (IPSec)

The IPSec [53,54] is a method proposed to solve the attacks mentioned before through interaction with the network layer. The principal feature of IPSec that enables it to support a variety of application scenarios is that it can encrypt or authenticate all traffic at the IP level. Thus, all distributed applications, including remote login, client/server, e-mail, file transfer, Web access, and so on, can be secured. Figure 2.5 shows a typical scenario of IPSec usage. An organization maintains local area networks at dispersed locations. Traffic on each LAN does not need any special protection, but the devices on the LAN can be protected from the untrusted network with firewalls. Since we live in a distributed and mobile world, the people who need to access the services on each of the LANs may be at sites across the Internet. These people can use IPSec protocols to protect their access. These protocols can operate in networking devices, such as a router or firewall that connects each LAN to the outside world, or they may operate directly on the workstation or server. In the diagram, the user workstation can establish an IPSec tunnel with the network devices to protect all the subsequent sessions. After this tunnel is established, the workstation can have many different sessions with the devices behind these IPSec gateways. The packets going across the Internet will be protected by IPSec but will be delivered onto each LAN as a normal IP packet.

IPSec is composed of the following main components:

- Two security mechanisms: an authentication-only function, referred to as the *Authentication Header* (AH) [55], and a combined authentication and encryption function, called the *Encapsulating Security Payload* (ESP) [56], that provide the basic security mechanisms within IP.

- *Security associations (SA)* that represent an agreement between two peers on a set of security services to be applied to the IP traffic stream between these nodes.
- *Key management infrastructure* that sets up SA between two communicating peers.

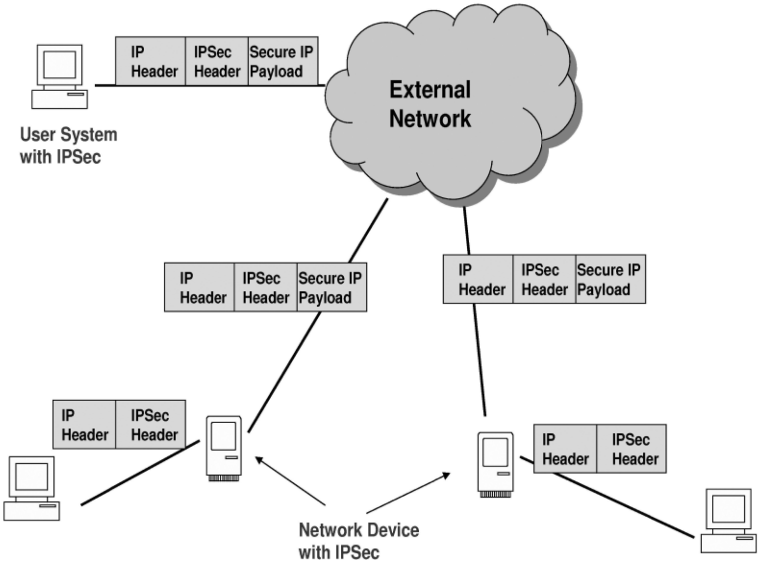


Fig. 2.5. Overview of IPSec

Both AH and ESP security mechanisms involve adding a new header to the IP packet, and the header is added between the original IP header and the layer-4 (Network Layer) header. In this way, only the two IPSec peers will have to deal with the additional headers, thus legacy routers will be able to handle IPSec packets just like normal IP packets. This feature lets far fewer IPSec-compliant devices on the Internet, thus making its deployment easier. IP AH and IP ESP may be applied alone or in combination. Each function can operate in one of two modes: transport mode or tunnel mode. With transport mode, AH or ESP is applied only to the packet payload, while the original IP packet header remains untouched. The AH or ESP header is inserted between IP header and layer 4 header, if any. In tunnel mode, AH or ESP is applied to the entire original IP packet, which is then encapsulated into a new IP packet with a different header.

For Virtual Private Networks (VPN), both authentication and encryption are generally desired, because it is important both to (1) assure that unauthorized users do not penetrate the virtual private network and (2) assure that eavesdroppers on the Internet cannot read messages sent over the virtual private network. Because both features are generally desirable, most implementations are likely to use ESP rather than AH. However, by providing both AH and ESP, IPSec provides implementers with flexibility in terms of performance and security. This flexibility is also extended to the key exchange function where both manual and automated key exchange schemes are supported.

2.6 Chapter Summary

In this chapter, we have looked at the different concepts and technologies that are useful in the grid computing context. Most secure systems look at providing confidentiality, authentication, integrity, and non-repudiation to the end users. Confidentiality is generally provided by encrypting the messages using symmetric or asymmetric encryption mechanisms. The former uses public/private keys for encryption purposes and the latter uses symmetric shared key between the sender and the receiver. Different types of authentication schemes include shared secret based authentication, public key based authentication, and third party based authentication mechanisms. The first scheme is perhaps the most prevalent one, where a shared secret or a password is used for authentication purposes. The second scheme is robust, however it may not be scalable in all purposes. The third scheme, which uses a Certification Authority (CA), or similar third parties for authenticating a user is one of the most popular authentication mechanisms used. Integrity schemes are implemented using Message Authentication Codes (MAC) and keyed MAC schemes. Both the schemes rely on hash functions for integrity checks. The latter (HMAC is an example of such a scheme) uses a key to prevent malicious users from generating the MAC. Several standard protocols like Kerberos, SSL/TLS, and IPSec have also been discussed in this chapter. In the next chapter we will provide a high level taxonomy of the grid systems to set the grid security landscape.

3 Taxonomy of Grid Security Issues

3.1 Introduction

When I started to write this book, one long time memory came rushing back. This was the period of my life which I really cherish. It was the period during my college days. During those days we used to travel long-distances by pooling together the vehicles we had. We had lots of wonderful experiences during that time which can itself be a topic of a book. Pooling helped us in optimizing the resources for every trip based on the number of people traveling and the distance to be traveled. However, it was a source of anxiety for us also. Whenever I gave my car to the pool I was worried about the car because it may not always be handled with care. In addition, we always used to have a few people for the trip who were complete strangers to me. Therefore, trust was a real issue. On the other hand, when I traveled in somebody else's car I felt anxious about my safety as most of our vehicles were at least a few decades old. My anxiety did not end here. After every trip, I used to lose a few of my favorite cassettes, CDs, books, or some of my other "valuable" possessions. Though, I am mentioning some of the anxieties of the trips, I loved them and looked forward to them. We did have our share of weird incidents. Like the one where we ran out of gas and were stranded in the middle of a desert. We also once got stranded after our keys got stolen in a hotel room. After a few of those incidents, we learnt to cope with them. We regularly used to *monitor* the gas usage of the cars, hand over the keys only to *authorized* valets, used some sort of "trust" mechanisms before inducting strangers into the group, a store for valuables with key with one of us, a check up of the vehicles to be used for the trips, and several other such mechanisms. Once these mechanisms were implemented the journey and the trip became more enjoyable as we spent time enjoying the trip rather than worrying about mundane affairs.

Once I started writing this book, I noticed an uncanny similarity between our college carpooling mechanisms and the grid system. Similar to the car pooling system, a grid system also is a mechanism to pool resources on-demand to improve the overall utilization of the system. Similarities do not end here also. The issues and concerns that we had for personal safety, trust, authorization, etc. are important issues for grid computing systems as well. For example, similar to the car pooling system where we were concerned about cassettes and CDs, in grid systems also one is concerned about the data processed. Moreover, the concerns of a user donating his/her host to the grid system are very similar to the concerns I had about my car. Similar to the car pooling system, the grid system also requires a monitoring system in place to monitor the resource usage, trust management system to create, negotiate, and manage trust between other systems or "strangers," and an authorization system to authorize the users to a access certain set of resources. In this chapter we will briefly talk about the different security issues and solutions in the grid system. However, this chapter is not meant to be comprehensive as all the components will be elaborated upon in the course of the book. This chapter would provide an overall landscape so that readers can choose the issues they are interested in.

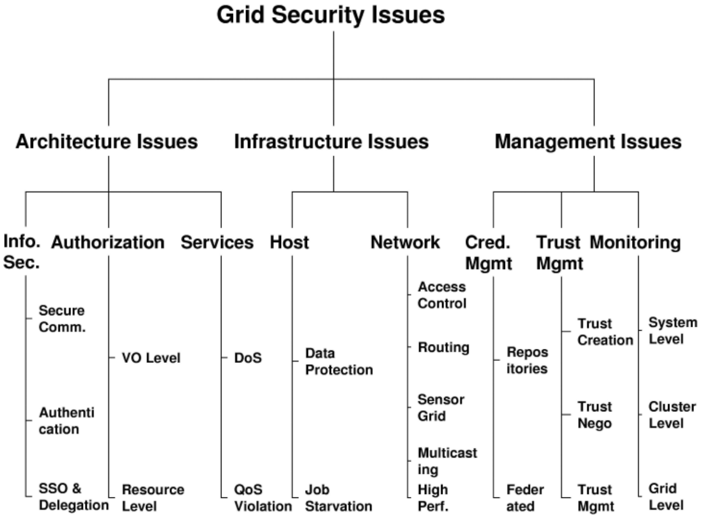


Fig. 3.1. Taxonomy of grid security issues

3.1.1 Grid Security Taxonomy

Figure 3.1 shows the categorization of the different security issues in a grid. The grid security issues can be categorized into three main categories: *architecture related issues*, *infrastructure related issues*, and *management related issues*.

Architecture Related Issues

These issues address concerns pertaining to the architecture of the grid. Similar to car pooling, where we were concerned about our cassettes and CDs, users of the grid are concerned about the data processed by the grid and hence there is a requirement to protect the data confidentiality and integrity, as well as user authentication. We categorize these requirements under information security. Similarly, resource level authorization is a critical requirement for grid systems. Finally, there are issues where users of the grid system may be denied the service of the grid or the Quality-of-Service (QoS) is violated. These fall under the purview of service level security issues.

Infrastructure Related Issues

These issues relate to the network and host components which constitute the grid infrastructure. Host level security issues are those issues that make a host apprehensive about affiliating itself to the grid system. The main subissues here are: data protection, job starvation, and host availability. A grid involves running alien code in the host system. Therefore, the host can be apprehensive about the part of the system which contains important data. Similarly, a host can also be concerned about the jobs that it is running locally. The external jobs should not reduce the priority of the local jobs, and hence lead to job starvation. Similarly, if the host is a server, it can be concerned about its own availability. There should be mechanisms to prevent the system from going down resulting in denial-of-service to the clients attached to the host.

Management Related Issues

The third set of issues pertains to the management of the grid. Managing credentials is absolutely important in grid systems because of the heterogeneous nature of the grid infrastructure and applications. Like any distributed system, managing trust is also critical and falls under the purview of management related issues. Similar to the car pooling case where monitoring of gas was mandated, grid systems also require some amount of resource monitoring for auditing purposes. Much of the information obtained from the monitoring systems is fed back to higher level systems like intrusion detection and scheduling systems.

3.2 Architecture Related Issues

Architecture level issues address the concern of the grid system as a whole. Issues like Information security, authorization, and service level security generally destabilize the whole system and hence an architecture level solution is needed to prevent those. In this section we will briefly touch upon the issues and some solutions.

3.2.1 Information Security

We define information security as the security related to the information exchanged between different hosts or between hosts and users. The concerns at the information security level of the grid can be broadly described as issues pertaining to *secure communication*, *authentication*, and issues concerning *single sign on and delegation*. Secure communication issues include those security concerns that arise during the communication between two entities. These include confidentiality and integrity issues. Confidentiality indicates that all data sent by users should be accessible to only "legitimate" receivers, and integrity indicates that all data received should only be sent/modified by "legitimate" senders. There are also issues related to authentication, where the identities of entities involved in the overall process can be accurately asserted. These are critical issues in all areas of computing and communication and become exceedingly critical in grid computing because of the heterogeneous and distributed nature of the entities involved there. In addition to the secure communication features users are also concerned about single sign on capability provided by the grid computing infrastructure. In single sign on the authentication is done once.

The information security issues exist in all fields of computing and communications and have been studied for quite some time. In the grid computing area, the researchers and practitioners have come together to create the Global Grid Forum (GGF) (now called OGF). They have released an open standard called Open Grid Standards Architecture (OGSA). There is a Grid Security Infrastructure (GSI) layer of OGSA which addresses most of the information security challenges mentioned above. The

Globus toolkit is an open source implementation of OGSA. Details about grid information security are provided in Chap. 4.

Solutions to Information Security Issues

The Grid Security Infrastructure (GSI), developed independently and later integrated as part of the OGSA standards, addresses all the stated architectural concerns. GSI is based on proven standards such as public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) and enables secure authentication and communication over computer networks. The latest version of the GSI based on Globus Toolkit 4.0 also allows Web services based security. Please see details provided in Chap. 4.

- **Secure Communication:** The GSI uses public key cryptography, as the basis for creating secure grids and SSL/TLS for data encryption. In public key cryptography, the entities generate public/private key pairs based on some cryptographically secure mathematical function. A message when encrypted by the public key can only be decrypted by the private key corresponding to the public key. The public keys are known to everyone.
- **Authentication:** A central concept in GSI authentication is the *certificate.* Every user and service on the grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service.
- **Single Sign on and Delegation:** The GSI provides a single sign on and delegation capability, which reduces the number of times the user must enter his/her pass phrase when multiple resources are used, which is common in a grid scenario. This is done by creating a proxy. A proxy consists of a new certificate (with a new public key in it) and a new private key. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is signed by the owner, rather than a Certification Authority (CA). The certificate also includes a time notation after which the proxy should no longer be accepted by others.

3.2.2 Authorization

Another important security issue is that of authorization. Like any resource sharing system, grid systems also require resource specific and system specific authorizations. It is particularly important for systems where the resources are shared between multiple departments or organizations, and department wide resource usage patterns are pre-defined. Each department can internally have user specific resource authorization also. The authorization systems can be mainly divided into two categories: *VO Level Systems* and *Resource Level Systems*. Virtual Organization or VO level systems have a centralized authorization system which provides credentials for the users to access the resources. Resource level authorization systems, on the other hand, allow the users to access the resources based on the credentials presented by the users.

Grid Authorization Solutions

Several authorization systems can be applied to the grid context.

- **VO Level Systems:** VO level grid authorization systems are centralized authorization for an entire Virtual Organization (VO). These types of systems are necessitated by the presence of a VO which has a set of users, and several Resource Providers (RP) who own the resources to be used by the users of the VO. Whenever a user wants to access certain resources owned by a RP, he/she obtains a credential from the authorization system which allows certain rights to the users. The user presents the credentials to the resource to gain access to the resource. In this type of systems, the resources hold the final right in allowing or denying the access to the users. Examples of VO level grid authorization systems are Community Authorization Service (CAS) Virtual Organization Membership Service (VOMS), and Enterprise Authorization and Licensing System (EALS).
- **Resource Level Systems:** Unlike the VO level authorization systems, which provide a consolidated authorization service for the virtual organization, the resource level authorization systems implement the decision to authorize the access to a set of resources. Therefore, VO level and resource level authorization systems look at two different aspects of the grid authorization. In Chapter 5, we have provided details of different resource level authorization Systems like Akenti, Privilege and Role Management Infrastructure Standards Validation (PERMIS), and the GridMap system.

3.2.3 Service Security

One of the most important security threats existing in any infrastructure is the malicious service disruption created by adversaries. Many such examples exist in the Internet space where servers and networks are brought down by a huge amount of network traffic and users are denied the access to a certain Internet based service. Since grid computing deployment has not reached the "critical mass" yet, the service level attacks are also currently nonexistent. However, with the grid computing poised for a huge growth in the next few years, this area should be looked upon with utmost concern by the grid security experts. The grid service level security issues can be further subdivided into two main types: *QoS Violation Issues* and *Denial-of-Service (DoS)* related issues. The first issue is about the forced QoS violation by the adversary through congestion, delaying or dropping packets, or through resource hacking. The second one is more dangerous where the access to a certain service is denied. More details about the attacks and solutions are provided in Chap. 6.

Solutions to Service Attacks

It is to be noted that the DoS attacks and QoS violation attacks are research topics for researchers in the areas of networks, services, and operating systems. In Chap. 6, we provide an overview of different research efforts that are being undertaken and the solutions that have been proposed.

- **DoS Solutions:** The solutions proposed for Denial-of-Service (DoS) attacks can be categorized into mainly two types: *preventive* solutions and *reactive* solutions. Preventive solutions try to prevent the attack from taking place by taking precautionary measures. Reactive solutions, on the other hand, react to a DoS attack and are generally used to trace the source of the attack. Some examples of preventive solutions are filtering, throttling, location hiding, and intrusion detection. Examples reactive solutions include logging, packet marking, Link testing, and others.
- **QoS Violation:** This is an active area of research and several architecture and solutions have been proposed. Most of these solutions rely on some amount of monitoring and metering systems which try to detect the QoS levels of the systems and then make decisions to raise the alarms. The WATCHERS project is an example of such a system. More details of this project and a grid accounting system are provided in Chap. 6.

3.3 Infrastructure Related Issues

A grid infrastructure consists of grid nodes and the communication network. The security issues related to the grid infrastructure are also of paramount importance.

3.3.1 Host Security Issues

Host level security issues are those issues that make a host apprehensive about affiliating itself into the grid system. The main subissues here are: data protection and job starvation. Whenever a host is affiliated to the grid, one of the chief concerns is regarding the protection of the already existing data in the host. The concern stems from the fact that the host submitting the job may be untrusted or unknown to the host running the job. To the host running the job, the job may well be a virus or a worm which can destroy the system. This is called the *Data protection issue*. *Job starvation* refers to a scenario where jobs originating locally are deprived of resources by alien jobs scheduled on the host as part of the grid system.

Solutions to the Host Security Issues

Several solutions have been proposed for data protection and job starvation issues.

- **Data Protection:** Solutions in this space use isolation to restrict the data to the grid or external applications. In Chap. 7 we discuss several isolation techniques *viz.* application level sandboxing, virtualization, and sandboxing. The first type of solution is through the use of proof carrying code (PCC) where the code generators generate proofs of application safeness and embed those in the compiled code. The second solutions looks at creating Virtual Machines (VM) on the physical machine resulting is strong isolation properties. The third type of solution, or the sandboxing solutions, traps system calls and sandboxes the applications to prevent them from accessing data and memory based on certain policies.
- **Job Starvation:** Different solutions which look at the problem of job starvation can be categorized as *advanced reservations* and *priority reduction* techniques. Under advanced reservation system, a user requests a set of resources (can be CPU, memory, disk space, etc.) for a specified amount of time for the set of

jobs to be run. The resources are booked based on the availability, security, QoS and other metrics. Once the resources are booked, the resource providers honor the contract and have every right to terminate the job once the contract expires. These techniques require schedulers to work hand-in-hand with the resources/hosts providing service to the end users. Priority reduction techniques, on the other hand, reduce the priorities of the long running jobs to reduce the possibility of starvation. Most of the solutions in this space are ad hoc in nature and look at specific solutions

3.3.2 Network Security Issues

In the context of grid computing, network security issues assume significant importance mainly due to the heterogeneity and high speed requirements of many grid applications. Moreover the grid inherits some of the generic network issues also. *Access control and isolation* are important requirements for traffic flowing through the grid networks. In this area, integration of grid technologies with VPN and firewall technologies assume significance. *Routing* of packets in networks based on routing tables is a specific network issue. Attacks in routing include link and router attacks which may cause significant destruction. Many of the issues still require research attention. *Multicasting* is an efficient means of information dissemination and may assume importance for grid networks in the future. Member authentication, key management, and source authentication are specific security issues in multicasting. Another topic of interest in grid networks is the integration of *sensor networks* with grid technologies. Several sensor network attacks like sybil attacks, wormhole, and sinkhole attacks, node hijacking, need to be tackled before the sensor grid vision can get realized. Finally, there are security issues in high performance interconnects.

Solutions to the Grid Network Issues

Many of the grid network issues are active areas of research where solutions are mostly developed in labs and not yet commercialized. In Chap. 8, we have included the research activities in many of these areas.

- **Access Control & Isolation:** Many of the grid and Web services solutions cannot work effectively with firewalls and virtual private networks (VPN) which have become ubiquitous in today's enterprises. The area requires significant research efforts. Some of the research efforts like Adaptive Grid Firewalls (AGF) and Hose have been included in Chap. 8.

- **Secure Routing:** This area of research is inherited from the traditional networking area. Most routing protocols use digital signatures and passwords for message exchange which do not solve the advanced attacks like source misbehavior. More research is needed in this area. Some topics like inconsistency detection are briefly touched upon in our discussion in Chap. 8.
- **Secure Multicasting:** This has been an active area of research for the last few years. Most of the solutions presented in this area are research outputs and rarely implemented in a large scale. However solutions like centralized and hierarchical member authentication systems, tree-based, and core based key management systems, and stream signing, and chaining type solutions are important and require mention. Details of the different techniques are provided in Chap. 8.
- **Sensor Grids:** Security in sensor networks is a very important issue due to the computational constraints imposed by the devices and network and bandwidth constraints. This is also an active area of research and several solutions have been proposed like SPINS and TinySec.
- **High Speed Networks:** One of the most important issues in the adoption of security solutions is performance. A security solution which requires firewall/intrusion detection, encryption/decryption, message authentication, distributed denial of service (DDoS) attack protection, etc. results in a significant overhead which significantly reduces the performance. We have discussed some hardware based solutions like CYSEP and protocol level solution like Infiniband Security in Chap. 8.

3.4 Management Related Issues

If we go back to the car pool example, we find that management was necessary there. Similarly, the grid management is important as the grid is heterogeneous in nature and may consist of multiple entities, components, users, domains, policies, and stake holders. The different management issues that grid administrators are worried about are credential management, trust management, and monitoring related issues.

3.4.1 Credential Management

Management of *credentials* becomes very important in a grid context as there are multiple different systems which require varied credentials to access them. Credential management systems store and manage the credentials for a variety of systems and users can access them according to their needs. This mandates for specific requirements from the credential management systems. For typical grid credential management systems mechanisms should be provided to obtain the initial credentials. This is called the *initiation* requirement. Similarly, secure and safe *storage* of credentials is equally important. In addition, the credential management systems should be able to access and renew the credentials based on the demand of the users. A few other requirements which are important for grid systems are *translation*, *delegation*, and *control* of the credentials. Based on the above requirements, credential management systems are mainly of two types: *credential repositories* or credential storage systems, and *credential federation systems* or credential share systems. The first set of systems are responsible for storing credentials while the second set of systems are responsible for sharing credentials across multiple systems or domains.

Different Credential Management Systems

Different types of credential repositories and credential federation systems have been developed. In Chap. 9, we provide a detailed account of some of the important systems which are useful from the grid context. The two systems are not competitive, rather complementary in nature.

- **Credential Repositories:** The basic purpose of credential repositories is to move the responsibilities of credential storage from the user to these systems. Some of the examples of credential repositories are smart cards, virtual smart cards, and MyProxy Online Credential Repositories. Smart cards are credit card sized tokens which contain the secret keys of the users. These are extremely secure, however they are expensive. Virtual smart cards embed the features of smart cards in the software where the keys never leave the user's system. MyProxy is a popular implementation of credential repositories specifically for grid systems.
- **Credential Federation Systems:** These systems, protocols, and standards are used for managing credentials across multiple systems, domains, and realms. A few of the examples in this space include VCMan, which is a specific solution for grid and Community Authorization Service (CAS) for inter-operability across

multiple domains. KX.509 is a protocol which provides interoperability between X.509 and Kerberos systems. A standard called the Liberty Framework has been developed by a consortium of 150 companies for creating and managing federated identities. Another popular open source solution in this space is Shibboleth.

3.4.2 Trust Management

Another important management issue which needs to be addressed is the issue of managing trust. Managing trust is not unique to digital or computing systems; it is used everyday and in every sphere of life. Trust is a multi-dimensional factor which depends on a host of different components like reputation of an entity, policies, and opinions about the entity. Managing trust is crucial in a dynamic grid scenario where grid nodes and users join and leave the system. Therefore, there must be mechanism to understand and manage the trust levels of systems and new nodes joining the grid. The trust life cycle is composed of mainly three different phases: *trust creation phase*, *trust negotiation phase*, and *trust management phase*. The trust creation phase generally is done before any trusted group is formed, and it includes mechanisms to develop trust functions and trust policies. Trust negotiation, on the other hand, is activated when a new untrusted system joins the current distributed system or group. The third phase, or the trust management phase, is responsible for recalculating the trust values based on the transaction information, distribution or exchange of trust related information, updating and storing the trust information in a centralized or in a distributed manner.

Trust Management Solutions

Trust management is an active area of research and several trust management systems have been proposed and implemented in a limited manner in the labs of different universities. The main characteristics of trust management systems are scalability, reliability, and security. In other words, the trust management systems should scale in terms of message overheads, storage, and computational overheads, should be reliable in face of failures, and should be secure against masquerade attacks, collusion, and sybil attacks. The different trust management systems can be broadly categorized into reputation based and policy-based trust management systems.

- **Reputation Based:** These types of systems are based on trust metrics derived from local and global reputation of a system or an entity. As part of the discussion in Chap. 10 we discuss the different reputation-based systems including PeerTrust, XenoTrust, NICE, Secure Grid Outsourcing (SeGO) systems.

- **Policy Based:** In policy based systems, the different entities or components constituting the system, exchange and manage credentials to establish the trust relationships based on certain policies. The primary goal of such systems is to enable access control by verifying credentials and restricting access to credentials based predefined policies. These types of system create a policy based trust language. Examples of such systems are PeerTrust Trust Negotiation and TrustBuilder.

3.4.3 Monitoring

Monitoring is the third and one of the most crucial management issues that needs to be tackled in a grid scenario. Monitoring of resources is essential in grid scenarios primarily for two reasons. Firstly, different organizations or departments can be charged based on their usage. Secondly, resource related information can be logged for auditing or compliance purposes. The different stages of monitoring are: *data collection*, *data processing*, *data transmission*, *data storage*, and *data presentation*. The data collection stage involves collecting data through different sensors located at different collection points. The gathered data can be static in nature like network topology, machine configuration, or dynamic like CPU and memory utilization, system load, etc. The Data processing stage processes and filters the data based on different policies and criteria from the data collected from the sensors. The Data transmission stage involves the transmission of collected and processed data to the different entities interested. Transmission involves sending the data in a format understood by other parties over a transmission medium, for example the network. There may be a need for storage of gathered or processed data for future references which is carried out in the data storage stage. Finally, the data presentation stage presents the data in a format understood by the different interested entities.

Different Monitoring Systems

Different monitoring systems available can be broadly categorized into system based, cluster based, and grid based monitoring systems. In Chap. 11, we provide details of different monitoring systems.

- **System Level:** The system level monitors collect and communicate information about standalone systems or networks. For network monitoring Simple Network Management Protocol (SNMP) is an example for managing and monitoring network devices. Examples of open source and popular system monitoring tools include Orca, Mon, Aide, Tripwire, etc.
- **Cluster Level:** The cluster level monitoring systems generally are homogeneous in nature and require deployment across cluster or a set of clusters for monitoring purposes. Popular examples of cluster level monitoring systems include Ganglia from University of Berkeley and Hawkeye from University of Wisconsin Madison.
- **Grid Level:** Grid level monitoring systems are much more flexible than other monitoring systems and can be deployed on top of different other monitoring systems. Many of the grid level monitoring systems provide standards and interfaces for interfacing, querying, and displaying information in standard formats. Examples of such monitoring systems include R-GMA, Globus Monitoring and Discovery Systems (MDS), Management of Adaptive Grid Infrastructure (MAGI), and GlueDomains. R-GMA combines the grid monitoring and information services with relational models. MDS is a Globus component for monitoring and discovering resources while MAGI is a grid management and monitoring system. GlueDomains is used mainly for network monitoring. Details of the different systems are available in Chap. 11.

3.5 Chapter Summary

Grid computing is an interesting and a high potential solution for most enterprises. However, security is one of the major impediments in widespread grid adoption. In this chapter we have provided a high level taxonomy of the grid systems. We have categorized the issues pertaining to grid computing security into three main buckets *viz.*, architecture related issues, infrastructure related issues, and management related issues. Architecture related issues are concerned with the overall architecture of the grid system like the concerns pertaining to the information security, concerns about user and resource authorization, and issues pertaining to the overall service offered by the grid system. The infrastructure related issues are concerned with the underlying infrastructure which include the hosts or the machines, and the network infrastructure. In addition, several management systems need to be in place for an all pervasive enterprise level and secure grid systems. There are three main types of management systems which are important from the grid perspective namely the credential management systems, the trust management systems, and the monitoring systems. All the three issues mentioned above are dealt with in this book along with existing solutions and potential concerns. In the next chapter, we will look at the Grid Information security architecture mainly from the perspective of the Grid Security Infrastructure (GSI) and its open source and popular implementation, the Globus toolkit.

4 Grid Information Security Architecture

4.1 Introduction

There are many possible definitions of information security. One such definition can be found in the paper by McDaniel et al., which states that it is *"The concepts, techniques, technical measures, and administrative measures used to protect the information assets from deliberate or inadvertent unauthorized acquisition, damage, disclosure, manipulation, modification, loss, or use."*[57]. In other words, any information security system should define mechanisms to protect the information within the system. Different types of information that need to be secured depend on the type of system. For example, information in case of storage systems like databases, file systems, etc. are the data stored within those systems. On the other hand, information in network systems are the messages or packets flowing through the system. Information security in each system defines mechanisms to protect information typical of that system. Computer science researchers have developed algorithms, protocols, and mechanisms, which are used across different systems, some of which had been discussed in Chap. 2. In this chapter we will see how these concepts can be used in the context of grid computing systems.

The standardization effort of grid security has led to the design of security standards in grid which is defined under Grid Security Infrastructure (GSI). The driving force behind the generic standardization efforts in grid computing is the Global Grid Forum (GGF) [58][1]. GGF is a forum of researchers and practitioners for exchanging information and defining standards for grid computing. The open standard as has been put forward by the GGF community is called Open Grid Standards Architecture (OGSA). It is based on the seminal work by Ian Foster and group in 1998 [59,60]. OGSA defines mechanisms based on Web services for different

[1] Recently GGF and Enterprise Grid Alliance (EGA) have merged to create the Open Grid Forum (OGF).

systems to communicate and share the heterogeneous grid resources. Please refer to the appendix for details about OGSA, OGSI [61], and Web Services Resource Framework (WSRF) [62-64]. The chapter is organized as follows: first we will briefly discuss the security standards which are defined as Grid Services Infrastructure (GSI). We will go through the grid information security requirements and then discuss GSI in relative detail by talking about its implementation in the open source Globus toolkit.

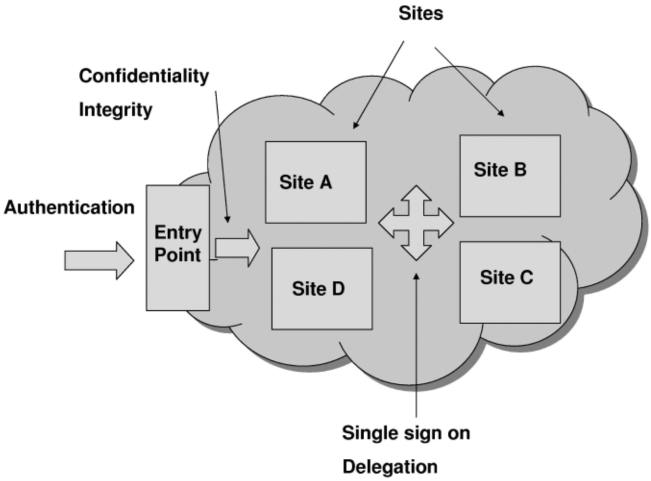


Fig. 4.1. Typical grid scenario

4.2 Grid Security Infrastructure (GSI)

Before discussing the grid security infrastructure, we need to understand the security requirements that drove the standards body to adopt such an infrastructure. As mentioned earlier, a grid defines a concept called the *Virtual Organization (VO)*. In a VO, different individuals, enterprises, organizations come together to share resources and services under a set of rules or policies guiding and governing the extent and conditions of sharing. VO can be formed across different universities, across different enterprises, as well as within an enterprise also. The level of heterogeneity defines the type of solutions. Therefore, the main aspect that separates grid systems from all the different systems are the heterogeneity involved and policy complications. We will talk about those in subsequent chapters. Here we will concentrate on the information security aspects and how they can be tackled.

Figure 4.1 shows a typical grid scenario consisting of sites which constitute a VO. A user submits a job to the grid which arrives at the entry point or the gatekeeper of the grid system. There should be mechanisms to authenticate the user at that point. When the job gets submitted to the grid then there is a need to provide confidentiality and integrity so that no one is able to see the contents of the information carried and is able to modify the contents. Finally, there should be mechanisms for single sign on and delegation. Discussions about the different information security requirements are provided below:

- **Authentication:** Grid security requirements should contain authentication mechanisms at the entry points. Different authentication mechanisms should be supported. It is possible to have different authentication mechanisms for different sites within a grid. Therefore, the security protocol should be flexible and scalable to handle all the different requirements and provide a seamless interface to the user. Furthermore, there is a need for management of context and sharing of context.
- **Confidentiality:** Grid security mechanisms should protect the confidentiality of the messages and the documents that flow over the grid infrastructure. The confidentiality requirements should include point-to-point transport as well as store and forward mechanisms. Similar to the authentication mechanisms, there may be a need to define, store, and share security contexts across different entities.
- **Integrity:** Grid security mechanisms should include message integrity which means that any change made to the messages or the documents can be identified by the receiver.
- **Single Sign on:** In a grid environment, there may be instances where requests may have to travel through multiple security domains. Therefore, there is need for single sign-on facility in the grid infrastructure.
- **Delegation:** There may be a need for services to perform actions on the user's behalf. A computational job may require accessing database many times. In that case there is a need to delegate the authority to some service which will perform the action on the user's

behalf. When dealing with delegation of authority from an entity to another, care should be taken so that the authority transferred through delegation is scoped only to the task(s) intended to be performed and within a limited lifetime to minimize the misuse of delegated authority.

4.2.1 Grid Security Model

Grid computing provides a virtualized view of the underlying grid resources. Such a virtualization also encompasses the security requirements. Therefore, there is a need for virtualization of security semantics to use standardized ways of segmenting security components like authentication, access control, confidentiality, etc. and to provide a standardized way to enable the federation of multiple security mechanisms. Therefore, this requires a loosely-coupled platform independent model of securing applications within and across organizations. Now the question arises about the paradigm involved in implementing such a loosely coupled, platform independent architecture.

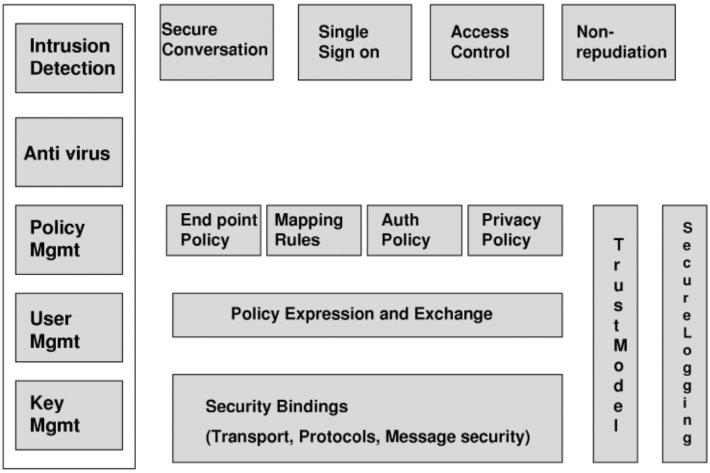


Fig. 4.2. Components of the grid security model

The last few years have seen the gradual adoption of Web services as an emerging architecture which has the ability to deliver integrated, interoperable services. Since Web services are gradually becoming a default and an industry standard, the OGSA grid computing model uses Web services as a model reference. Since confidentiality, integrity, policy management, trust management are also integral to Web services, the grid security infrastructure integrates the Web services standards like WS-Security, WS-Policy, WS-Trust, etc. in the specification. However, the Grid Security Infrastructure does not exclude transport layer security like Secure Socket Layer (SSL) on top of HTTP or HTTPs. Users are free to use HTTPs which provides confidentiality, integrity, and authentication. However, if there is a need to traverse multiple intermediaries, WS-Security can be used in conjunction with XML encryption, XML signatures and so on.

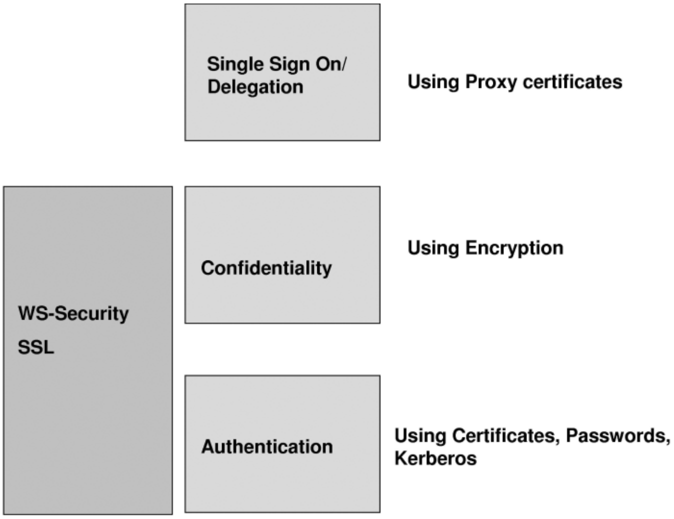


Fig. 4.3. High level view of GSI

Figure 4.2 shows the different components of the grid security model as described in [65]. As part of the chapter we would be looking at authentication, confidentiality, and single sign on/delegation aspects. As shown in Fig. 4.3, in GSI three types of authentication is generally discussed – using X.509 certificates, using passwords, and using Kerberos. For confidentiality mainly key based encryption algorithms are used. Sometimes, the need arises for having a session key and therefore, session management. For single sign on/delegation proxy certificates are generally used. Provisions for both transport layer security mechanisms like SSL and message layer mechanisms like WS-Security are provided.

4.3 Authentication in GSI

The most prevalent mechanisms of authentication in a GSI based grid is the certificate based authentication mechanism where a public key infrastructure (PKI) is assumed to exist which allows the trusted authority to sign information to be used for authentication purposes. In addition to certificate based mechanism, Kerberos and password based mechanisms have also been implemented.

4.3.1 Certificate based Authentication

Certificate based authentication mechanism has been implemented in all versions of Globus [66]. It assumes that each user within the grid system possesses a public private key pair, and there exists a trusted third party or Certificate Authority (CA) to sign and certify the users. The GSI certificate includes following information:

- A subject name, which identifies the person or object that the certificate represents.
- The public key belonging to the subject.
- The identity of a Certificate Authority (CA) that has signed the certificate to certify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

Logging into the Grid System

Figure 4.4 shows the four steps involved in allowing a user to access a grid system using the certificate based authentication in GSI. The different steps are:

1. The first step is to know the public key of the CA. This information is used to verify the validity of the certificate obtained from CA. The certificate is stored in the local host.

2. The second step is to create the public private key pair using any common protocol. The private key thus obtained is also stored in a secure place in the local host. A different credential service like MyProxy can also be used for this purpose. Please refer to Chap. 9 for details about MyProxy. In this step, the user also generates the certificate request, which is its public key signed with the user's private key. This is done so that the CA can verify the authenticity of the information.
3. In the third step, the CA first verifies the information obtained from the user and then signs the request with its public key. The certificate is then sent to the user.
4. The last step is to store the certificate which would be used for all subsequent authentication purposes. The following information is stored at the local host: (i) the public key of the CA, (ii) the user's public key, and (iii) the signed certificate.

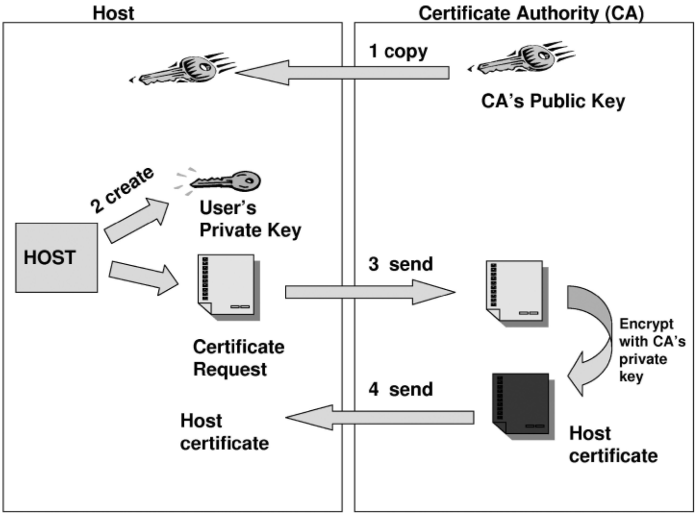


Fig. 4.4. Steps for logging into the grid

Mutual Authentication

Mutual authentication is an important aspect that needs to be considered where two hosts mutually authenticate each other if both of them trust the third party or the CA.

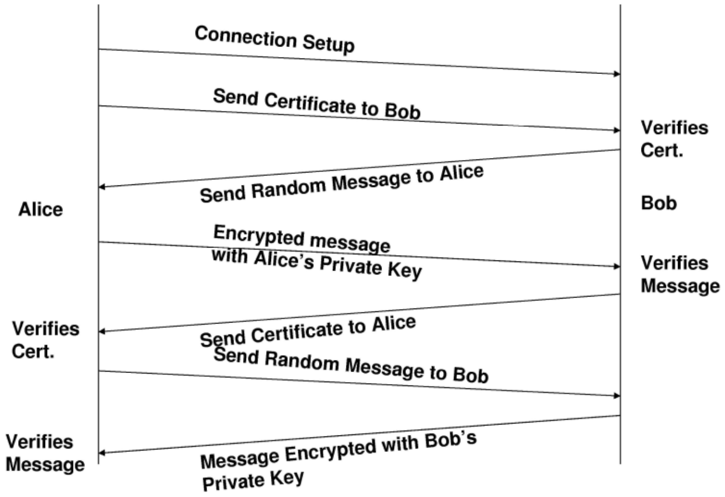

Connection Setup
Send Certificate to Bob
Verifies Cert.
Send Random Message to Alice
Alice
Bob
Encrypted message with Alice's Private Key
Verifies Message
Verifies Cert.
Send Certificate to Alice
Send Random Message to Bob
Verifies Message
Message Encrypted with Bob's Private Key

Fig. 4.5. Example of mutual authentication

Let us assume (see Fig. 4.5) Alice and Bob are authenticating each other.

1. First Alice sets up a connection with Bob.
2. Alice then sends her certificate over to Bob for authentication. The certificate is a standard certificate and holds the information about the identity of Alice, her public key, and the information about the CA.
3. After receiving the information from Alice, Bob first validates the received certificate to make sure that the certificate has actually been signed by CA and the authenticity of the public key. Bob then creates a random number or a message and sends it to Alice.
4. When Alice receives the random message, she encrypts it with her private key and sends the encrypted message back to Bob.
5. Bob then decrypts the message received from Alice and checks that the decrypted message is really the one that it sent before.

The main purpose of this step is to validate that Alice actually possesses the private key corresponding to the public key she has communicated to Bob.

6. At this point Bob trusts the identity of Alice. If mutual authentication is needed, Alice would also like to validate Bob's identity. In that case, steps 2 - 5 are repeated with Bob sending Alice his certificate and Alice sending the random message to Bob.

4.3.2 Password based Authentication

Though certificate based authentication systems are more secure, it introduces overheads in terms of public key infrastructure. In reality, password based systems are still used quite widely in enterprises. Therefore, to cater to a wide range of enterprises, the GSI design team felt the need for allowing passwords as means of authentication in Globus based grid systems. The Globus Toolkit 4.0 (GT4) [67,68] has the provision of allowing users to authenticate through username and password. The GT4 security allows SOAP messages to be secured using Transport Layer Security (TLS) or using WS-Security standards. The former is referred to as Transport Layer Security, while the latter as Message Layer Security. In the case of transport layer security, authentication is either carried out using X.509 credentials or in an unauthenticated mode ("anonymous" mode). In this mode, authentication can be done using username and password in the SOAP message. However, true multi-credentials are supported in case of message level security. Since it uses Web services standards like WS-Security and WS-SecureConversation (refer to the appendix for more details), it is neutral to the specific types of credentials used to implement this security. Web services standards allow GSI in GT4, to use usernames and passwords in addition to digital certificates. However, it is to be noted that more advanced security features like confidentiality, integrity, delegation are not present in password authentication based systems.

Based on our discussions in the previous section, it is quite clear that there are a lot of inadequacies in integrating GSI with a password based system. Firstly, due to the lack of confidentiality, there is a possibility of adversaries tapping into the system. Secondly, due to the constant change of policies and lack of trust on the host system, users do not store long term credentials in the host. To have a more secure system, researchers have come forward to integrate One Time Password (OTP) technologies with Globus.

One Time Passwords (OTP)

This is a step in the forward direction to remove some of the inadequacies of the password based systems. In this type of technology the passwords change over time, like the RSA® SecureID. The OTP technologies protects compromised user's password and allows the grid systems or data centers to securely transfer a short lived credential to the user. There is also a need for a secure exchange of credentials. The researchers from Lawrence Berkeley National Laboratory (LBNL) have developed a protocol that integrates the OTP technology with secure key exchange called OPKeyX [69]. The algorithm works as follows:

- A one time password is derived, which is a function of the key and a random number which can be the current time.
- A Diffie-Hellman key exchange algorithm is used to decide on a session key. The one time password derived in the previous step is used to encrypt the key exchange mechanism.

The OpKeyX protocol has been integrated with both transport level as well as Message Level security of GSI. In the case of the former, OpKeyX is used as the key exchange protocol in TLS. In the case of the latter, OpKeyX is used as the key exchange protocol in WS-SecureConversation.

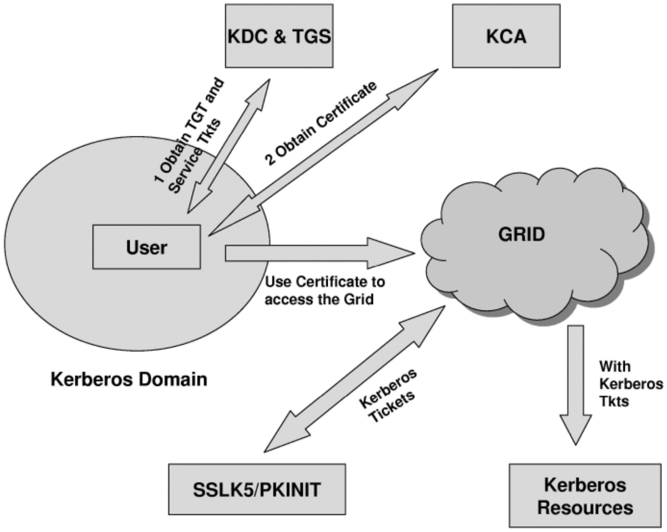


Fig. 4.6. GSI Kerberos integration

4.3.3 Integration with Kerberos

Kerberos is one of the most popular authentication systems used in enterprises. Please refer to Chap. 2 for details about Kerberos. GSI, in its current form, does not support Kerberos based interaction. In other words, Globus security does not accept Kerberos credentials as an authentication mechanism. To make this integration possible, there is a need for gateways or translators which accept GSI credentials and convert it to Kerberos credentials and vice versa. KX.509/KCA [70] can act as a GSI to a Kerberos gateway while SSLK5/PKINIT can be used as a Kerberos to GSI gateway.

Figure 4.6 shows, at a high level, how this integration can be achieved. The KCA is able to convert the Kerberos tickets into a valid X.509 certificate which can be accepted by the grid system based on GSI credentials. If there is a need to access a resource within a Kerberos domain, then there is a need to convert GSI credentials to Kerberos credentials, which can be done using the SSLK5 module. This mechanism can work in simple cases; however it cannot provide more complex security mechanisms like delegation across Kerberos and GSI domain and so on.

4.4 Delegation in GSI

Another very important requirement for a grid based security system is delegation where another entity gets the right to perform some action on user's behalf. This is especially important in case of grid because of the possibility of multiple resources involved in grid based transactions. It may be unnecessary or very expensive to authenticate each and every time a resource is accessed. On the other hand, if the user issues a certificate allowing the resource to act on its behalf then the process will become a lot simpler. This type of certificate issued by the user to be used by some other entity is called a *proxy certificate*. A proxy is made up of a new certificate containing two parts, a new public and a new private key. The proxy certificate has the owner's identity, with a slight change to show that it is a proxy certificate. The certificate owner, not a CA, will sign the proxy certificate. As part of the proxy certificate there is an entry with a timestamp, which indicates at what time the proxy certificate expires; by default it has a short term validity period of say a few hours.

Let us take an example to understand the delegation process. Let us assume a host A wants to delegate the responsibility of submitting a job in a host C to another host B. A owns a certificate signed by the CA. It then creates a proxy certificate and sends it to B. B then uses the proxy certificate to submit jobs on A's behalf. Figure 4.7 shows the different steps involved in the delegation process.

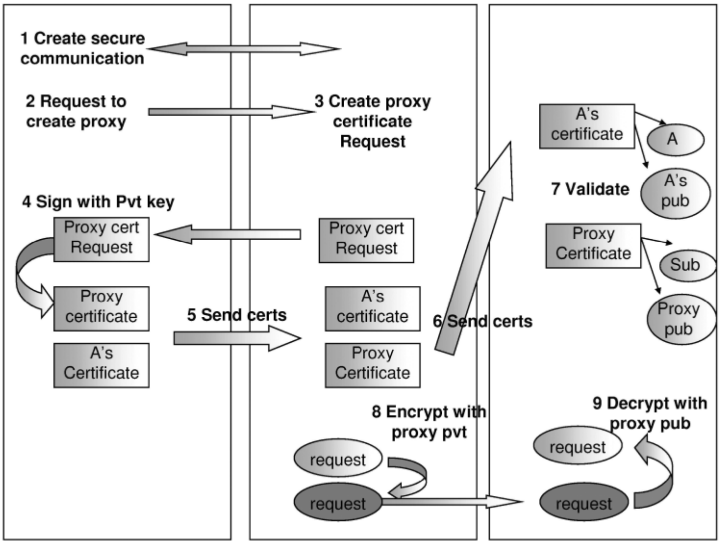


Fig. 4.7. Overview of the delegation activity in GSI

Different steps involved in the delegation process are as follows:

1. A secure communication is set up between the communicating parties, in this case, A and B. This can be done using SSL or some Web services security standards based protocol.
2. When a delegation is required, a delegation request is sent to B.
3. B creates a proxy certificate request which contains the information about a proxy public key and other identification information. It is to be noted that this step is similar to the certificate request sent to a CA. B stores the proxy private key securely.
4. Once receiving the request, A signs the certificate request with its private key to create the proxy certificate.
5. A then ships the certificates to B so that it can start the delegation process.
6. During delegation, B sends A's certificate, as well as the proxy certificate to C.

7. On receiving the certificates, C first obtains A's public key by decrypting the certificate with CA's public key. Once A's public key has been obtained and validated, it decrypts the proxy certificate with A's public key to obtain and validate the proxy public key.
8. During the actual transmission of information between B and C, B uses the proxy private key to encrypt the information.
9. C uses the proxy public key to decrypt the information.

4.5 An Example: Security in Globus Toolkit 4.0 (GT4)

Let us now take a look at the security implementation of the Globus Toolkit 4.0 [67,68].

4.5.1 Message Protection in GT4

GT4 uses two mechanisms to protect the SOAP messages being transferred between the different components, viz. transport-level security and message level security. Transport-level security protects the data transferred at the transport layer using standards like Transport Layer Security (TLS). Message level security, on the other hand, works at a higher layer and uses Web services based standards like WS-Security, WS-SecureConversation, etc. by protecting the SOAP messages that are being transferred over the transport channel.

Transport-Level Security

Transport level security, in GT4, is implemented using the Transport Layer Security (TLS) standards (refer to Chap. 2). GT4 uses the SSL/TLS protocol over HTTP for securing the communication between the client and the server. For providing secure communication, X.509 credentials are generally used for authentication. However, GT4 security does not limit itself to X.509 standards, it also supports unauthenticated communications, often referred to as "anonymous transport-level security." In this mode of operation, authentication may be done on a different level, e.g. via username and password in a SOAP message. GT4 implements the transport security using a secure socket implementation which is able to provide the security properties. The transport level security in GT4 is the default security mechanism used in GT4. The main reason for that is the performance overhead introduced by message level security mechanisms.

Message-Level Security

GT4 also uses Message-level Security (MLS) as an alternative to transport-level security, where encryption, authentication, and integrity mechanisms are employed at the message layer, rather than at the transport layer using Web services standards like WS-Security and WS-SecureConversation (see the appendix). WS-Security standards provide mechanisms to provide confidentiality, authentication, and integrity to the SOAP messages. GT4 security uses these mechanisms to provide security on a per-message basis. However, it is to be noted that this does not establish security context similar to SSL/TLS. To create the security context, WS-SecureConversation is used. WS-SecureConversation is a proposed standard that allows for an initial exchange of message to establish a security context which can then be used to protect subsequent messages in a manner that requires less computational overhead.

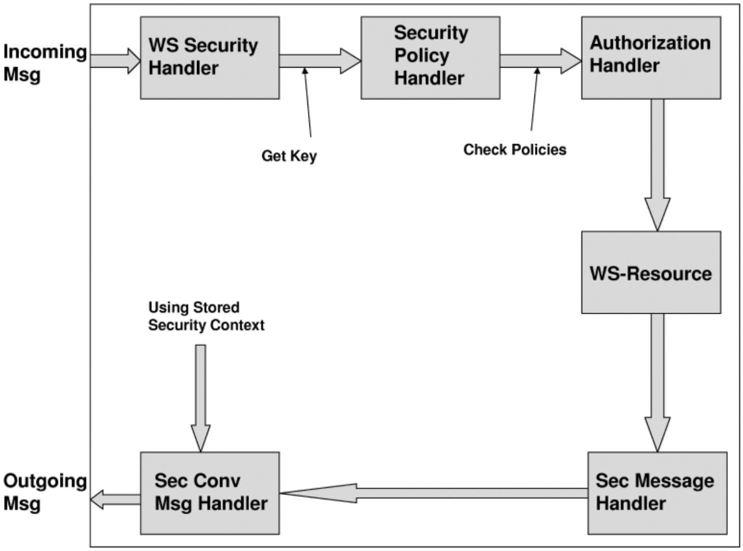


Fig. 4.8. Overview of the GT4 message handling

GT4 MLS provides two mechanisms, GSI Secure Conversation and GSI Secure Message security, for authentication and secure communication.

- **GSI Secure Conversation:** In the GSI secure conversation approach, the client establishes a context with the server before sending any data. This context is very similar to the context established during a SSL/TLS session. The context establishment phase helps the client and the server to create and store a shared secret used for future conversations. Once the context establishment is complete, the client can securely invoke an operation on the service by signing or encrypting outgoing messages using the shared secret created in the context. This mechanism is faster as once the context is established, a symmetric key is used for encryption and signing purposes.
- **GSI Secure Message:** The GSI secure message approach differs such that no context is established before invoking an operation. The client simply uses existing keying material, such as an X509 certificates, to secure messages and authenticate itself to the service. This is a slower mechanism that the previous one.

Figure 4.8 shows the architecture of GT4 security in case of any message arrival. When a message arrives from the client several security handlers are invoked.

- **WS Security Handler:** This handler extracts any keying material that is present in the message. Validation through checking the signatures are also carried out in this step.
- **Security Policy Handler:** This handler checks for any service specific policies that may be present. The policies can be specified during service deployment.
- **Authorization Handler:** This handler specifies the amount of authorization present in invoking a service.
- **Secure Conversation Message Handler**: This handler is one of the two outbound message handlers. This deals with encrypting and signing messages using a previously established security context.
- **Secure Message Handler**: This is another outgoing message handler. This deals with messages by signing or encrypting the messages using X509 certificates.

Comparison between the Approaches

Comparing between the two mechanisms, message-level and transport-level security, two main points need to be considered: end-to-end security and performance.

- **End-to-End Security:** The transport-level security works as a point-to-point security mechanism and does not work across multi-hop connection. This is one of the benefits of message-level security. It works across hops and is a comprehensive end-to-end solution.
- **Performance:** However, when we are looking at the performance overhead associated with Web services based security mechanisms, it is quite significant. Based on study by Shirasuna et al. [71], the GSI with Web services security performs a few times slower than GSI with SSL. Another interesting aspect of the study is that most of the time taken for Web services security comes from XML manipulations. The authors have suggested stream based pipelining at each step so improve the performance. Till a more efficient mechanism of XML manipulations happen, the Web services based security standards will remain significantly slower than the SSL based systems.

4.5.2 Delegation in GT4

GT4 supports delegation through the use of X.509 based proxy certificates. Proxy certificates allow bearers of X.509 certificates to delegate their privileges temporarily to another entity. GT4 supports the delegation process through two components: A *Delegation Factory Service (DFS)* and a *Delegation Service (DS)*. The DFS is responsible for creating the WS-Resource while DS is responsible for managing the delegated credentials. The delegation process is as follows:

- The DFS publishes its certificate chain, including the service's certificate, as a resource property.
- The DS client extracts the public key from the DFS certificate, after obtaining and validating the certificate chain.
- The client then creates the proxy certificate it is going to delegate by binding, i.e. signing, the service's public key to the proxy certificate information using its private key.
- Finally, the client passes the certificate chain that starts with the proxy certificate to the delegation factory service, which upon receipt replies with the address to the WS-Resource of the delegated credential.

Delegation process is secured using the transport level security described before. Mutual authentication, authorization, and integrity protection is provided. The interface to the delegation process is based on WS-Trust specification.

4.6 Chapter Summary

One of the main concerns for secure grid systems is to have a robust architecture to secure the information flowing through the system. Grid Security Infrastructure (GSI) is an effort to standardize the security requirements and its manifestations in the context of Virtual Organization (VO) based grid systems. Different security requirements handled by GSI are: authentication, confidentiality, integrity, single-sign-on, and delegation. GSI traditionally supports certificate based (X.509) authentication mechanisms. Recent versions of GSI (implemented in Globus Toolkit 4.0) supports password based authentication, and research efforts are underway to integrate One Time Password (OTP) and Kerberos authentication with GSI. Confidentiality is supported through transport level security using SSL/TLS protocols, and message level security using Web services standards. GT4 is one of the few implementations where message level security is used for grid confidentiality purposes. In a grid system delegation assumes enormous importance since jobs can run on multiple sites and requirement of multiple authentications is a huge overhead. In such cases, proxy certificates are used to delegate authority to some other entity or system. The chapter provides a detailed discussion on the different security requirements and their implementations. In the next chapter, another important grid security architecture issue or grid authorization will be discussed.

5 Grid Authorization Systems

5.1 Introduction

Until now, we have looked at issues pertaining to the user – issues that make a user comfortable with the grid system, namely authentication, confidentiality, integrity, and single sign on/delegation. Now, we will look at issues pertaining to the resource which constitute the heart of the grid system. In this chapter, we will particularly look at the issues related to the authorization and access control of resources within a grid system.

Many readers have access to some kind of library like a public library, office library, etc. I also have access to my company's library because I am an employee of the company. For that reason, the authorities have issued me a library card so that I can access the resources of the library namely the books and the CDs at my convenience. To issue the library card, someone must have checked my credentials and found that I was worthy of accessing the resources of the library. Therefore, there exists a *system* which *authorizes* the users to access the library resources by issuing the library card. Similar to the above example, people need such authorizations daily to access a variety of resources in their day to day lives. Take an example of database resources: there are database administrators who have access to the tables, where they can modify, delete, or add tables. There are also users who can only access the data of the table.

Sometimes, the term *authorization* is mistaken with *access control.* Therefore, before going further into the discussions, let us try to clarify the differences. Subsequently, whenever the terms are referred to in this book, they will conform to the explanations provided below. Authorization can be loosely defined as the act of providing and checking the authority of the user or a job on a specific set of resources. Therefore, in the library example, the act of issuing me the library card, and the act of allowing me to access to the library resources fall under the purview of the authorization system. On the other hand, access control, by definition is broader. It is a more general way of controlling the access to the set of resources, including the time of the day, IP addresses, and other parameters. Therefore, in the library example, the library closes at 9pm everyday can be an access control mechanism. Though I have the library card, I would not be able to access the library resources when the library is closed. Therefore, authorization mechanisms can be thought of as a subset of access control mechanisms. Generally a good authorization system limits the access of system resources to outsiders, and a fine grained access control mechanism, in addition, provides resource control for the authenticated/authorized users.

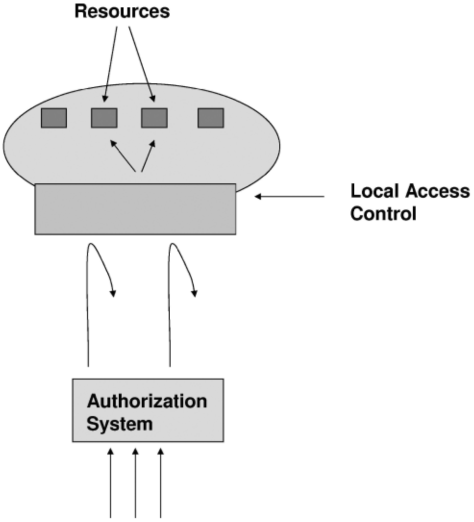


Fig. 5.1. Interaction of authorization and access control in a grid system

Figure 5.1 shows the possible interaction of authorization and access control systems in a grid computing environment. Let us assume that only two out of the three users are authorized to access the grid resources. However, the local access control mechanism, based on some criterion, prevents all the users from accessing the resources. This example shows how the authorization and access control mechanisms can be combined to provide fine grained resource access control.

5.1.1 Different Access Control Models

Let me try to describe the different access control models based on my dilemma to read a book called *The God of Small Things,* by Arundhati Roy. I knew that my friend Bob owned it; however I was not sure whether he will allow me to borrow the book, as the access to the book was at his *discretion.* Similarly, I also knew that my aunt had borrowed the book from the local library. However, neither my aunt nor I knew the library policies regarding lending a borrowed book to someone else. Moreover, I knew that the local library had another copy. However, they had a graded book access policy, based on the donation one gives to the library. I was unsure about my *role* there too. This simple dilemma does bring out the different levels of access that the access provider has on the resource it controls. There are three main types of access control, namely Mandatory Access Control (MAC) [72,73], Discretionary Access Control (DAC) [74], and Role Based Access Control (RBAC) [75].

Mandatory Access Control (MAC)

One of the access control models is Mandatory Access Control (MAC) which is also known as the Lattice Based Access Control (LBAC). In such an access control mechanism, the access to certain objects or resources is expressed in terms of security labels attached to subjects and objects. A label on an object is called a *security classification*, and a label on a user is called a *security clearance.* A system following the MAC access control mechanism is similar to the library example where my aunt borrowed a book from the library. The book is the object or the resource, and my aunt and I are the subjects. The library may enforce a policy that library card holders (like my aunt) have higher *clearance* than nonholders (like me). Moreover, the books may have different *classifications.* For example, a manuscript may have higher classification than a fiction book. Based on the subject clearance and object classification, the library can enforce stringent policies. Similar policies are generally enforced in a MAC system. Things become a little more complicated than above as there can be write policies, read policies, and so on. Generally a lattice of security labels is formed which determine the unidirectional information flow. Therefore, these types of access control mechanisms are also called Lattice Based Access Control (LBAC) [76, 77]. Depending on the nature of the lattice, the one-directional information flow enforced by MAC can be applied for confidentiality, integrity, or a combination of them. There is also variation of MAC schemes where the unidirectional information flow is partly relaxed to achieve selective downgrading of information or for integrity applications [72]. MAC schemes are generally used in high security environments where security clearances and classification becomes very important for accessing objects or resources.

Discretionary Access Control (DAC)

In Discretionary Access Control (DAC) mechanisms [73], the owner or the creator of the object has the discretionary authority over who else can access the object. In the book dilemma that I had, Bob held the discretionary right to allow or disallow me from borrowing his book. In real life, such access controls exist everywhere, from files in operating systems to inviting people to marriage parties. Since the earliest formulation of DAC several variations of the DAC policies had been developed, which are particularly concerned about how the owner's discretionary power can be delegated to other users, and how access can be revoked. Based on the research, several types of DAC mechanisms are possible:

- **Strict DAC,** where the owner is the only one who has discretionary authority to grant access to an object or resource and the ownership cannot be transferred. For example, in a strict DAC scenario, Bob is the only person to grant me the access to his book and in no way can he delegate the responsibility to anyone.
- In a **Liberal DAC scenario,** the owner can delegate responsibility for granting access to an object to other users. There can be different levels of delegation. For example, there can only be one level of delegation where Bob delegates the responsibility for granting access to his book to Alice. However, Alice does not have the authority to further delegate. However, there can be multi-level delegation. As mentioned in Chap. 4, delegation assumes significant importance in a grid scenario.
- DAC with a **change of ownership** allows a user to transfer ownership of an object to another user. This is similar to Bob selling his book to me. After that it is my responsibility to grant rights and delegate authority to other users.

Role Based Access Control (RBAC)

Role Based Access Control (RBAC) [75] after its definition and initiation received enormous attention from the security community. In RBAC, permissions are associated with roles (see Fig. 5.2), and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies the management of permissions. Roles can be created for the various job functions in an organization and users are then assigned roles based on their responsibilities and qualifications. Users can be easily reassigned from one role to another. Roles can be granted based on new permissions as new applications and systems are incorporated, and permissions can be revoked from roles when needed. In the book dilemma that I had, the library may have different roles based on the level of donation own pays, and there may be different permissions assigned to each role.

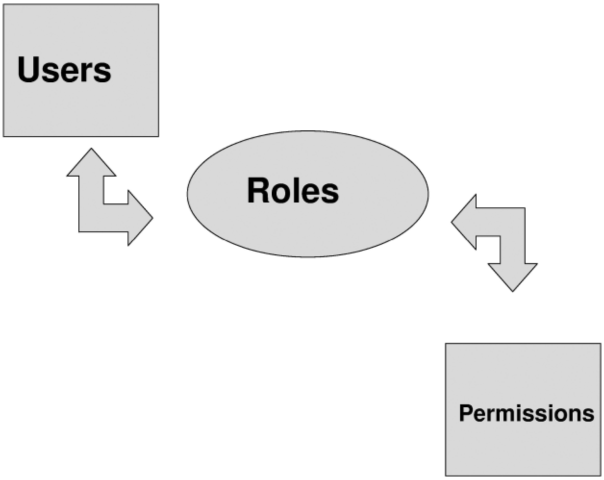


Fig. 5.2. Role based access control

An important characteristic of RBAC is that by itself it is policy neutral. RBAC is a mechanism of articulating policies rather than embodying a particular security policy (such as unidirectional information flow in a lattice). RBAC is a scalable and flexible mechanism for articulating access control policies. It is scalable, as the number of associations compared to a typical user to permissions mapping is less, and it is flexible as Sandhu et al. [78] has shown that DAC and MAC are different manifestations of the RBAC system. They have shown that by tuning the different components RBAC can be converted to different forms of DAC and MAC access mechanisms.

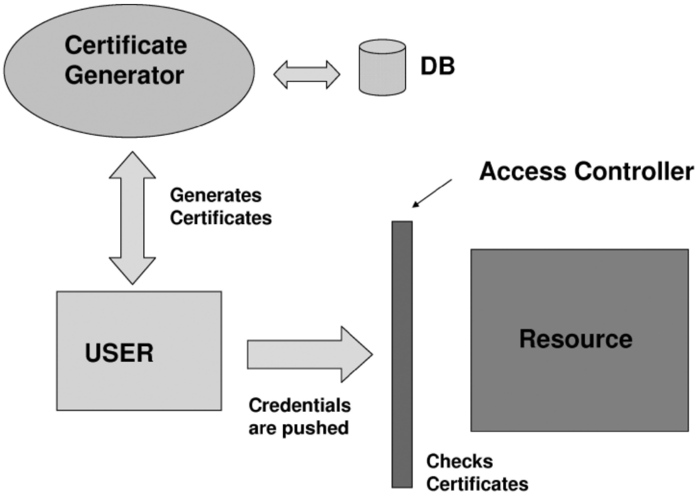


Fig. 5.3. General certificate based push model

5.1.2 Push vs. Pull Authorizations

In the last subsection we discussed the different types of access control models. There are also different ways of authorizing a user to a resource. Let us look at the different type of authorizations based on my attempt to access the books of my company library. Based on my credentials, which are my employee identification, my years of experience, my qualifications, and my expertise I have been issued a library card. The library card specifies my name, my access to the certain library room (there are many), and the validity period of the library card. The information is then signed by me and then sealed by the manager authorizing my entry to a particular library room. Once I have the card, I can access the library room without anyone authorizing my identity again. Once I need to access the resources of that room of the library, I swipe my card and access the resources. These types of authorization systems where authorization credentials are "pushed" are called push based authorization systems. Push based authorization systems are pretty common in practice. All certification based systems employ this mechanism. A general certificate based push model is shown in Fig. 5.3. In such a mechanism, there exists a certificate generator who checks the user's credentials and generates a certificate so that the user can access the resource. The access controller allows access to the resource based on the certificate validity.

Instead of generating the library card, if the library authorities had allowed me to access the library resources through my employee card, then it would have been a classic case of "pull" model. In the pull model, the users provide the minimum credentials to the access controller and it is the responsibility of the controller to check the validity of the user based on the policies of the system. The pull model is also quite widely used in different authorization systems. The file systems in different operating systems employ pull mechanisms, where the user provides their minimum credentials, and the access policies corresponding to the user are "pulled" by the operating system. Based on the policies, the operating system either allows or disallows the user to access the resources. Figure 5.4 illustrates the pull model, where the user supplies the minimum credentials like the username, password, and the controller makes the access decision based on the user policies pulled from the database.

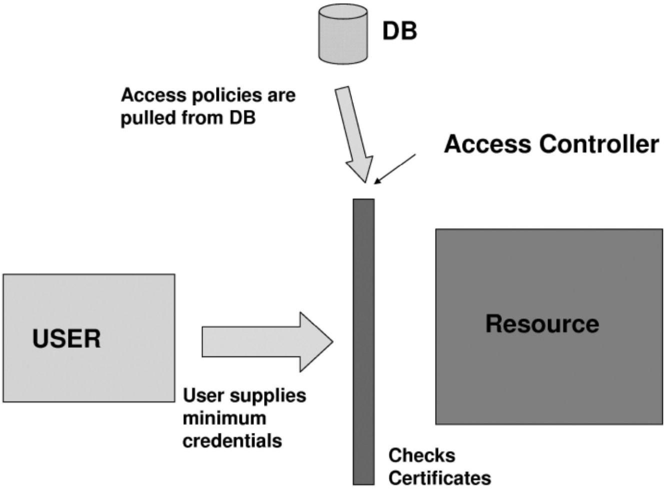

DB
Access policies are pulled from DB
Access Controller
USER
User supplies minimum credentials
Resource
Checks Certificates

Fig. 5.4. A simple pull based authorization system

As has been mentioned, both these mechanisms are very popular. However, which authorization system should be used at what time? To answer this question, a comparison based on the following parameters is provided: scalability, flexibility, usability, and revocation.

- **Scalability:** Generally, push based mechanisms are considered to be more scalable than the pull based mechanism. This is because in the former there is a decoupling between the certificate generator and the access controller. Therefore, the two operations can take place at two different times. On the other hand, in a pull based mechanism the access controller itself checks the database and grants access to the user based on the user policies. This may limit the scalability of the system in many cases.
- **Usability:** Another parameter that is generally considered in case of authorization systems is how user friendly the system is. Here the pull based mechanisms are better as the users do not have to obtain certificates from the certificate generator and it is the access controller's responsibility to grant access to the user.
- **Multiple Stakeholders:** In many cases, there may be multiple stakeholders involved for the resources. It may then be scalable to ask the users to get the certificates from the stakeholders before granting access to the resources. However, this is done at the price of loss of usability of the system. As we will see in our subsequent discussions, though both Akenti and VOMS involve multiple stakeholders, Akenti uses a pull based mechanism while VOMS uses a push based mechanism.
- **Revocation:** Another concern for the push based mechanism is the revocation policies. Generally, this is obtained through expiration time provided in the certificate, after which the certificate is considered to be invalid. However, there is a time for which a compromised user may still be able to access the resources. This is generally a very big concern for extremely secure systems.

5.2 Characteristics of Grid Authorization Systems

After having discussed the different generic access control and authorization mechanisms, let us now try to understand the authorization requirements of a grid computing system. To understand the scope of the problem, let us look at the library example a little differently. The library not only contains books of different languages and areas, but also CDs, DVDs, Playstations, and others. The library cards are issued to library users, based on the subscription they pay, which may vary depending on their usage and interests. Therefore, a user interested in reading French romantic literature may be charged differently from a user who is interested only in Playstations. Moreover, the different parts of the library may be owned by different people, even multiple people, i.e., there may be multiple stakeholders for the resources of the library. If we were to design an authorization system for such a library, the first thing we need to consider is that the system does not have too much overhead. In other words, there is a need to authorize users; however there should not be a long queue in front of the library. Therefore, **scalability** is one of the primary concerns for designing such a system. Secondly, one has to keep in mind the effect in case the system is tampered. In that case, a user may be given more or less authorization than what the user deserves. Therefore, **security** is surely a very important concern. Moreover, it is possible that after a user has been authorized and allowed to enter, the authorities get the information that the user is a thief. Therefore, there should be a mechanism to deny him/her access to the resources, once such information is available. In other words, there should be means for **revocation** of the user authorization. Lastly, if different stakeholders in the library employ different authorization systems, is the current system interoperable with them? Therefore, there is a need for **inter-operability** of the authorization mechanisms.

As most readers would have guessed by now, there is an uncanny similarity between the library example provided above and a grid system. In a grid system, there is a set of heterogeneous resources, having one or more stakeholders. Similar to the above example, there also may be different policies, systems across the multiple resources and any authorization system should be able to operate seamlessly. Based on the above argument let us look at the different characteristics we feel are needed in a grid authorization system, *viz.* scalability, security, revocation, and interoperability.

5.2.1 Scalability Issues

Scalability is one of the most important and desirable characteristics of a grid authorization system. A system is supposed to be scalable if there is no perceived difference when the system is scaled up in terms of entities accessing the system. There are two aspects to grid scalability, one is based on the number of users, and the other is based on the amount of grid dynamism. The first one is straightforward – the grid authorization system should perform well when the number of users increases. In addition, grid systems have an inherent dynamism embedded into them. In a grid system, users may join or leave the grid system quite frequently. Furthermore, resources may be added to or removed from the grid infrastructure in an on demand basis. These aspects of dynamisms have a significant effect on the design of the grid authorization system. Based on the two different types of scalability parameters, there are two different types of scalability: performance scalability and administrative scalability.

- **Performance Scalability:** Number of users is the primary measure for this type of scalability. The authorization system should not be a bottleneck in the grid infrastructure and should scale even if the number of users increase significantly. This aspect has a profound influence on choosing either the push based or pull based model for authorization.
- **Administrative Scalability:** If the system is highly dynamic, then administrative scalability assumes enormous importance. A rule based authorization which maps the user to the resources may be simple to implement, however it does not provide administrative scalability, as there is a need to change the tables every time a user is added or deleted from the system. Centralized authorization mechanisms are generally used to tackle administrative scalability issues.

5.2.2 Security Issues

Like any other system, one has to analyze the security vulnerabilities existing in grid authorization systems. If an adversary hacks into the grid authorization system, one has to understand the effect of such a malicious activity. Two types of compromises are possible in a grid authorization system: user level and system level. In the former case, a user is compromised allowing the adversary to use the grid as the user would. The second type of compromise is where the authorization system is taken over by the adversary.

- **User Level:** In this type of compromise, an adversary poses as a user to the grid system. Once having gained access into the grid system, the malicious user can create havoc by even generating denial-of-service attacks. More details about this are provided in Chap. 6. The adversary could have gained access into the system by tampering into the user credentials. Therefore, the authorization systems should provide secure credentials using the available standards and the communication to the user should be done through secure mechanisms like the SSL. To prevent replication attacks, where the adversary sends an old credential, time stamp should be added to the communications between the authorization system and the users.

- **System Level:** In case of centralized authorization systems, it is sometimes easy to compromise the authorization system rather than individual users. The adversary can employ two types of techniques to compromise an authorization server. Firstly, he/she can try to gain access to the server faking an administrative account. After getting access, the adversary can do all the things with the authorization system that an administrator is authorized to do. Secondly, the adversary can employ Denial-of-Service (DoS) attacks on the authorization system. Different DoS techniques are discussed in Chap. 6. If authorization is mandatory to access the grid system, a DoS attack on a centralized grid authorization system will lead to a DoS attack on the grid system as a whole.

5.2.3 Revocation Issues

Another important issue that needs to be considered before designing a grid authorization system is the issue of revocation of authorization. Consider the following scenario: A user logs into the grid system and is authorized to access the resources of the system. After some time, it is learnt that the user has been compromised. In this case, the user should be denied access to the resources. Two mechanisms are generally designed to tackle this problem.

- In the first type or *active mechanism*, there is a communication between the user and the receiver access control mechanism based on which the user is denied further access to the resource. This type of mechanism can operate very quickly and the revocation can happen as soon as the compromise is detected. Generally this is done through the use of Certificate Revocation Lists (CRL) issued by the authority, and the verifying authority or the access controller needs to check whether a CRL exists for the credentials send by the user. There are two types of overheads associated with such systems. There is an overhead of generating and sending the CRLs to the access controller. However, the more significant overhead is because each time the access controller needs to see whether there is a CRL associated with each user credential. This may lead to a loss of scalability, especially is there are a huge number of users in the grid system.

- The other type of revocation mechanism is more *passive* and is done through expiration times provided in most certificates. During the generation of certificates, an expiration time is provided after which the certificate is deemed invalid. In terms of scalability, these types of passive revocation mechanisms are better than their active counterparts. However, the scalability comes at a cost. Let us assume that the certificate is generated at time *T* and the expiration time is (*T*+*t*). Now the user is compromised just after time *T*. Then for a period of *t*, the adversary is capable of compromising the system further. If the time *t* is small, then the system is more secure. However, smaller *t* also indicates that there are more number of authorizations required, reducing the scalability of the system. Therefore, there is a trade-off between the scalability and security, which is tuned by the choice of the time *t*. The above problem is illustrated in Fig. 5.5.

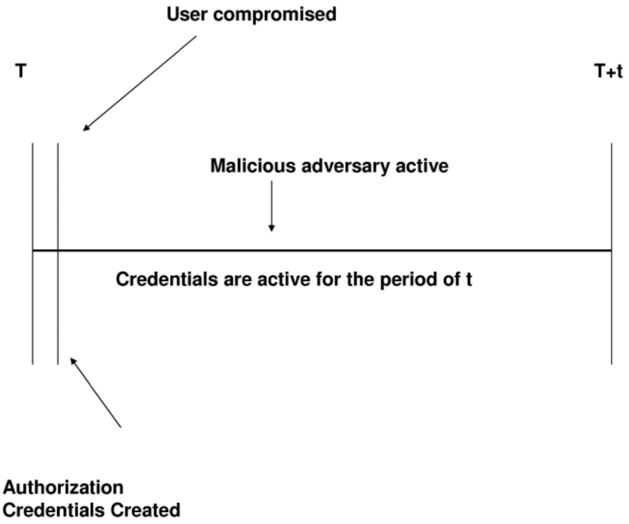

User compromised
T
T+t
Malicious adversary active
Credentials are active for the period of t
Authorization
Credentials Created

Fig. 5.5. Illustration of the expiration time problem

5.2.4 Inter-operability Issues

A grid system is characterized by heterogeneity not only at the resource level but also at the policy level. Therefore, the authorization systems should be inter-operable across multiple systems having heterogeneous policies. There may be issues of multiple authorization systems, as well as multiple communications protocols, and algorithms. One of the ways that most systems try to tackle inter-operability issues is through standardization. Standards are evolving for the grid systems, so as we discuss the different authorization systems we will talk about standards that have been employed. We will also make recommendations about the possible standards that may play a part in making the inter-operability work in grid authorization systems.

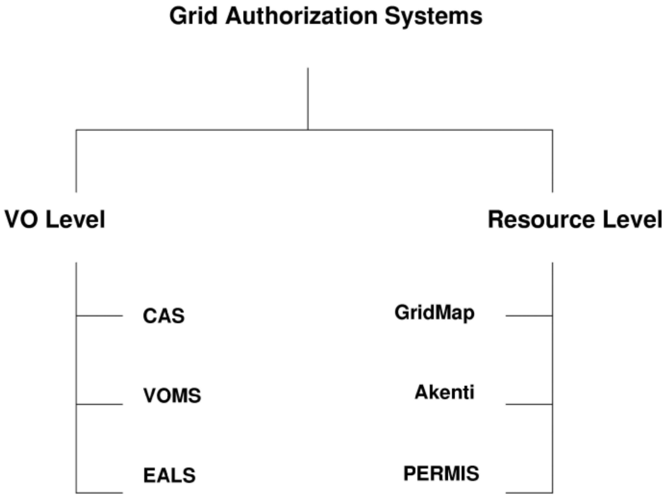


Fig. 5.6. Different types of Grid authorization systems

5.2.5 Grid Authorization Systems

We have discussed the characteristics of the grid authorization systems. In this section, we will discuss some authorization systems that have been used in popular grid implementations and other distributed systems. As illustrated in Fig. 5.6, the grid authorization systems can be mainly divided into two categories: *VO level systems* and *resource level systems*. Virtual organization or VO level systems have a centralized authorization system which provides credentials for the users to access the resources. Resource level authorization systems, on the other hand, allow the users to access the resources based on the credentials presented by the users.

Examples of VO level grid authorization systems are Community Authorization Service (CAS) [79], Virtual Organization Membership Service (VOMS) [80], and Enterprise Authorization and Licensing System (EALS) [81]. Examples of resource level grid authorization systems are Gridmap, Akenti [82], and Privilege and Role Management Infrastructure Standards (PERMIS) [83]. In the next few sections we will discuss in detail these different systems.

5.3 VO Level Authorization Systems

VO level grid authorization systems are centralized authorization for an entire Virtual Organization (VO). These types of systems are necessitated by the presence of a VO which has a set of users, and several Resource Providers (RP) who own the resources to be used by the users of the VO. Whenever a user wants to access certain resources owned by a RP, he/she obtains a credential from the authorization system which allows certain rights to the users. The user presents the credentials to the resource to gain access to the resource. In this type of system, the resources hold the final right in allowing or denying the access to the users.

5.3.1 Community Authorization Service (CAS)

Community Authorization Service (CAS) has been developed as part of the Globus toolkit. CAS looks at the problem of scalable representation and enforcement of access policies within distributed virtual communities. Such communities may comprise of many communities, each participating as a resource provider and/or resource consumer. The problem of authorization is handled using a trusted third party called the Community Authorization Service (CAS) server which is responsible for managing the policies and governing access to the community's resources.

The Community authorization service can be viewed as a service which has been given the authority by the community to authorize users on its behalf. The CAS makes the implicit assumption that the users of the community and the resources agree to have the CAS as the authorization service. The CAS server, acting on the community's behalf, provides capabilities to users based on the policies maintained at the CAS policy database. The users use the capabilities provided to them by CAS and show them to the resource server to grant them access to the resources based on their capabilities. The resources may have their own local access control policy. Let us assume that the authority granted by the resource provider to the community is ***C***. Let the capability provided to the user by the CAS server be ***U***, and the resource allows ***R*** restrictions. Then the effective capability of the user on the resource is $C \cap U \cap R$. The process is described in Fig. 5.7.

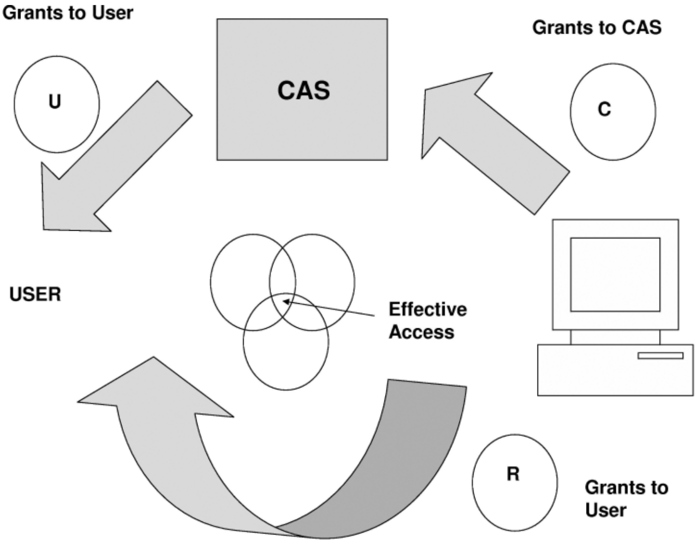


Fig. 5.7. Effective capability in case of CAS

The different steps involved to authorize a user in CAS are as follows:

1. **Resource – Community Interaction:** In this step the resource provider provides certain capabilities to the community to its resources.

For example, the resource provider allows “read” actions to 30% of its resources to the community. These policies are then put into the CAS policy database.

2. **Community — User Interaction:** In this step, the user is granted certain capabilities by the community. However, it is to be noted here that the community cannot grant more capabilities to the user than what is provided to it by the resource providers. In the previous example, the community can only grant “read” access to the set of the resources of the resource provider, nothing else.

The first step is implemented by allowing an individual within the community to acquire an administrative role. Once the role is acquired, the administrator can then enroll users, resources, and identify policies which connect users and the resources based on the capabilities provided to the community by the resource providers.

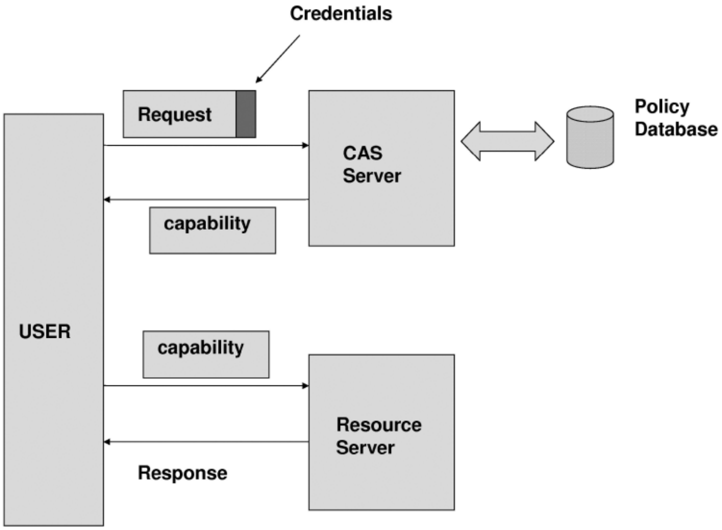


Fig. 5.8. Overview of CAS

The community - user interaction is achieved by the interaction between the CAS server and the policies stored in the policy database based on the policies granted to the community by the resource providers. The different components involved in achieving this step are:

- **CAS Database:** The policy database contains entries about the different users contained in different user groups, resources contained within resource groups, and actions contained within action groups, and a set of user access policies. The policy statements define which *users* can access which *resource* or *resource groups*, and what are the permissions granted to the user. The permissions are denoted by a set of actions or action group and a service type. The action denotes a type of action like "read," "write," "execute," etc., which the user can perform on a resource or a resource group. Service type defines a namespace in which the action is defined.
- **CAS Server:** CAS server's responsibility is to get the user request, check the validity of the user credentials, and generate the user capabilities based on the policies stored in the policy database. The policies are granted based on restriction policies defined through the extensions of X.509 certificates.

Figure 5.8 illustrates a typical use of CAS for authorization. A user may want to access certain resource in the grid owned by the community. The user sends a request to the CAS server for a capability to access the resource, and presents a set of credentials to the CAS server to identify himself/herself. The CAS server delegates an appropriate capability to the user based on the policies stored in the policy database. The user then requests the resource with the capabilities given to it by the CAS. The resource server then allows or disallows the user to access the resource based on the local policies of the resource for the user. For example, a user may have the capability to access a particular resource, however the resource may decide to temporarily ban the user from accessing based on certain condition. Then the user will not be able to access the resource though he/she possesses the capabilities from the CAS. Therefore, in a CAS authorization system the resource is the ultimate authority to decide which user can access its resources, and CAS is a facilitator in the process.

Based on the discussions made above, let us try to discuss applicability, and several other characteristics of CAS.

Applicability of CAS

From the inherent design of CAS, it is clear that CAS makes an assumption that there is a user community, a set of resource providers, and a set of users, and there are interactions among them based on Fig. 5.8. In other words, it is a virtual organization type of setup, where the resources affiliate their systems to the greater community of users who use the resources for compute intensive jobs. Such communities like the e-sciences community and others will greatly benefit from the CAS authorization system. There are case studies like the Earth System Grid (ESG) [84], which is a distributed network of storage systems containing environmental data. However, to make CAS viable for the enterprise, several considerations need to be made:

- **Policies:** Enterprises not only have compute intensive, batch jobs but a plethora of other applications and complicated policy mechanisms. Policies may not be dependent on simple actions and resources, but a host of other parameters like time, events (possibly through message queues), authentication mechanisms, licenses, etc. Therefore, more sophisticated policies need to be developed to cater to the enterprise community.
- **Integration:** Many enterprises already have different authorization and access control mechanisms in place. Enterprises use Citrix®, IBM®, or Microsoft® products for access control purposes. Following popular standards like XACML (see the appendix for details) and integrating with the different enterprise products will help them bridge the gap.
- **Push Based model:** This requires the applications to interact with the authorization system, in this case CAS, and get the capabilities to access the resources. Application wrappers need to be written to make this process transparent to the users.

Scalability

As mentioned in Sect. 5.2.1, there are two types of scalability: performance scalability and administrative scalability. Since CAS has been designed for the scientific community, where the number of users is both large and dynamic, the scalability issues have primary importance in the design of the CAS system. In both counts of scalability, CAS scores pretty high. In case of *performance scalability*, since CAS uses a push model, it is inherently scalable, because the resources do not need to check the users individually which had to be done in case of a pull-based models. In case of administrative scalability also, CAS performs well because there is only a single place that the administrators need to update in case of user joining the community, or leaving the community. However, there is a problem of single point of failure, as the whole grid system will become invalid if the CAS fails. Some type of replication can be carried out to make the system robust. Another solution that can be employed is a hierarchical CAS system, so that even if one system fails the effect is limited to a particular level in the hierarchy.

Security

Let us now consider the cases when a malicious adversary tries to attack the CAS system. The adversary can do this in two ways: It can masquerade as a valid user/administrator and get into the system or it can launch a denial-of-service attack on the CAS system.

- **Masquerade Attack:** In this type of attack the adversary masquerades as a valid user/administrator and compromises the system. A GSI credential based authentication mechanism is used to prevent such an attack. Therefore, it is difficult for an adversary to launch a masquerade attack. If an adversary can break into a CAS system as an administrator, then it can provide more access to certain users, and less access to certain other users based on adversary's discretion. The first problem is not significant if some resource level checks exist, as the resource will not allow the user to get more resources than the CAS is allowed to authorize. However, the second problem is more significant, as the CAS can deny access to the user causing an effect similar to denial-of-service.
- **Denial-of-Service (DoS):** A DoS attack on the CAS is possible by sending a lot of authorization requests to the CAS. Each request may flow over a SSL channel which authenticates the user. However, that does not stop a malicious user from sending thousands of connection requests per second which can easily bring the CAS server down. Once the CAS server is down, the grid system is paralyzed. Therefore, techniques such as ingress filtering and others need to be employed to prevent such an attack from becoming a reality. More details about DoS attacks and solutions are provided in Chap. 6.

Revocation Mechanism

CAS does not provide explicit revocation mechanisms. If a user credential is compromised then the user can be removed from the CAS database and can be denied credential once the current set of credential expires. However, any credentials previously delegated to the user will be honored, Therefore, if an adversary is able to gain an assertion from the CAS system, then it is free to use it as long as the credentials are valid. The only way to prevent such an adversary from creating havoc in the system is to have local access control mechanisms in the local resources. The issued credentials have a limited lifetime and generally expire rapidly enough for most applications.

Inter-operability Issues

As IT systems have become cluttered with many products, services, processes, mostly from different vendors, therefore inter-operability is a major issue if any system is used within an enterprise. Consider a scenario where there are multiple authorization systems. Will CAS be applicable there? Currently, CAS is tied strongly to the Globus toolkit and uses GSI credentials for authentication purposes. Many of the enterprise systems use non-GSI credentials. Moreover, standards like BPEL [85], Web services standards are liberally used in enterprises. Therefore, one way to make CAS acceptable across enterprises would be standardization. Recent efforts suggest that researchers and developers of CAS are looking at this issue. One significant step has been to integrate CAS with SAML. Efforts have also been undertaken to make CAS SAML 2.0 compliant. Furthermore, there are thoughts to use XACML as a means for expressing policies. Details of different Web services security standards can be found in the appendix.

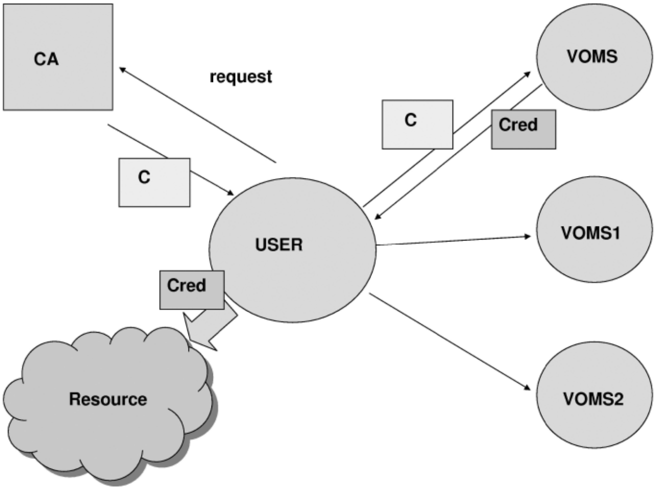


Fig. 5.9. High level view of VOMS

5.3.2 Virtual Organization Membership Service (VOMS)

VOMS is the authorization system developed for the European Data Grid (EDG) as part of the DataGrid and DataTag projects. VOMS and CAS are similar in many respects especially the community they cater to. Both VOMS and CAS cater to the scientific community where a Virtual Organization (VO) is defined as a community of users, institutions, and resources (if any) in the same administrative domain. Moreover, there are Resource Providers (RP) who offer resources like CPU, network, storage, etc. to different VOs based on certain understandings. Similar to CAS, VOMS has a policy database where the authorization policies are stored.

In a VOMS system, a user may be a member of as many VOs as possible, and each VO can be a complex structure with groups and subgroups in order to clearly divide its users according to their tasks. A user, both at VO and group level, may be characterized by any number of roles and capabilities; moreover roles and capabilities can be granted to the user indefinitely or on a scheduled time basis. As mentioned in Fig. 5.9, a user gets user certificate from the Certificate Authority, which is submitted to the VOMS systems. As shown in the figure, a user may get credentials from multiple VOMS systems, and present those credentials to the resources to gain access to the resources.

Since there are a lot of similarities between CAS and VOMS, therefore most of features including characteristics, applicability, security vulnerabilities, and inter-operability are similar. We will not repeat those in this discussion. Readers are advised to go through the CAS discussion for the same. However, we will mention some unique characteristics of VOMS, which make the system different from the CAS system. Following are some of the unique features:

- **Attribute Certificates:** VOMS uses a short lived credential which includes user information, server information, and the validity period. Later versions of VOMS would include the Attribute Certificate (AC), which is actually an Internet standard. VOMS puts the AC in the noncritical extension of the certificate that the user generates, so that it is compatible with “non-VOMS” system. The use of ACs makes VOMS quite compatible with other authorization systems like Akenti.
- **Multiple Roles:** VOMS allows a role based mechanism, where a user can take up multiple roles in multiple VOs.

5.3.3 Enterprise Authorization and Licensing Service (EALS)

Enterprise Authorization and Licensing Service (EALS) has been developed in Software Engineering Technology Labs (SETLabs) of Infosys® Technologies in Bangalore (India). The EALS system has been built with the focus on enterprises and authorization required to cater to the users there. The design of EALS is based on three principles which makes it different from CAS and VOMS system:

- **Pull Based:** Unlike CAS or VOMS, EALS is based on the pull based model where the credentials are pulled from the EALS system. Since EALS has been designed to work with a wide range of middle and grid job submission systems, a system developed on a pull based model fits nicely to the single sign on model that most enterprises adopt. Moreover, pull based models require less overhead from the applications, as the system uses redirection mechanisms to call the appropriate services.
- **Integration with Standards:** One of the design decisions of EALS was to use as much standards as possible so that it can cater to a wide range of systems, as long as the systems follow the standards. Therefore, consciously SAML had been used for transferring authorization credentials.
- **Licensing:** Unlike e-sciences applications, applications used in enterprises have strict licensing requirements. Therefore, the user policies need to be integrated with licensing policies of the applications.
- **Role Based:** EALS allows access to certain resource based on the role a user has, and the permission the role has for the set of resources.
- **Password Based:** Though PKI based systems provide very good security, most of the enterprises still either use password based systems or Kerberos for authentication purposes. The EALS system does not assume a PKI infrastructure and can work with password based system equally effectively. EALS also includes an authentication subsystem, however systems may decide to not use it, and continue to use the enterprises' authentication mechanism.

Figure 5.10 shows the high level view of the EALS authentication and authorization system. The authentication and authorization systems are de-coupled, so that any other system can be used instead of the EALS authenticator. The user submits the job to the Access System (AS). The AS is a generalized term for all the systems including a browser based job submission engine or a customized thick client. The request gets redirected to the EALS authenticator which checks the credentials appropriate for the system. The authenticator generates a SAML authentication assertion generating a token mentioning that the user has been authenticated. The advantage of using SAML is that any system which can generate SAML authentication assertions can be integrated with this system. The job with the token then reaches the Policy Enforcement Point (PEP) which checks with the EALS authorizer for the policy pertaining to the user. EALS authorizer checks with the policy database, which includes the licensing as well as user policies. Say a user **X** wants to run an application **A.** The user X can access resources R1, R2, and R3. The application A is installed in resource R1, R2, R4 (fixed machine license). The authorizer will respond that the user (who is already authenticated) can access resource R1 and R2 through application A. The PEP passes the information to the middleware which can be PBS, CONDOR, or any other open source grid middleware. The information can either be passed as a SAML, if the middleware is capable of understanding, or a restricted view can be given to the middleware on which it can make scheduling decisions.

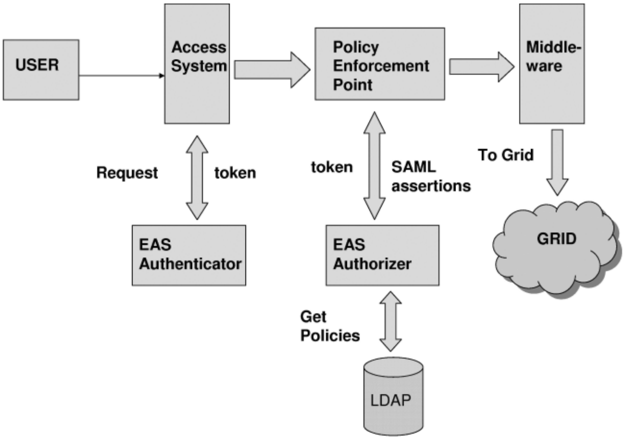


Fig. 5.10. EALS authorization and authentication system

The EALS system is geared towards achieving integration with enterprise level software and middleware. However, there may be a question of performance and scalability of the EALS system compared to the CAS and VOMS systems. Since there are multiple levels of redirections, there is an initial latency involved. Moreover, the pull based model reduces the scalability of the system if the policies are complicated and take some time to parse and understand. However, most of the enterprise grid requirements are within one enterprise and for a select set of users. Therefore, interoperability with the enterprise middleware is of primary importance.

5.4 Resource Level Authorization Systems

Unlike the VO level authorization systems, which provide a consolidated authorization service for the virtual organization, the resource level authorization systems implement the decision to authorize the access to a set of resources. Therefore, VO level and resource level authorization systems look at two different aspects of the grid authorization. As will be shown later in this chapter, the two authorization systems complement each other, and can be implemented together to provide a holistic authorization solution. The different systems that will be discussed as part of the resource level authorization systems are Akenti, Privilege and Role Management Infrastructure Standards Validation (PERMIS), and GridMap system.

5.4.1 Akenti

Akenti, developed by Lawrence Berkeley National Laboratory (LBNL), is an example of resource level authorization. Though developed with Web resources in mind, the concept was later extended to include resources in a grid computing VO setup. The Akenti model consists of *resources*, which may include Web resources, distributed grid resources that are being accessed via a *resource gateway (*or a Policy Enforcement Point or PEP*)* by a set of *users* who are the part of the virtual organization. The model also assumes that each resource may have multiple *stakeholders* having a set of access constraints on the set of resources. The Akenti system allows the users to access the resource(s) based on the identity of the users and the access policy set on the resources by the resource stakeholders. The stakeholders express their access constraints through a set of self-signed certificates which are known to be stored in a secure remote server. The certificates express the attributes a user must have to access the resources. At the time of resource access, the resource gatekeeper or the PEP asks the Akenti server what access the user has to the resource. The Akenti server finds all the relevant certificates, verifies the certificates, and the returns the decision to grant the access to the user.

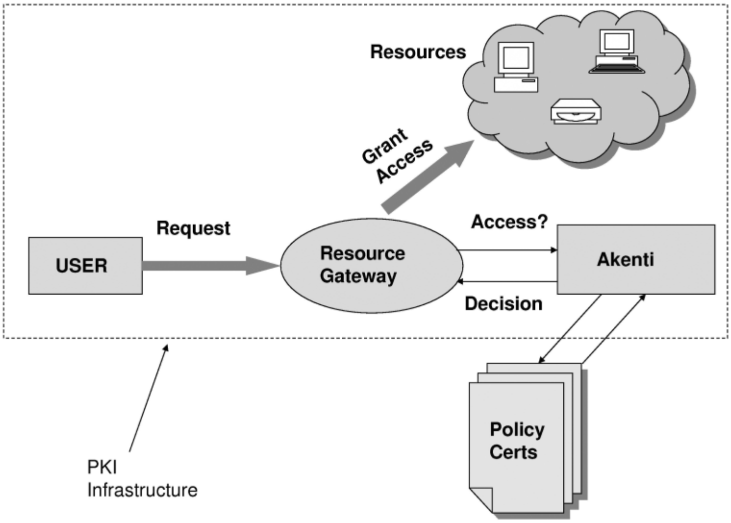


Fig. 5.11. A high level view of the Akenti authorization system

Figure 5.11 shows a high level view of the Akenti authorization system. To access a certain resource, a user sends a request to the resource gateway or the Policy Enforcement Point (PEP) with the user credentials or the user identity certificates. The PEP asks the Akenti server about the decision to grant access to the user. The Akenti searches for the different certificates of the stakeholders which include the policy certificates and the use condition certificates, which may be in remote locations. Once the certificates are obtained, the Akenti server verifies them and grants the user the access to the resource which is enforced by the PEP or the resource gatekeeper.

Akenti assumes that all the certificates follow the X.509 specification, and SSL/TLS channels, are used to authenticate a user who wants to access the resource. Policy certificates are created by unrelated stakeholders from multiple domains, and the access to a certain resources is determined by the combined policy on the resource by the different stakeholders. As is clear from the discussion, Akenti uses a classical pull model. The user presents the credentials to Akenti, and it "pulls" the different certificates from remote locations and makes the access decision.

Akenti policies are expressed in XML and stored in three types of signed certificates: *policy certificates*, which specify the sources of authority for the resources; *use condition certificates*, which specify the constraints that control the access to a resource; and *attribute certificates* which specify the attributes to the users that are needed to satisfy the use conditions. The connection between the different certificates is provided in Fig. 5.12.

Let us look at the different certificates in detail:

- **Policy Certificates:** Policy certificates are signed certificates, generally located with the resources. It contains information regarding the stakeholders and the location of the use-condition certificates of the stakeholders for the resource. These certificates are signed by CAs which are used for validation by the Akenti policy engine. A policy certificate may also contain a list of URLs where to search for the attribute certificates. Different resource groups controlled by Akenti authorization system have a single policy certificate, which is generally stored in a known and secure place.
- **Use Condition Certificates:** The second type of certificate used by the Akenti authorization system is the use-condition certificate. Each stakeholder group for a resource creates one or multiple use-condition certificates for the resources. A use-condition certificate consists of a set of constraints that determine the rights a user must have to access a set of resources. Akenti use-condition certificates allow the use of a X.509 distinguished name as an attribute. It also allows resource or real-time attributes to be specified.
- **Attribute Certificates:** Attribute certificates contain an attribute-value pair and the principle to which the attribute applies. The attribute certificates are signed by the attribute authorities that have been specified in the use-condition certificate. They generally apply to a single resource, or a group of resource or resource realm.

To determine whether a user can be granted access to the resource, the Akenti policy engine finds all the use-conditions by searching in the URLs specified in the policy certificates and verifying the issuer and signature on each certificate. If the use-condition certificate cannot be found for each stakeholder, access to the resource is denied. Attribute certificates are searched by following the URLs specified in either the policy certificate or the use-condition certificate. Akenti caches all the certificates so that the search latency is reduced. The lifetime of the cached certificates is set in the policy certificate for the resource.

Let us now focus on the different characteristics and applicability of the Akenti system.

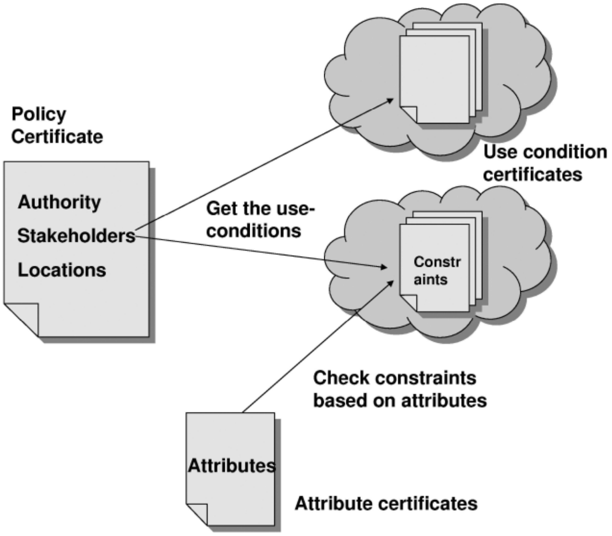


Fig. 5.12. Relationship between the different certificates

Comments on Akenti

Akenti was initially designed to control Web resources, where there are multiple stakeholders for the resources and there are lots of users trying to access the resources. With the advent of grid and collaborative computing, Akenti has been seen as an important alternative to different centralized VO based systems like CAS, VOMS, etc. Akenti does provide flexibility as the use condition certificates can be changed over time. Let us look at the different characteristics and applicability of the Akenti system.

- **Applicability:** Akenti has been designed for providing access to the Web sites and Web resources. In a VO sense, a centralized policy database with a CAS and VOMS system does suffice most of the times. If there are resource specific policies, then that can be implemented at the local resource level. Akenti, on the other hand, can be applied in cases where there is an enterprise level grid having multiple enterprises or stakeholders managing the grid. Then, an Akenti type of system will have a lot of relevance. An integration of VO level systems and Akenti can also be thought of. However, in its current form Akenti cannot be directly used in an enterprise scenario because of lack of support for upcoming standards like SAML, and lack of support for more sophisticated enterprise policies in terms of different types of licenses and so on.
- **Scalability:** Akenti uses a pull model, therefore there is a loss of scalability when the number of users is large, as the Akenti system checks the policies and makes an access decision every time. However, from the administrators' point of view the solution Akenti is scalable as attribute certificates, policy certificates, and use condition certificates are decoupled and can be changed without affecting the main system.
- **Security and Revocation:** Akenti security characteristics are similar to any other authorization system, and do not introduce any extra security vulnerability. However, the certificates need to be securely stored. Compared to CAS and VOMS, since Akenti applies a pull based mechanism, revocation is faster as changes made to the use-conditions are effected immediately.

5.4.2 Privilege and Role Management Infrastructure Standards Validation (PERMIS) Project

The Privilege and Role Management Infrastructure Standards Validation (PERMIS) Project is a European Commission funded project which has members from Barcelona (Spain), Bologna (Italy), and Salford (UK). For the pilot phase, three different business problems in the above mentioned three cities motivated the need for the PERMIS system. The problem in Bologna was to allow the architects to download road-maps of the city, and update them. In Barcelona, the problem was to allow the car hire companies to have online access to the parking ticket database, and to see whether a ticket has been issued, and send the information of the driver to the city so that a fine can be levied in case of violation. In Salford, the problem was to have an online tender application, where only authorized users can submit tenders. As is clear from the above problems, there is a common thread of authorization flowing through each of them, and the PERMIS project resulted in the creation of a role based X.509 privilege management infrastructure that caters to these different applications.

Figure 5.13 shows the high level view of the PERMIS Privilege Management Infrastructure (PMI). A user asks for a certain application or resource and makes a request to the resource gateway or the application gateway. The PERMIS system does not make any assumption regarding the authentication to be employed and leaves it to the domains to deploy the authentication mechanisms. PERMIS assumes a Role Based Access Control (RBAC) model, where each user is mapped to a role and the policies are assigned to the roles. The PERMIS PMI allows access to certain resource based on the policies. PERMIS consists of PERMIS Authorization Enforcement Point (AEF) and the Authorization Decision Point (ADF). ADF contacts the LDAP to obtain the policies and make the authorization decisions which it supplies to the AEF to enforce the decisions.

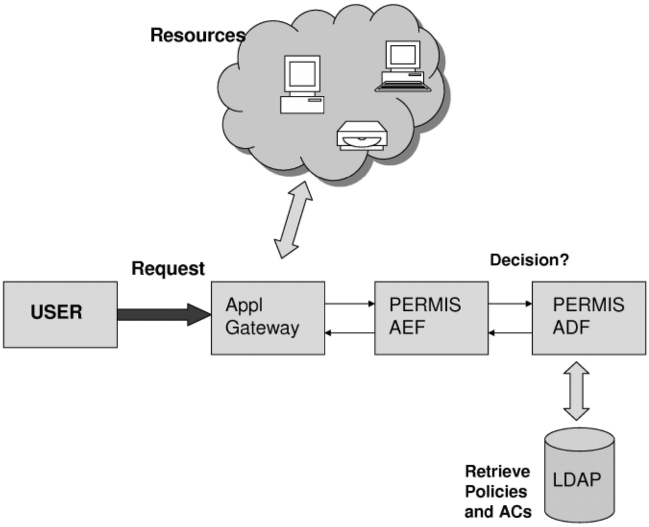


Fig. 5.13. Overview of the PERMIS PMI

The different components of the PERMIS PMI system are the *authorization policy* and *privilege allocator.*

- **Authorization Policy:** The authorization policy specifies *who* has access to *which* resources under *what* conditions. The policy based authorization allows the domain administrator to apply authorization policy for the whole domain in a role based manner. As a policy language, PERMIS uses XML for policy specification. The policy language specifies a set of subjects who are allowed to access resources through the *SubjectPolicy*. The relationships between the different roles are specified in *RoleHierarchyPolicy*, while *SOAPolicy* specifies which sources of authority *(SOA)* are allowed to allocate the roles, and *RoleAssignmentPolicy* specifies the policy for role assignment. *ActionPolicy* specifies the different actions the roles are allowed to perform on the resources like "open" a file and so on. *TargetPolicy* and the *TargetAccessPolicy* connect the roles, the target domains, and the actions they are allowed to perform.
- **Privilege Allocator (PA):** The privilege allocator is a tool to allocate privileges to users. Since PERMIS uses RBAC, PA allocates roles to users in the form of Attribute Certificates (AC). To design a PERMIS system, the first step would be to decide on the number of the type of roles, and assign the users to the roles. The role assignment AC is signed by the Attribute Authority (AA) and hence tamper resistant. The ACs are then stored in a LDAP server and can be accessed publicly. Besides creating the attribute certificates for the role assignment, authorization policy is also created. It is a X.509 certificate consisting of the holder and issuer name as that of the source of authority (SOA) and is signed by the SOA, and stored also in the LDAP.
- **PERMIS AEF and ADF:** The PERMIS authorization enforcement function and authorization decision function provide authorization capabilities. AEF authenticates a user and requests the ADF to make an authorization decision. The ADF accesses the LDAP servers, gets the policies and the attribute certificates, and then makes an authorization decisions based on the obtained information. The interaction between AEF and ADF can happen through four function calls, namely, initialize, getcreds, decision, and shutdown. During the initialization call the AEF passes the name of the trusted SOA and the list of URIs from where the ADF can get the policy and role ACs. Decision calls are made to ask the ADF for authorization decision based on all the policy and role ACs. Getcreds are used to manage a session, where the AEF can specify the

duration of the open session after which the credentials can be refreshed.

Both PERMIS and Akenti look at the same problem of providing authorization in a distributed scenario. Architecturally, both are similar and do seem to be identical in lot of respects. However, there are certain differences which we would like to point out here.

- **Based on Policies:** The use of attribute, policy, and use condition certificates helps the Akenti system to work on hierarchical and distributed policies. The assumptions of multiple stake holders for resources motivated the designers for the use of the three kinds of certificates. PERMIS on the other hand, has policy attribute certificate which is stored in an LDAP server.
- **Support for RBAC:** PERMIS supports a classical RBAC model, where users are mapped to roles and roles are mapped to permissions. Akenti can support both DAC and RBAC models, where principals can be given permissions or group memberships, and the group attributes can be given permissions.
- **Policy Decisions:** In a PERMIS system, the ADF replies with a "yes"/"no" Boolean decision. However, the AEF can ask the ADF many times for multiple resources and actions. Akenti returns with a capability certificate which mentions what the user is allowed to do. PERMIS does not have the capability for such a capability certificate.
- **Push vs. Pull:** While Akenti works strictly in a pull model, PERMIS can be configured to use either a pull or a push model.

The community authorization service has been used as an authorization system for Globus. However, use of Security Assertions Markup Language (SAML) allowed the integration of the Globus toolkit with PERMIS authorization system. This case study not only demonstrates the integration of Globus with PERMIS, but it also demonstrates the usefulness of standardization as a powerful tool for integration. In this case an important standard like SAML had been used for this purpose.

Figure 5.14 shows how SAML can be used in case of authorization purposes. Since SAML credentials can be used for both authentication and authorization purposes, the SAML message can flow from the authentication system to the Policy Decision Point (PDP) before passing to the Policy

Enforcement Point (PEP). The PEP can then enforce the policy of whether the user is allowed to access the set of resources or not.

PERMIS and Globus – A Case Study

Figure 5.15 shows the high level working of the PERMIS authorization system on top of the Globus toolkit described in [86]. SAML is used for transferring the access request from the container or the application gateway to the PERMIS PMI. The response is also packaged in the form of SAML response to the application gateway which uses it to allow the users from accessing the resources. It is to be noted that the PERMIS authorization system uses the PKI based Globus authentication mechanism.

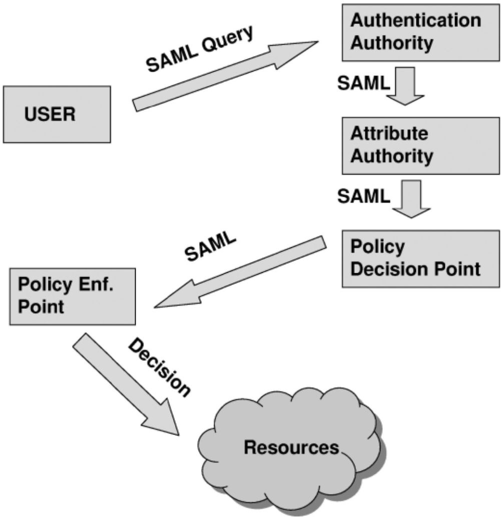


Fig. 5.14. Use of SAML in authorization

To make PERMIS system work with Globus, some extensions had to be added to the SAML to make the system more efficient. Moreover, PERMIS was modified to interact with Globus over SAML.

- **Extensions to SAML:** A set of extensions to SAML had been proposed and used as part of the integration exercise. A standard

SAML response contains the list of actions a particular user is allowed. This is useful in case of delegation and authorization transfer as in case of EALS. However, in case of a more direct sort of interaction where a PEP is only interested in an access/denied sort of answer, this may lead to a lot of overhead and hence inefficiency. A more concise version of SAML request/response had been used to improve efficiency.

- **Extensions to PERMIS:** While the extensions to SAML were mainly mandated by the need for efficiency, the extensions to PERMIS were mandated by necessity. ADF was modified to pose as a grid service so that the integration with the Globus toolkit becomes possible. Initially, PERMIS supported only LDAP distinguished names. However, to make the system more amenable to integration with different types of naming policies Uniform Resource Identifier (URI) were used to target names, and policies were extended to support URI target identifiers.

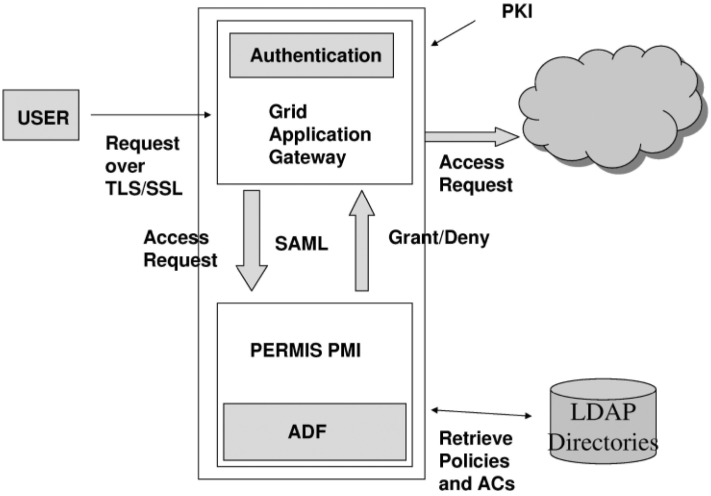


Fig. 5.15. Grid resource access over SAML using PERMIS

5.4.3 Authorization Using GridMap

This is the earliest authorization system used in Globus. Though more sophisticated systems like Community Authorization Service (CAS) and other authorization systems discussed in this chapters have been developed, GridMap is still one of the most widely used authorization system is Globus mainly due to its simplicity. In a GridMap system, the static policies of *which* user can access the resource and *how* is placed in each local resource. The decision to grant access to a resource is based on the information present in the GridMap file. As mentioned earlier, this authorization system is simple to implement and does not require too much overhead. However, lack of scalability really hampers the use of GridMap system in a wide scale.

- **Scalability:** The main reason for the development of the CAS system is the lack of scalability of the GripMap authorization model. It scales very poorly in terms of administrative scalability. In a dynamic grid environment, where Globus is mostly deployed, GridMap file needs to be changed continuously which hampers the performance of the system tremendously.
- **Security:** Unlike CAS and VOMS systems, the denial-of-service effect is pretty limited to the resource the GridMap file pertains to. However, the GripMap file is stored without encryption in the host system. Revocation also can be done fast, as only one file needs to be changed to effect the changes.

5.5 Comparing the Different Authorization Systems

In this section we will summarize the different authorization systems described in this chapter. First we will compare the systems with respect to the different characteristics mentioned before. Later, we will provide a roadmap of the possible adoption of the systems in the enterprise scenario.

5.5.1 Comparison

Let us now compare the different authorization systems described in this chapter with respect to the scalability, security, revocation, and interoperability characteristics.

Scalability

Push based systems are generally more scalable than their pull-based counterparts. Furthermore, for administrators it is more scalable to have the policies in a centralized system rather than in each and every node of the grid system. In both these counts, both CAS and VOMS score highly. Since both of them use a push-based architecture and has a centralized database. EALS, on the other hand, uses a pull-based model and a centralized policy database. Therefore, it is restricted in terms of number of users supported. The resource based models Akenti and PERMIS are both scalable, with PERMIS supporting both Push and Pull based models. The worst in this category is the Gridmap system as administrators need to update each and every system for addition and deletion of nodes, users, or policies.

Security

Most of the systems mentioned in this chapter are immune to masquerade attack as they support authentication of some type. Certificates are most prevalent means of authentication while EALS supports passwords, certificates, or other types of credentials like biometrics. However, most on the push based systems are prone to DoS attacks as most of them depend on a centralized database for storing policies. Since VOMS supports multiple stakeholders, even if one of the databases storing a particular stakeholder’s certificates/credentials is under DoS attack, the other resources would be unaffected. Pull-based systems like Akenti and EALS can distribute the requests to multiple servers in case a DoS attack is detected. Gridmap is mostly unaffected as the attacker needs to attack a significant number of resources to have a big impact.

Revocation

CAS and VOMS do not have explicit revocation mechanisms. Therefore, once an adversary gains access to the system then it can access all the resources based on the obtained credentials. EALS and Akenti, being pull-based systems, have inherent revocation mechanisms as these can be added to the policies and the effect will be immediate. The same argument can be extended to the Gridmap system also; however the administrator needs to change the policy in each and every resource in the grid system.

Inter-operability

Another characteristic which is important to the grid authorization systems is how inter-operable the systems are. CAS and PERMIS have been made to inter-operate using SAML standards. However, if they are to be used extensively in the enterprises, policies need to be exposed as XACML standards and exchanged using SAML. Also, there is a need to integrate with different identity management systems like LDAP, Windows ® Active Directory, and so on. One step in that direction would be to integrate with the Liberty framework for federated identity management. EALS is the most advanced in this regard as it has adapters for most industry products and adheres to most of the common industry standards and practices. Table 5.1 compares the different systems mentioned in this chapter.

Table 5.1. Comparisons between the different authorization systems

	VO Based			Resource Based		
Params	CAS	VOMS	EALS	Akenti	PERMIS	Grid-Map
Push/ Pull	Push	Push	Pull	Pull	Push or Pull	Pull
Users Supported	High	High	Medium	Medium	High	Medium
Administrative Overhead	Low	Low	Low	Low	Low	High
Authentication	Using GSI	Using GSI	Passwords/Certificates	Certificates	Certificates	Using GSI
Revocation	No	No	Fast	Fast	Can be fast	Have to be updated
Interoperability	Uses SAML	Can use SAML	Through SAML, XACML	May be complex in some cases	Through SAML	Minimal
Decision Making	Requires separately	Requires separately	Integrated in Scheduler and License manager	Single step; through Capability certs	Two steps; “yes/no answer”	Based on policies
Multiple stakeholders	No	Yes	Yes	Yes	No	No

5.5.2 Roadmap to Grid Authorization Systems

Authorization is perhaps one of the most important needs for an enterprise today. Though grids are mostly concentrated to high computing jobs or enterprise batch jobs, the grids are shared across the enterprises, sometimes across geography. Therefore, authorization is needed mainly for accounting purposes. For example, there are three departments A, B, and C in an enterprise. The enterprise wants to enforce a host of different policies based on the usage of the grid. The policies can be really complex, as licensing information, transient system level information, and user level information needs to be incorporated. Following are some of the recommendations:

- **Beyond Schedulers:** Most of the authorization decisions are currently implemented at the scheduler level. The enterprise grid vendors like Altair® PBS [87], Platform® LSF Multicluster [88] requires administrators to manage policies so that the scheduler is able to schedule based on the implemented policies. This is not a scalable model and complex policies based on all the different enterprise requirements cannot be easily handled. Two ways can be used to apply authorization. First would be to provide adapters so that the authorization systems can interact with the underlying schedulers, and second would be through standardization.
- **Beyond Batch Jobs:** Currently, most of the grid systems are used as batch job systems in enterprises. However, to take grid forward it needs to cater to enterprise needs of subsecond jobs, messaging systems, workflows (and possible integration with BPEL), and so on.
- **Towards Federation:** Many enterprises are now looking at federated identity management solutions. Grid authorization systems should be able to interact with the Liberty frameworks and Web services standards to make this possible.

5.6 Chapter Summary

Grid authorization systems are extremely important in the grid context mainly due to the distributed nature of the grid systems. The different characteristics of grid authorization systems are security, scalability, revocation, and inter-operability. Like any other systems, security is important where the adversary can pose as a valid user or compromise the authorization system as a whole. Grid systems may have thousands or potential users, hence scalability assumes enormous importance. Revocation criterion is determined by whether the system uses pull-based or push-based authorization. Finally, grid systems may also involve multiple stakeholders and encompass multiple authorization domains and systems. Hence, interoperability is extremely critical. To organize the discussion of the grid based authorization systems, we have categorized the systems into two main types: Virtual Organization (VO) based systems, and resource based systems. Virtual organization level systems have a centralized authorization system which provides credentials for the users to access the resources. Resource level authorization systems, on the other hand, allow the users to access the resources based on the credentials presented by the users. Examples of VO level grid authorization systems are Community Authorization Service (CAS), Virtual Organization Membership Service (VOMS), and Enterprise Authorization and Licensing System (EALS). Examples of resource level grid authorization systems are Gridmap, Akenti, and Privilege and Role Management Infrastructure Standards (PERMIS). These different systems have been discussed in this chapter. In the next chapter, we will look at the third component of the architecture issues, *viz.* grid service security.

6 Service Level Security in Grid Systems

6.1 Introduction

Last week, when I visited my bank, I found that there was a huge queue in front of the transaction counter. It took me half an hour to reach the counter and carry out my transaction. The queue, on that day, was created because of huge demand for bank transaction, as it was the last day before a series of holidays. Therefore, a huge surge of demand affected the Quality-of-Service (QoS) that I generally receive and expect from my bank. On that day, there was a legitimate reason for the delay. However, the same effect can be simulated to create delays in the banking transactions. Let us imagine that there is an adversary, who wants to delay the services offered by the bank. He can employ a few people who can unnecessarily waste bank's time and thus reducing the overall service offered by the bank to the legitimate customers. It may be because of personal enmity or competition, or even just for fun. Such a malicious action is theoretically feasible. However, it is hard to imagine someone employing such a delaying tactics to reduce the quality of service in the banks. The reason is that the amount of effort involved may be more than the effect that the adversary achieves. In the digital world, however, it is an entirely different issue. In the digital world, unlike in the case of real worlds, it is possible to assume multiple identities and create attacks on the systems, servers, and infrastructure providing some valuable services. There have been instances, especially in the Internet scenario, where malicious adversaries created attacks to reduce the service to the customers. The extreme impact of such a type of attacks is called Denial-of-Service (DoS) attacks, where the services are denied to the legitimate end-users using a variety of techniques. In this chapter, we will try to analyze the different types of attacks and solutions which can be effective in a grid based environment.

6.1.1 Components of Service

The word service or *services* is finding wide usage is day-to-day business transactions. According to Merriam Webster, one definition of service is "*the occupation or function of serving,*" or "*the work performed by one that serves.*" As one can observe, the definition of service is intrinsically linked with the *service provider* or one who serves. Therefore, a service should always contain four basic components:

- A *Service Provider* or one who is providing the service to the users.
- A set of *Service Consumers* who accesses the service provided by the Service Provider.
- A *Service Infrastructure* on which the service is provided.
- A set of *Service Publishers* which publish the type and nature of service provided.

We can extend the definition of service and its components to real-life examples. Let us take the example of a banking service. Here the service provider is the bank with the customer service executives being the front-end to whom the customers of the bank are exposed to. The service consumers are the customers of the bank, and the service infrastructure includes the host of database and other servers, communication networks, and the buildings and different other infrastructure that support the bank. Finally, the service publisher may be a Web site which describes the services provided by the bank, which may help the service consumers in making a service decision. Generally, service is published in multiple channels. For example, there may be Web sites indicating the number of banking service providers in a district, and banks may have call-centers to provide more details about the service they are providing.

6.1.2 Service Vulnerabilities

Let us now step into the shoes of the adversaries who are hell-bent in disrupting the service offered by a service provider. Generally, the adversaries go by the principle of maximum effect. Among the four components mentioned in the previous subsection, the service infrastructure and the service publisher, if compromised, will have the greatest effect. The reason is that, if infrastructure is compromised the service to a large number of customers is disrupted. Similarly, if the service publisher publishes wrongly or maliciously, the effect will be devastating. Furthermore, the effects can be minimized if the infrastructure is protected, or if the publisher publishes through multiple channels. However, all these come under the purview of service disruption prevention mechanisms. If we take a look at different service providers, we find that enormous efforts are being put in to make the infrastructure secure. There are also laws and regulations to keep the adversaries from manipulating the published information. This is true in case of physical world. Therefore, in digital world also, there are needs for techniques and methods to counter such threats. Before looking at the different methods, techniques, and research outputs available in the domain of service level security, let us look at the different vulnerabilities and threats present there.

The different categories of threats present in services are Quality of Service (QoS) violation, unauthorized service access, and Denial-of-Service (DoS).

QoS Violation

Let us assume that there is a pizza delivery company whose unique selling point is to deliver pizza within 30 minutes to the customer call. If the company is not able to deliver within the stipulated time, the customer gets a free pizza. If a malicious "pizza-eater" tries to stop the company from delivering on time, the "pizza-eater" gets a free pizza and the company loses a lot of goodwill. Now, translate the same problem to the digital world. A company may have end up losing a lot of money if Service Level Agreements (SLA) are not met.

Unauthorized Service Access

In this type of threat, illegitimate or unauthorized users get access to the service. This problem is similar to the traditional problems of authentication and authorization. Standard authentication and authorization techniques discussed in this book can be used to solve this problem.

Denial-of-Service (DoS)

Perhaps the most deadly of the service level threats is the threat of denying the service to the service consumers. This type of attack is popularly known as Denial-of-Service (DoS) attack. In this type of attack, the service infrastructure is crippled by a huge amount of artificial load resulting in denial of service to the legitimate users or service consumers. The importance of securing DoS attacks especially for applications over the Internet has grown rapidly due to a series of attacks that shut down some of the world's most high profile Web sites, including Amazon® and Yahoo® Several such attacks have also been reported in CERT advisories [89]. DoS attack detection and prevention is perhaps the most active area of research in security. Research as well as implementation ideas will be discussed subsequently in this chapter.

6.2 DoS Attacks and Countermeasures

Denial-of-Service (DoS) attacks have a simple objective, to deny the service to the service consumers. This is generally achieved by overwhelming the service infrastructure with huge amount of data packets. In DoS attacks, the packets are routed correctly but the destination and the network becomes the target of the attackers. DoS attacks are very easy to generate and are very difficult to detect, and hence are attractive weapons for the hackers. In a typical DoS attack, the attacker node spoofs its IP address and uses multiple intermediate nodes to overwhelm other nodes with traffic. DoS attacks are typically used to take important servers out of action for a few hours, resulting in service denial for all the users served by the server. It can also be used to disrupt the services of the intermediate routers. Generally, DoS attacks can be categorized into two main types: (a) ordinary and (b) distributed. In an ordinary network based denial of service attack, an attacker uses a tool to send packets to the target system. These packets are designed to disable or overwhelm the target system, often forcing a reboot. Often, the source address of these packets is spoofed, making it difficult to locate the real source of the attack. In the Distributed DoS (DDoS) attack, there might still be a single attacker, but the effect of the attack is greatly multiplied by the use of attack servers known as "agents." To get an idea of the scope of this attack, over 5000 systems were used at different times in a concerted attack on a single server at the University of Minnesota. The attack not only disabled that server but denied access to a very large university network [89].

6.2.1 Effect of DoS attacks

Let us now take a look at the trends of different types of cyber attacks that are taking place in the Internet scenario. Figure 6.1 shows the different types of attacks (in percentage figures) based on the CSI survey of 2001 [90]. The chart shows that DoS attacks is one of the most important attacks as perceived by the respondents. Though the survey had been done a few years ago, evidences suggest that the percentage of DoS attacks has increased rather than showing any decreasing tendency. Figure 6.2 shows the different categories of attackers who are mostly responsible for attacking the Internet infrastructure, based on the same survey. It shows that unlike the popular imagination of cyber terrorism, and corporate based cyber warfare, most of the attackers are independent hackers or disgruntled employees doing it for the fun or due to animosity against his/her employer.

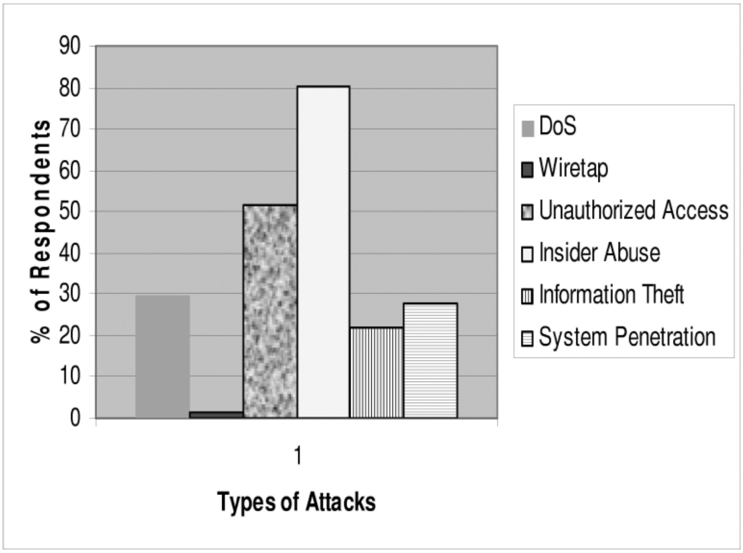


Fig. 6.1. Attack trends [90]

Though there have been indications about the importance of DoS attacks, the actual data are mostly hidden because most companies prefer to keep the attack stories hidden from public. One of the interesting works on the importance of DoS attack has been carried out by Moore et al. [91]. In this paper, the authors have tried to answer the simple question about how prevalent are denial-of-service attacks in the Internet today. The results are far-reaching and remain to this day an important warning about the importance of tackling DoS attacks. As a means to demonstrate this, the authors described a traffic monitoring technique called “backscatter analysis” for estimating the worldwide prevalence of denial-of-service attacks. Using backscatter analysis, the authors have observed that 12,805 attacks on over 5000 distinct Internet hosts belonging to more than 2000 distinct organizations during a three-week period. The authors further estimated a lower-bound on the intensity of such attacks – some of which are in excess of 600,000 packets-per-second. The paper showed the importance of DoS attacks in the context of the Internet.

As is quite evident from the above paragraph, denial-of-service attack is becoming one of the most potent attacks carried out over the Internet. With whatever little data is available, the damages seem to run in millions. Most of the attackers are amateurs rather than corporates or rogue countries engaged in cyber warfare. Now the natural question that comes to the mind is, how do these amateurs have the enough firepower to break the security of biggest corporates of the world? There are two reasons for this. Firstly, the wide availability of DoS launching tools. If one searches for the "DoS attack tool," one would get over 1000 hits and lots of open freeware for launching DoS attacks. Secondly, the defense against this type of attack is still in its nascent stage, and lots of research is required to provide enough protection against DoS attacks. For the rest of the chapter, we will concentrate on the different types of DoS attacks and techniques and methods available to mitigate them.

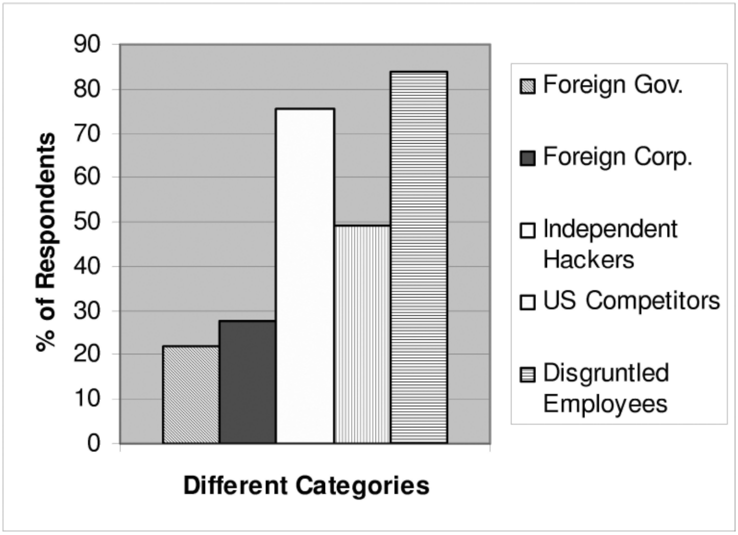


Fig. 6.2. Different types of attackers

The above paragraph highlights the importance of DoS attacks in an Internet scenario. Grid infrastructure is also distributed in nature and DoS attacks are quite relevant in grid scenario as well. Most of the grid infrastructure that has been implemented is much smaller than the Internet and hence the DoS attacks have limited impact. However, with the grid infrastructure destined to grow the impact is going to get bigger and there is a need to understand the vulnerabilities in the underlying infrastructure and implement the existing solutions. In the rest of this chapter, we are going to provide an overview of different DoS attacks along with solutions and research ideas.

6.2.2 Distributed Denial-of-Service Attacks

One of the deadliest forms of DoS attacks is when the attackers are distributed in nature. Such an attack is called Distributed Denial-of-Service (DDoS) attack. According to the CIAC (Computer Incident Advisory Capability), the first DDoS attacks occurred in the summer of 1999 [92]. In February 2000, one of the first major DDoS attacks was waged against Yahoo.com. This attack kept Yahoo® off the Internet for about 2 hours and cost Yahoo® a significant loss in advertising revenue [93]. Another recent DDoS attack occurred on October 20, 2002 against the 13 root servers that provide the Domain Name System (DNS) service to Internet users around the world. They translate logical addresses such as www.abc.edu into a corresponding physical IP address, so that users can connect to Web sites through more easily remembered names rather than numbers. If all 13 servers were to go down, there would be disastrous problems accessing the World Wide Web. Although the attack only lasted for an hour and the effects were hardly noticeable to the average Internet user, it caused 7 of the 13 root servers to shut down, demonstrating the vulnerability of the Internet to DDoS attacks [94]. If unchecked, more powerful DDoS attacks could potentially cripple or disable essential Internet services in minutes.

An example of distributed denial-of-service attack is provided in Fig. 6.3. In this example, the attacker hacks into one of the machines and uses it to launch the attack. This machine is called the attack host. The attacker then installs the attack code into the slave machines, also known the zombies. These slave machines are generally the machines in the same network as the attack host on which it has some amount of control. The attacker and the zombies want to attack the victim as shown in the figure. The zombies send a huge number of packets to a set of machines called reflectors, with the return address as that of the victim. The reflector, unknowing of the whole episode, returns the response to the victim. Overwhelmed by the number of packets, the victim goes down. The attack is easy to generate, as downloadable tools are available to install the zombies and launch the DDoS attack. The attack is very difficult to prevent and detect as the packets are coming from a host of different reflectors, who themselves are unaware that they are aiding in the DDoS attack.

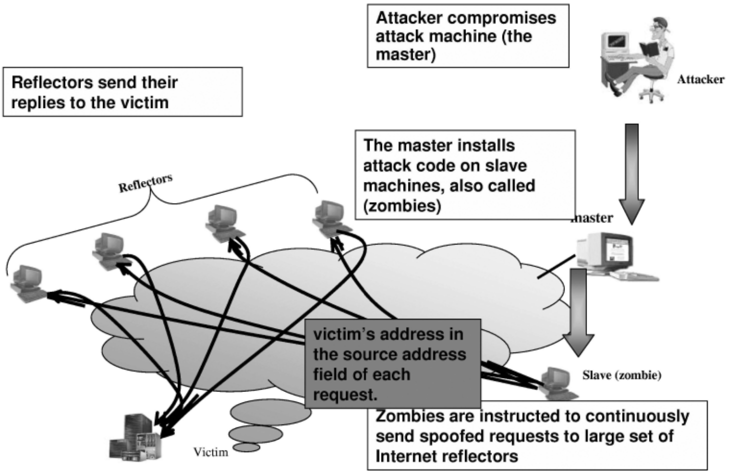

Attacker compromises attack machine (the master)
Attacker
Reflectors send their replies to the victim
The master installs attack code on slave machines, also called (zombies)
Reflectors
master
victim's address in the source address field of each request.
Slave (zombie)
Zombies are instructed to continuously send spoofed requests to large set of Internet reflectors
Victim

Fig. 6.3. Example of DDoS attack using a relector

Let us now discuss some of the common DDoS attacks carried out by the malicious adversaries. Most of these attacks target a particular network protocol like the TCP, UDP, etc. Subsequently, we will discuss some of the common modes of attacks. However, it is to be noted that the expanse of DDoS attacks go beyond these attacks only, and can be used as a technique with other commonly used protocols.

SYN Flood Attacks

Perhaps the most popular among the DDoS attacks is the SYN flood attack. This type of attack targets the Transfer Control Protocol (TCP) to create the service denial. The TCP protocol includes a three-way handshake between sender and receiver, before data packets are sent. The protocol works in the following manner:

- The initiating system sends a SYN (Synchronize) request. This indicates the system's intention in creating a TCP session.
- The receiving system sends an ACK (acknowledgement) with its own SYN request. This indicates that the receiving system would like to carry on with the connection.
- The sending system then sends back its own ACK and communication can begin between the two systems. It has been proved that three-way handshake is an efficient and effective way in creating a network connection. If the receiving system is sent a SYN_X packet but does not receive an ACK_{Y+1} to the SYN_Y it sends back to the sender, the receiver will resend a new ACK + SYN_Y after some time has passed. The processor and memory resources at the receiving system are reserved for this TCP SYN request until a timeout occurs.

In a DDoS TCP SYN flood attack, the attacker instructs the zombies to send bogus TCP SYN requests to a victim server in order to tie up the server's processor resources, and hence prevent the server from responding to legitimate requests. The TCP SYN attack exploits the three-way handshake between the sending system and the receiving system by sending large volumes of TCP SYN packets to the victim system with spoofed source IP addresses, so the victim system responds to a non-requesting system with the ACK+SYN. When a large volume of SYN requests are being processed by a server and none of the ACK+SYN responses are returned, the server begins to run out of processor and memory resources. Eventually, if the volume of TCP SYN attack requests is large and they continue over time, the victim system will run out of resources and be unable to respond to any legitimate users. SYN flood attacks are illustrated in Fig. 6.4.

PUSH + ACK Attacks

In this type of attack, the attacker again uses the properties of the TCP protocol to target victims. In the TCP protocol, packets that are sent to a destination are buffered within the TCP stack and when the stack is full, the packets are sent to the receiving system. However, the sender can request the receiving system to unload the contents of the buffer before the buffer becomes full by sending a packet with the PUSH bit set to one. PUSH is a one-bit flag within the TCP header. TCP stores incoming data in large blocks for passage on to the receiving system in order to minimize the processing overhead required by the receiving system each time it must unload a non-empty buffer. The PUSH + ACK attack is similar to a TCP SYN attack in that its goal is to deplete the resources of the victim system. The attacking host or the zombies and the reflectors send TCP packets with the PUSH and ACK bits set to one. These packets instruct the victim system to unload all data in the TCP buffer (regardless of whether or not the buffer is full) and send an acknowledgement when complete. If this process is repeated with multiple agents, the receiving system cannot process the large volume of incoming packets and it will crash.

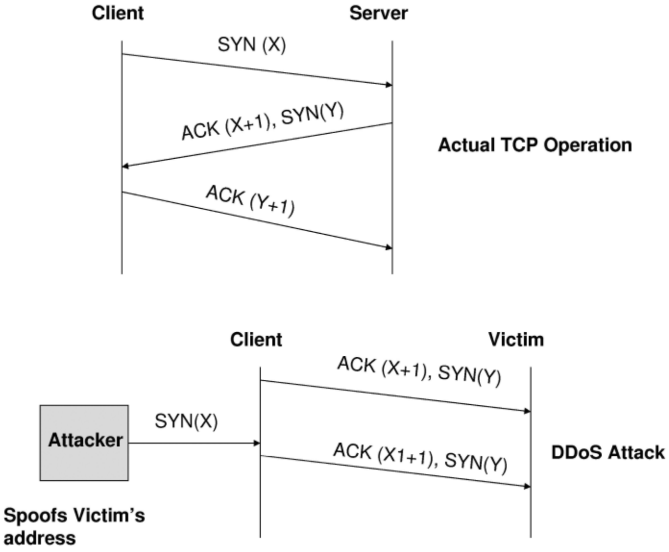


Fig. 6.4. TCP SYN flood attack

Smurf Attacks

In a DDoS Smurf attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing), with the return address spoofed to the victim's IP address. The attacking packets are typically ICMP ECHO REQUESTs, which are packets (similar to a "ping") that request the receiver to generate an ICMP ECHO REPLY packet. The amplifier sends the ICMP ECHO REQUEST packets to all of the systems within the broadcast address range, and each of these systems will return an

ICMP ECHO REPLY to the target victim's IP address. This type of attack amplifies the original packet tens or hundreds of times.

Routing Table "Poisoning"

Routing tables are used to route packets over the Internet. They are created by exchange of routing information or updates between routers. Poisoning attacks refer to the malicious modification or "poisoning" of routing tables. This can be achieved by maliciously modifying the routing information update packets required by the routing protocols. This attack can result in wrong entries in the routing table and could lead to a breakdown of one or more domains of the Internet.

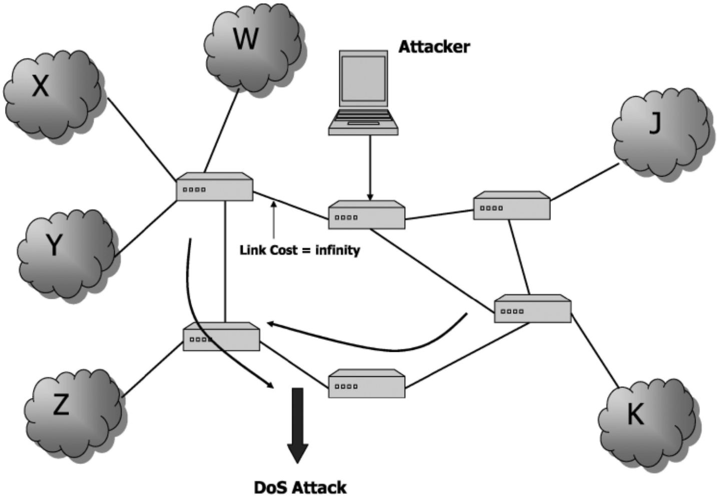


Fig. 6.5. A router attack scenario

Attack on the Internet infrastructure can lead to enormous destruction, as different infrastructure components of the Internet have implicit trust relationship with each other. Consider the scenario listed in Figure 6.5. In this scenario, an intruder wishes to attack Domain Z, which contains a high-profile server. Most of the links are fairly heavily loaded but are under capacity (70 - 80% usage). The attacker compromises Router A, so that the router increases the cost of link B to an artificially high value, say

(10,000). Traffic, in the Internet, is generally routed along the shortest path. Since link B has a high cost, packets will be routed around B. Thus, packets will be routed through the border router of domain Z. This would cause enormous congestion at domain Z. Artificial congestion, thus created, will slow down the services to the clients of domain and many clients will be denied access to the server located in that domain.

DNS Hacking

Domain Name System (DNS) is a distributed, hierarchical global directory that translates machine/domain names to numeric IP addresses. The DNS infrastructure consists of 13 root servers at the top layer, top-level domain (TLD) servers (.com and .net), as well as country code top-level domains (.us, .uk and so on) as the lower layers. Due to its ability to map human memorable names to numerical addresses, its distributed nature, and its robustness, DNS has evolved into a critical component of the Internet. Therefore, an attack on the DNS infrastructure has the potential to affect a large portion of the Internet.

DNS consists of a distributed database which lends to its robustness and also leads to various types of vulnerabilities, which can be categorized into three main types:

- **Cache poisoning:** Generally, to hasten the process of query response, DNS servers store the common information in a cache. If the DNS server is made to cache bogus information, the attacker can redirect traffic intended for legitimate site to a site under the attacker's control.
- **Server compromising:** Attackers can compromise a DNS server, thus giving them the ability to modify the data served to the users. These compromised servers can be used for cache "poisoning" or DoS attacks on some other server.
- **Spoofing:** In this type of attack, the attacker masquerades as a DNS server and feeds the client wrong and/or potentially malicious information. This type of attack can also redirect the traffic to site under attacker's control and also launch a DoS attack on the unsuspecting client.

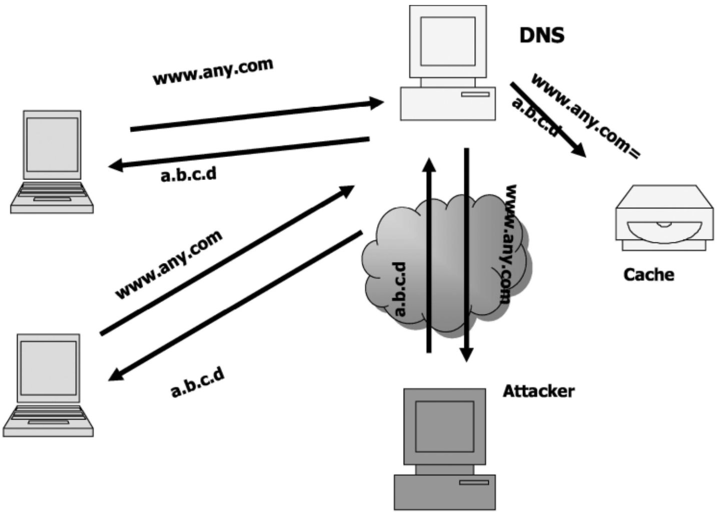


Fig. 6.6. DNS cache poisoning

Figure 6.6 shows an example of Cache poisoning

Application Layer DoS

Unlike most of the previous attacks, which exploit the infrastructure to launch a denial of service attack, application layer DoS attacks specifically target the application vulnerabilities to satisfy their malicious intents. This type of attacks has several advantages compared to the traditional DoS attacks. Firstly, the traditional DoS monitoring techniques are unable to detect attacks of this nature as most of the times the attacks are indistinguishable from normal traffic. Secondly, they can sometimes represent a much more sophisticated attacking tool for the adversary which can be launched much more easily than traditional attacks. Lastly, most of these attacks are very difficult to trace as most of them use HTTP or HTTPs to carry out the attacks. Some common attacks of this type are as follows:

- **XML Based DoS Attacks (XDoS):** In this type of attack, the attacker takes advantage of the XML to launch a denial-of-service attack on the host by exhausting the resources like the memory, CPU, etc.

XML documents with loops and long hierarchies can sometimes exhaust the resources. The most popular example of XDoS attack is the *entity expansion attack*. This type of attack uses the echo capabilities to launch a full-fledged attack by expanding the contents of a message. Another attack generally used is using arrays in SOAP message. A Web service that expects an array can be the target of a DoS attack by forcing the SOAP server to build a huge array in the machine's RAM, thus inflicting a DoS condition on the machine due to memory pre-allocation. In another attack called the *Quadratic Blowup Attack*, the attacker defines a single huge entity (say, 500 KB), and references it a significantly high number of times (say, 50,000 times), inside an element that is used by the application.

- **Schema Poisoning:** XML schemas provide formatting instructions for parsers when interpreting XML documents. Denial-of-service attacks against the grammar are straightforward if the schema is compromised. Manipulation of data, launching denial-of-service attacks in other servers, etc. can be easily achieved.
- **Routing Detours:** WS-Routing specifications provide mechanisms to route messages to destinations through complex environments. Similar to the routing table poisoning attacks described above, a compromised node can result in looping and partitioning in the environments.
- **SQL Injection Attack:** This type of attack allows the adversary to launch different type of attacks by executing multiple commands in the input field. It can be used to manipulate the data as well as denial-of-service attacks like buffer overflow attack.
- **Replay Attack:** In this type of attack, the attacker can launch a DoS attack by replaying old messages. Since the messages had been sent from valid IP addresses, the messages are considered to be valid also. A timestamp based approach can prevent this attack.

6.2.3 Existing DoS Countermeasures

Solutions proposed in literature for DoS attacks, can be broadly categorized as (i) preventive and (ii) reactive. Preventive DoS solutions take precautionary steps in preventing DoS attacks. A wide range of solutions have been proposed, however, this problem still remains an open one. The reactive solutions aim at identifying the source of the attacks. This is very important because attackers spoof their addresses, thus techniques are needed to trace back to the source of the attack. We discuss in the subsequent subsections some of the interesting solutions.

6.2.4 Preventive DoS Counter-measures

The preventive DoS techniques are used to detect and reduce the effectiveness of the attacks. In this section we will talk about some of the methods that have been proposed to detect and prevent the DoS attack, namely filtering, location hiding, and the throttling techniques.

Packet Filtering Techniques

All preventive DoS detection techniques are based on some prior information, on the basis of which the filtering is carried out. A few filtering techniques are described in Cisco® white papers [95]. One of the most common methods for detecting and preventing potential attacks is to use *egress filtering*. Egress filtering refers to the practice of scanning the packet headers of IP packets leaving a network (egress packets) and checking to see if they meet certain criteria. If the packets pass the criteria, they are routed outside of the subnetwork from which they originated. If the filter criteria are not met, the packets will not be sent to the intended target. Since one of the features of DDoS attacks is spoofed IP addresses, there is a good probability that the spoofed source address of DDoS attack packets will not represent a valid source address of the specific sub-network. If the network administrator places a firewall or packet sniffer in the subnetwork that filters out any traffic without an originating IP address from this subnet, many DDoS packets with spoofed IP source addresses will be discarded, and hence neutralized. This is a common technique and has been deployed as a defense mechanism in routers [95]. It is to be noted that these types of measures can minimize attack to some extent, however can no way guarantee an absolute defense against DoS attacks.

Similar to egress filtering, different ingress filtering mechanisms have also been proposed and implemented. In this type of mechanism, the filtering is done on all packets coming into the network. In [96], the authors have presented techniques for preventing DoS attacks through filtering techniques. They presented a technique called Distributed Packet Filtering (DPF), where decision to drop or accept the packet is made based on the incoming packet interface. The route information plays a major role in determining whether a packet would be dropped or not. The route information stored at each node indicates the source address and the corresponding interface that the packet is supposed to come from.

Application Filtering

Depending on just filtering packets does not provide enough protection from XDoS attacks where there is a need to understand the XML documents to prevent the Denial of Service attack. The XML level firewalls examine the received SOAP messages or native XML messages. Several companies like Reactivity have developed this type of firewalls. Once the target Web service is resolved, the XML firewall can apply a stored security policy based on the target address, originating caller identity, message content, and in some cases, the successful execution of prior policies. Most of the common XDoS attacks like entity expansion attacks can be filtered by adding specific policies at the XML firewall level. It is to be noted that this type of filtering has a significant effect on performance as complex policies need to be applied to the incoming XML messages. Moreover, newer attacks are constantly being invented in the growing field of Web services. Therefore, the filtering techniques need to keep up with the different types of attacks.

Location Hiding

In this type of prevention mechanism, the actual location is hidden from the end users, preventing the attack from taking place. An architecture built on this principle is called Secure Overlay Service (SOS) [97]. The goal of the architecture is to allow communication between a confirmed user and a target. The target select a subset of nodes N that participate in the SOS overlay to act as *forwarding proxies* The filter *only allows* packets whose source address matches the address of some overlay node *n* in N. It is assumed that the set of nodes that participate in the overlay is known to the public and to the attacker as well. Attackers in the network are interested in preventing traffic from reaching the target. By hiding the actual target, SOS reduces the effectiveness of DoS attacks. However, there are some concerns: Firstly, the SOS architecture may result in additional latency because of the multiple forwarding and routing that takes place. In some applications, this may be really critical. Secondly, the architecture assumes that the attack is coming from outside and does not concentrate on insider attack. A high level overview of the SOS architecture is provided in Fig. 6.7.

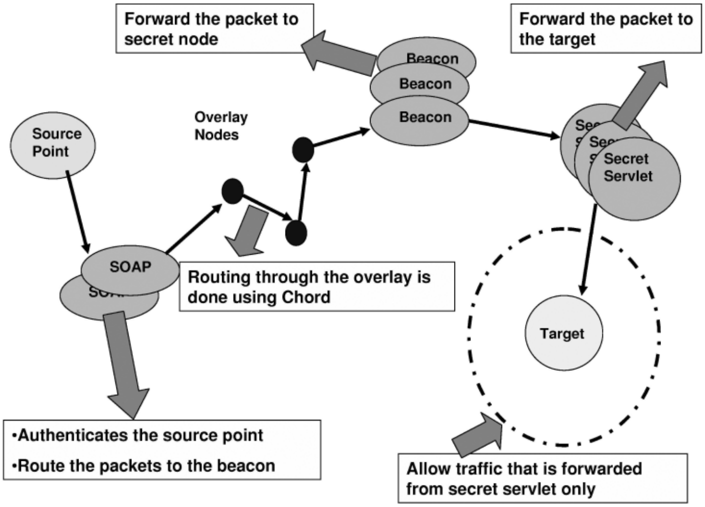

Forward the packet to secret node
Forward the packet to the target
Beacon
Beacon
Beacon
Source Point
Overlay Nodes
Secret Servlet
SOAP
Routing through the overlay is done using Chord
Target
•Authenticates the source point
•Route the packets to the beacon
Allow traffic that is forwarded from secret servlet only

Fig. 6.7. Location hiding through secret overlay service

Throttling

One proposed method to prevent servers from going down is to use max-min fair server-centric router throttles [98]. This method sets up routers that access a server with logic to adjust (throttle) incoming traffic to levels that will be safe for the server to process. This will prevent flood damage to servers. Additionally, this method can be extended to throttle DDoS attacking traffic versus legitimate user traffic for better results. This method is still in the experimental stage; however similar techniques to throttling are being implemented by network operators. The difficulty with implementing throttling is that it is still hard to decipher legitimate traffic from malicious traffic. In the process of throttling, legitimate traffic may sometimes be dropped or delayed and malicious traffic may be allowed to pass to the servers. One of the projects which use throttling as a means of mitigating DoS attacks is the D-WARD project from UCLA. D-WARD is a DDoS defense system deployed at source end network, which autonomously detects and defeats attacks originating from these networks. It includes observation and throttling components, which can be part of source router, or can be a separate unit to interact with the source router to obtain traffic statistics and install rate limiting rules. The observation component monitors two-way traffic at a flow granularity to detect the attack. Flow classification, connection classification, TCP normal traffic model, ICMP normal traffic model, UDP normal traffic model are used to differentiate the malicious flow and the legitimate flow. Once the attack flow is found, the misbehavior flow is under the control of rate limiting rules. D-WARD can detect some attacks at the source edge network and it attempts to determine outgoing attack traffic. But since there is no coordination among instances of agents, the detection may be error prone. Source end defense defense is a promising scheme that can be applied in the active defense system. However, it faces a lot of challenges such as detection sensitivity, agent coordination, and liability. When the defense system is deployed in the source end, there are fewer strong signals to indicate the attack than at the victim end, at which there are usually apparent signals such as high volume of network traffic. So a high sensitivity is essential for source end defending.

Intrusion Detection Systems (IDS)

It would not be right to categorize Intrusion Detection Systems (IDS) as just a preventive DoS measure. IDS systems [99, 100] basically consist of a set of detectors that detect attacks based on a set of policies and information. In principle, it works similar to alarm systems implemented in many buildings and apartments for protection against burglars. In [100], the authors have categorized IDS systems into two main categories: *anomaly detection systems* and *signature detection systems*. The former type of IDS systems, intrusion is detected based on abnormalities of system behavior. The detector forms an opinion based on the normal behavior of the system through a long term observed behavior and system policies. In signature detection system, an intrusion is detected based on a specific signature or a model. It is to be noted that the signature is based on long term information about the intrusion behavior.

Several grid based IDS systems have been conceived, designed, and implemented. Figure 6.8 shows the basic components of a grid based IDS system. Most of the grid based IDS systems consist of several components: A set of sensors which are able to monitor the state of the grid systems. The information supplied by the sensors are then collected and analyzed by IDS systems like SNORT [101]. The information is then logged through an interface to query the information, and suitable alarms and action mechanisms are then provided. Several grid based systems, described in the literature, are SANTA-G [102], which uses SNORT as the IDS system and R-GMA for querying the monitored information. More information about R-GMA is provided in Chap. 11. Another example of grid based IDS is GIDA [103] which also uses a similar structure. IDS on Oracle® 10G database is provided in [104]. IACID [105] from USC, provides a Grid based IDS system having separate network and host IDS systems.

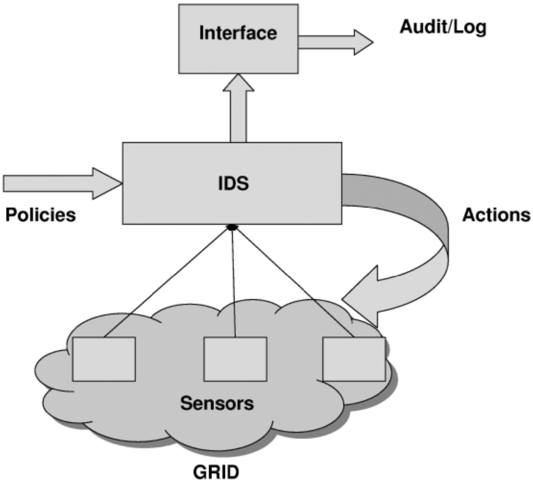


Fig. 6.8. Grid based IDS systems

6.2.5 Reactive DoS Countermeasures

Reactive techniques aim at identifying the attacker after the attack has been completed. This is an active area of research because the current identification techniques are totally manual, and may span over months. The current solutions can be broadly categorized into: (i) link testing, (ii) logging, (iii) ICMP traceback, and (iv) IP traceback.

Link Testing

This technique involves iteratively checking the upstream link until the source is reached. This type of identification technique assumes that the attack remains active after the completion of the trace. One type of link testing approach is called *input debugging,* where routers develop an attack signature based on some attack pattern. The victim informs the operator about the signature which then checks the packets, and iteratively carries out this process. This is employed in some routers now, though the process is time-consuming. Another suggested link testing is through *controlled flooding* [106]. In this type of technique, the victim floods all the links based on the assumption that packet drop taking place from an attacked link is much more than from any other link. This technique suffers from being a mode of DoS attack by itself.

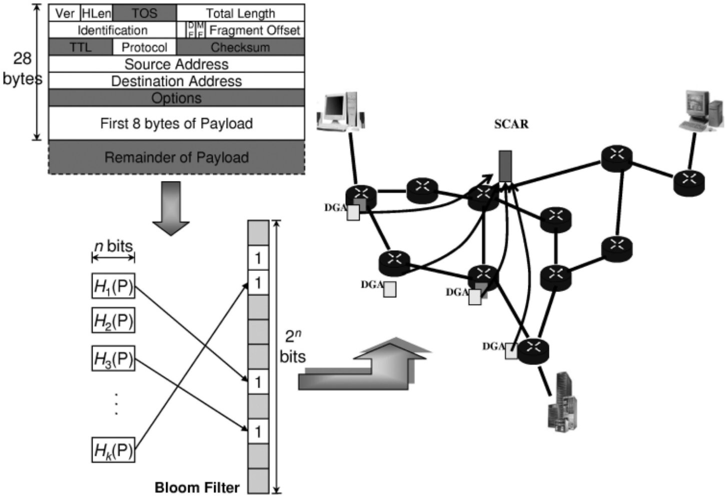


Fig. 6.9. Logging based traceback – SPIE architecture

Logging

A simple technique has been suggested in [107], where logging of data packets are done at key routers. Traceback is carried out by using data mining techniques. Another interesting work in this area is reported in [108], where the authors have presented a hash-based technique for IP traceback that generates audit trails for traffic within the network. The origin of packets can be traced back to the source based on the audit trails.

Figure 6.9 shows a high level description of the Source Path Isolation Engine (SPIE) architecture. Each router consists of a Data Generation Agent (DGA) which computes the hash of the packets and stores them in a Bloom Filter. The information is flushed after every time interval *t,* where *t* is a design parameter. As soon as the attack is detected, the SPIE Traceback Manager (STM) calculates the attack signature of the packet or packets used for the attack. It then contacts the centralized SPIE Collection and Reduction Agent (SCAR). SCAR polls DGA for the information stored, creates a local attack graph, and sends the information back to the STM. The STM then assembles the local graphs, plugs holes in the graph, and finally makes the traceback information. This technique suffers from scalability problem, as enormous resources are required to carry out logging based identification. Another negative, which is associated with all traceback schemes, is that they can traceback to a single attacker. However, most of the attacks are carried out using reflectors, in which case the traceback schemes can rarely be used.

ICMP Messages

In the Internet draft [109], the author has proposed a scalable technique where each router stores packet with a low probability. Whenever a packet is stored the router sends an ICMP traceback message towards the destination. When attacked, the destination can traceback to the source based on the router ICMP messages. This scheme has a problem as the ICMP messages can be used by an adversary to cause DoS attacks.

Packet Marking

One of the earliest efforts to identify the source of the packet through IP traceback was done in [110] through probabilistic marking of packets at each router. In this technique, a router marks any packet flowing through it with a very small probability. Getting sufficient number of packets (in case of DoS attacks), the destination can retrace the attack path. The process is described in Fig. 6.10. The scheme introduces a huge amount of overhead on the packets, which is reduced by node sampling where only one field is reserved for marking and the information get overwritten. The attack path, in this case, is retraced by relative number of packets marked at each router. This type of traceback is called *node sampling*. The authors also introduced a concept called *edge sampling* where in addition to the information of the node the distance to the node is also maintained. The schemes were further extended in [111], where the authors showed that using partial network information, the number of packets required to traceback can be substantially reduced.

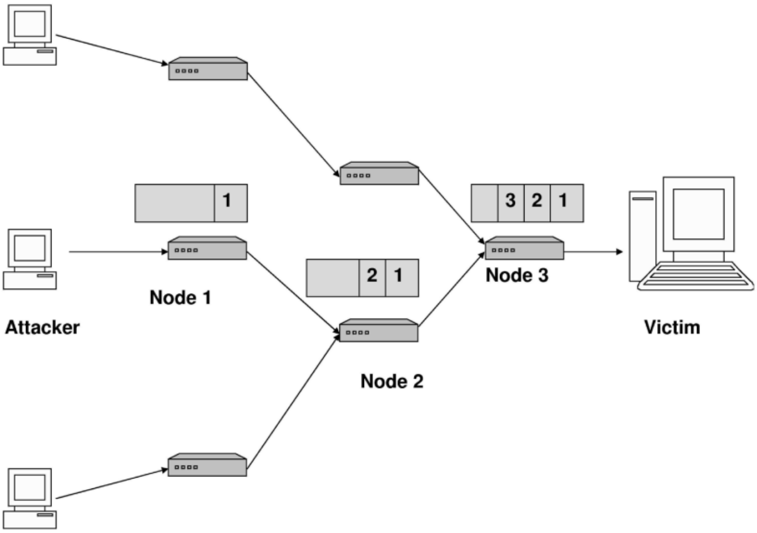


Fig. 6.10. Example of packet marking

6.2.6 Comparison between DoS Countermeasures

Table 6.1 provides a summary of the different techniques mentioned in the section. It is to be noted that DoS attacks cannot be mitigated by one solution alone and multiple solutions should be employed to improve the effectiveness. Among the different available solutions, the preventive solutions like application filtering, packet filtering, and intrusion detection are the only techniques that have been successfully implemented. However, most of these solutions have limited success and more research and development efforts are needed. The reactive solution space is sparser. Though several interesting research ideas like packet marking and link testing have been proposed, the implementations have not been carried out due to the complex nature of the analysis involved.

Table 6.1. Comparing the different DoS countermeasures

Solution	Type	Effective for	Strengths	Weaknesses
Packet Filtering	Preventive	Detection, Prevention	Different policies can be employed	Need to be updated
Application Filtering	Preventive	Detection, Prevention	Different application level policies can be used	Still more research is needed
Location Hiding	Preventive	Detection, Prevention	Prevents the attacker from knowing the victim	Introduce latency
Throttling	Preventive	Detection, Prevention	Able to reduce the effect of attacker	May result in performance drop
Intrusion Detection	Preventive/ Reactive	Detection, Prevention	Different policies can be employed. Goes beyond DoS.	Requires extensive system support.
Link Testing	Reactive	Identification	Identification through link testing	Huge bandwidth overhead, may be a tool for DoS attacks
Logging	Reactive	Prevention, Detection, Identification	Can identify using very small number of packets	Results in a huge storage overhead and is a slow process
ICMP	Reactive	Detection, Identification	Can be done using existing tools	Overhead in terms of messages, can also be a DoS tool
Packet Marking	Reactive	Identification	Less overhead, generally one field in the packet	Requires a large number of packets to make the identification

6.3 QoS Violation Attacks and Countermeasures

As grid computing moves towards enterprise level adoption, the issue of Quality of Service (QoS) assumes enormous importance. Grid computing, like the Internet, is typically best-effort in its performance. This best effort type of service may not be sufficient for enterprise users who demand tight QoS for their applications and jobs running on the grid infrastructure. The issue becomes especially important in cases where different types of services are expected from the grid infrastructure based on the amount of money the customers pay.

Developing QoS architecture for a grid system is a complex issue as it contains a heterogeneous mix of systems and infrastructure. Therefore if a grid system is to deliver a certain quality-of-service, each layer within the grid stack including network, host, and service, also has to deliver a certain level of QoS. An attacker targeting the QoS of the grid system can launch the attack by selectively attacking the different components of the grid infrastructure. For example, in some cases slowing the network traffic or creating network level congestion can significantly affect the QoS of the entire grid system. In this section, we will discuss the different QoS attacks and solutions that have been proposed in literature.

6.3.1 Different Types of QoS Violation Attacks

The QoS attacks can mainly be achieved in two ways: One way to achieve such an attack would be to spoof a legitimate user and create disruption by widespread SLA violation. These attacks are difficult to detect and prevent as the typical intrusion detection systems would not be able to detect such attacks. In this type of attack, the attacker gets into the grid system and carries out network level QoS attacks by dropping packets or delaying packets. Another type of attack can be carried out by “greedy” users wanting to get more out of the system. Since most of these attacks remain undetected by standard IDS systems they can be carried out surreptitiously.

Dropping Packets

In this type of attack, the attacker chooses to drop packets flowing through the grid networks. Packets can be dropped in a specific order or in a random manner. To carry out such an attack the adversary takes over a component within the grid infrastructure like a router through a virus or a worm. If TCP is used as a transport protocol then such selective dropping of packets results in TCP reducing its sending window as a result of which the network throughput decreases. Unlike DoS attacks, dropping of selective packets are very difficult to detect as packets get dropped naturally in any network. In many systems especially in multi-media systems UDP is used as a transfer protocol. Selective dropping of packets in those systems will result in lesser QoS received by the end systems. Similarly some adversaries may choose protocol or application specific packet dropping by selectively dropping the DNS packets or ICMP packets like "HELLO" packets which are used by many protocols for heart beat purposes. Solutions to these problems are mostly complex and protocol specific. However many monitoring techniques have been discussed in literature (discussed subsequently) which attempt to detect attacks of this nature.

Delaying Packets

In this type of attacks, the attackers don't drop packets; rather the packets are delayed to reduce the effective QoS delivered by the system. This can be achieved through routing table poisoning as mentioned before or by misrouting the packets flowing through the system. This type of attack generally increases the latency of packets resulting in higher end-to-end latency. Looping of packets may also be a result of this type of attack which may eventually lead to the dropping of packets. The solutions proposed are mostly adhoc and through monitoring of QoS parameters.

Resource Hacking

We mention a similar type of attack in Chap. 7, where resources of individual hosts are compromised. Similarly, network and infrastructure resources also can be compromised. This type of attack can be mitigated by provisioning and monitoring of resources in an efficient manner.

6.3.2 Existing Solutions

Let us now discuss the different solutions available to detect and mitigate SLA violations. The solutions can be of mainly two types: monitoring and auditing systems, and protocol specific solutions. As part of the monitoring and auditing discussions, we will mention the network monitoring mechanisms and grid auditing systems like GridBank. We will also briefly touch upon an interesting work in detecting packet dropping called WATCHERS in University of California at Davis.

SLA Violation Detection in Networking Infrastructure

In this type of mechanism, SLA violations in networks are detected by monitoring packets and measuring the delays and packet losses suffered by the packets [112 - 115]. Delay bound guarantees made by a provider network to user traffic flows are for the delays experienced by the flows between the ingress and egress routers of the provider domain. *Delay measurements* either use delay of real user traffic or injected traffic. The first approach is intrusive because encoding timestamps into the data packets would require changing the packets at the ingress and rewriting the original content at the egress after appropriate measurements. The second approach is nonintrusive in that one can inject probe packets with desired control information to enable an egress router to recognize such probes, perform measurements and delete the probes from the traffic stream. Packet loss guarantees made by a provider network to a user are for the packet losses experienced by its conforming traffic inside the provider domain. To compute the loss ratio, the number of packet drops, as well as the number of packets traversing the domain, is required. Loss ratio is defined as the ratio of the number of packet drops within the domain to the total number of packets passing through the domain. Core routers can detect the number of packets dropped, and edge routers can compute the number of packets traversing the domain. This loss measurement mechanism can be called the *core-assisted* scheme for loss measurement. An alternative mechanism is to use stripe-based probing to infer loss characteristics inside a domain. In stripe based mechanism, a series of packets or “stripes” are sent which do not introduce intermediate delays [116].

WATCHERS Project

The WATCHERS [117] from UC Davis was proposed to detect and react to routers that maliciously drop or misroute packets. WATCHERS is based on the “principle of packet flow conservation” *i.e.*, the number of incoming packets for a router, excluding those destined to it, should be the same as the number of outgoing packets, excluding those generated by it. In order to validate the conservation law, multiple decentralized counters are periodically and synchronously exchanged among neighbors of the target suspected router. Subsequently, each neighboring router runs a validation algorithm to diagnose the health condition of the target router. Furthermore, WATCHERS is robust against Byzantine faults. While WATCHERS offers theoretically an interesting way to deal with malicious packet dropping it cannot handle the packet dropping problems in today’s Internet effectively. First, the number of messages for counter value exchanges can be very large. Secondly, the “principle of packet flow conservation” does not hold “deterministically” for today’s Internet environment. For instance, an innocent router might drop packets for good reasons such as preventive congestion control or insufficient resources to keep all incoming packets. Though, WATCHERS as a solution may not be viable in an Internet environment, it may be a useful tool in a controlled grid system where the number of packets dropped is not as large as that in the Internet.

Grid Accounting Systems

Researchers in the grid community have started to realize the importance of QoS in grid computing systems. Several research projects like GridBank [118], from University of Melbourne and SweGrid [119] from Royal Institute of Technology (Sweden) have tried to address this issue through their accounting and auditing systems. The former is more of an accounting system where there are charging and payment modules. It also includes a service cost negotiation (e.g. $ per hour) which is carried out by the Grid Resource Broker (GRB). GRB negotiates service cost per time unit (e.g. $ per hour). GridBank issues GridCheques (similar to credentials) for the service consumers and grid resource meters gather resource specific usage information to be used for charging purposes. SweGrid system goes beyond being just an accounting system through SLA negotiation, monitoring, and management. The SLAs are negotiated through the negotiation phase and monitored using agents. Any SLA violation may result in renegotiation or moving the job to some other grid service provider.

6.4 Chapter Summary

The third component of the secure grid architecture is to protect the service offered by the grid system. The different service level threats are QoS violation, Denial-of-Service (DoS), and unauthorized access. It is to be noted that due to the limited deployment of grid systems, not many instances of these attacks are there in the grid systems. However, instances are there where DoS attacks have ripped the Internet. There is a need for grid researchers to study the effects of the DoS attacks and solutions in the grid computing perspective. DoS attacks can be performed in multiple different ways. The most popular is called the TCP SYN attack where attacker overflows the buffer of the grid system through millions of SYN packets. Other transport layer attacks like PUSH+ACK and ICMP are also very popular. At the network layer, DoS attack can be carried out through routing table poisoning, where bogus and maliciously intended packets corrupt the routing tables of the networks resulting in a DoS attacks which are extremely difficult to detect and prevent. At the application layer, DoS attacks can be carried out by poisoning and spoofing the DNS entries. Other application layer attacks include XML based DoS attacks where intentional loops are created in the XML documents which effectively creates a DoS attack.

Other varieties of attack include SQL injection, schema poisoning, replay attacks, and so on. The solutions can be broadly categorized into preventive and reactive. The former, like application filtering, intrusion detection, and other solutions, tries to prevent the DoS attacks from taking place by either generating alarms, or drop suspicious packets or request, or reroute to balance the load. The latter type of solution reacts to the attack and tries to identify the location of the attacker. Solutions like Logging, ICMP traceback, and packet marking fall into this category. In this chapter different attacks and solutions have been discussed and contrasted. Different QoS violation attacks and solutions have also been discussed in this chapter.

This brings us to the end of the architectural security issues. In Chap. 7 and 8, we will look at the infrastructure related issues starting with the host level security in the next chapter.

7 Host Level Security

7.1 Introduction

When I rented my apartment for the first time, I was worried about a few things. I was worried whether the new tenant would keep my apartment in a good condition. I was worried whether my assets in my apartment would be in order once the tenant starts living there. I was worried about the cleanliness of the tenant, whether he would pay electricity bills in time, and so many other things. In short, I was worried about the general health of the apartment and the resources that were in the apartment. The reason was that I was not sure how my apartment would be used by the new tenant, as I had no prior information about the tenant and his habits. Therefore, I made it a point to visit the apartment once every two months to make sure that everything was in order. I also checked that the electricity bills had been paid properly or not. I continued this for six months before feeling entirely confident about my tenant.

Donating hosts to be used as part of the grid is like renting an apartment. Given the background provided in the previous paragraph, I fully understand the anxiety of the users in donating their machines to grids. They are concerned due to the lack of confidence about the jobs that are running on their machines. Since the jobs are being submitted by faceless entities, the hosts can think that the adversaries are hidden among them. One way to tackle this problem would be to create a mechanism of trust, and provide access controls to prevent untrusted users from accessing the hosts.

Figure 7.1 shows the model described above. A trusted set of users is defined through the distribution of digital certification, passwords, keys etc. and then access control policies are defined to allow the trusted users to access the resources of the hosts. These mechanisms have been discussed in detail in Chap. 5.

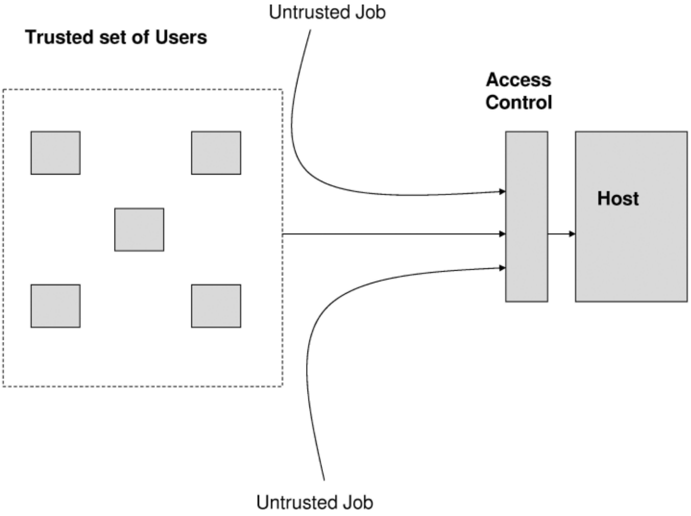


Fig. 7.1. A secure architecture based on trust and access control

However, does that mean that if a secure architecture is in place the concerns of the users will be nonexistent? In that case, in enterprises, the users would be donating their hosts for an enterprise-wide grid without any concern. On the contrary, in enterprises, where all the users are trusted, the users are unwilling to let their hosts/machines be used by jobs from other departments/groups. The concern, in those cases, may be less about the malicious users, rather about the bad jobs. Therefore, there is a need for mechanisms to be in place so that the users feel confident that their data, jobs, and machine would be safe in spite of bad jobs running on their systems.

Now, let us look at the different issues that any host would be concerned about when affiliating itself with the grid. Following are the issues of concerns for the host:

- The host will be concerned with its data being corrupted by outside jobs running on the system. We will call this issue as the issue of *data protection*.

- The host will also be concerned about its own jobs. The jobs may be starved by resource consuming outside jobs. We call this issue as the issue of *job starvation*.

Data protection issues are generally tackled through isolation where the local data and jobs are isolated from the outside jobs through different mechanisms like sandboxing, virtualization, etc. Job protection is tackled through a combination of techniques including priority reduction, advance reservations, etc. In the rest of the chapter, we will discuss these issues and the solutions in detail.

7.2 Data Protection Issue

The issue of data protection involves protecting the host data from outside jobs. A job from a malicious user can corrupt local data, crash the local jobs, and make the local system unusable. It is similar to a tenant using an apartment. The tenant can destroy the assets of the apartment, and make the apartment unusable for later use. One way to localize the damage would be to keep the part of the apartment rented out as only accessible to the tenant and keep the rest of the apartment inaccessible to the tenant. This is exactly what the *isolation* solution talks about. It keeps the outside jobs in the protected environment, so that even if the job is malicious, it remains confined to the isolated environment. This isolation can be achieved through several mechanisms:

- **Application Level Sandboxing:** This mechanism also known as Proof Carrying Code (PCC) enables the code provider to generate proofs of the safeness of the code and embed it inside the compiled code.
- **Virtualization:** Virtualization is a solution where the applications are run on isolated environment called Virtual Machines (VM).
- **Flexible Kernels:** These systems typical include kernels which can be extended in a flexible manner for better performance and isolation.
- **Sandboxing:** Perhaps the most popular of the isolation systems, these systems typically enforce isolation through interrupting system calls and loadable kernel modules.

7.2.1 Application Level Sandboxing

Application level sandboxing is a technique where the isolation and security capabilities are embedded in the application. Security features are hardwired into the application which can be verified before executing the application on a remote system. Cryptographic mechanisms can be used to determine whether a piece of code was produced by a trusted person or compiler. These concepts were used in the development of SPIN kernel [120]. Another seminal work done in the area of application level sandboxing was Proof Carrying Code (PCC) [121] developed by George Necula and team from CMU.

Proof Carrying Code (PCC) introduces the concepts of *code producer* and *code consumer*. In the former system, the code is produced and in the latter system, the code is executed. PCC is a mechanism by which a code consumer is convinced that the code produced by the code producer is not malicious is nature. To achieve that, the code producer is required to provide a safety proof which guarantees that the code conforms to a formally defined safety policy. The code consumer then validates the safety proof using a validator to ascertain the safeness of the code.

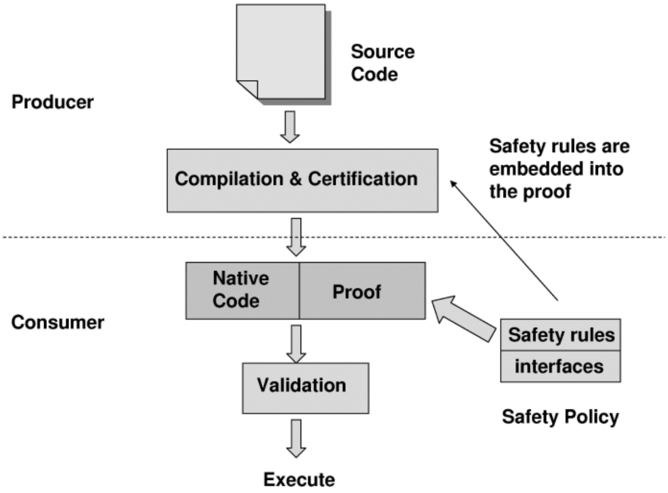


Fig. 7.2. Overview of the proof carrying code system

Figure 7.2 shows the overview of the PCC system. Specifically, PCC systems consist of the following components:

- **Safety Policy:** Safety policy is a set of policies defined by the code consumer and it is assumed to be known to all the code producers. The policies state precisely under what conditions the execution of foreign code is safe in its environment. It consists of two components: *safety rules* and *interface*. The safety rules indicate all the operations and preconditions that the code is authorized to perform. An example of such rules can be that a foreign code can only access certain portion of the data in the code consumer's system. The interface, on the other hand, includes standards methods or invariants by which the consumer and producer can interact without needing to exchange any information.
- **Certification:** In this stage, the code producer compiles the code based on the safety policies of the code consumer and generates a proof which guarantees that the source program adheres to the safety policies. This certificate is embedded in the native code and sent as a whole to the code consumer.
- **Validation:** Once the code consumer receives the foreign code along with the proof, it validates it against its safety policies. The validation is generally faster, as it is done at the native code level and generally the validation algorithms are straightforward. Once the proof is validated, the foreign code is accepted to be safe, and the native code is loaded for execution. The validation process can be further accelerated, if it can be performed offline and only once.

The PCC concept is quite powerful and provides a unique way of verifying the safeness of a particular code generated in a foreign machine. There are distinct advantages of using PCC in a wider scale. They are:

- The entire burden of formally proving the safety of the code is shifted to the code producer from the code recipient, whose job is to perform a fast, simple and easy-to-trust proof-checking process.
- No cryptographic mechanisms or third-party tool is required as the checking is done on the intrinsic properties of the code.

- PCC does the checking statically before the code execution, which not only saves time but also detects the potentially dangerous code early before execution.

However, there are several drawbacks which make PCC unsuitable for grid based applications.

- **Complexity:** Though the PCC concept is very powerful, the case studies used are simple and hence generation of the proofs is simple and feasible. However, for a complex system like the grid, the policies will be fairly complex and generation and validation of the proofs require a thorough analysis and experiments.
- **Heterogeneity:** Grid systems are typically heterogeneous in nature and therefore there may exist many different systems and code producer need to develop proofs for each and every system and embed them in the code. This becomes very expensive and infeasible to be used in real life systems. Added to this, there may be different policies for different systems within the grid which makes the generation, validation, and transfer of proofs more complex and infeasible.
- **Applications:** In real-life, especially in an enterprise, there are a large number of pre-compiled third party applications which are run on a grid based system. Therefore, PCC requires the application vendors to make their applications conform to the PCC requirements. Until that happens, PCC will remain a very powerful concept with limited implementation especially in the enterprises.

7.2.2 Virtualization

A typical data center today hosts different applications in different servers resulting in over-provisioning of resources and low utilization. Therefore, for some time there has been a move towards consolidation of servers to increase the overall utilization of the data centers. Research and development in the area of server consolidation has resulted in *virtualization* solutions in the server consolidation space. These solutions typically allow applications to run on self-contained environments called virtual machines (VM). It is possible to create different instances of VMs on individual servers, resulting in a better provisioning environment and higher overall utilization. Not only different instances of VMs can run but these instances can also host completely different operating systems. Therefore, virtualization techniques allow legacy systems to run on new systems seamlessly. In addition to these advantages, a by-product of virtualization is isolation. Therefore, virtualization techniques allow the creation of secure environments and can be used as an isolation solution. It is to be noted that the main goal of virtualization solutions is to provide higher resource utilization and server consolidation. The ability to provide secure and isolated environments come as a by-product. Therefore, there is a need to create flexible policies on the virtualized environment. Research is currently being carried out in this regard [122].

To provide virtualization, there is a need for a layer of software which provides the illusion of a real machine to multiple instances of virtual machines. This layer has been traditionally called Virtual Machine Monitor (VMM). There are also concepts called the *host operating system* and *guest operating system*. The former is the operating system or OS which hosts the VMM, and the latter is the operating system which is hosted on top of the VMM. It is also possible for the VMM to run directly on the hardware. In that case, host operating system is not required, and VMM will the minimal OS. There are three popular virtualization technologies: *hosted virtualization*, *para-virtualization,* and *shared kernel based virtualization techniques.*

- The **Hosted Virtualization** model is one where the VMM and the guest OS run on the user space of the host OS. The applications running on the host OS and the guest OS share the same user space. Generally, this model does not require any modification to the host OS. However, since there are multiple redirections, the performance of such a model suffers significantly. VMWare® GSX Server is an example of hosted virtualization system.
- The **Para-Virtualization** model is one where the operating systems are modified and recompiled so that the multiple redirections of the hosted model can be avoided. The performance of the para-virtualization based systems is comparatively better than the hosted virtualization based systems. Xen [123] and Virtuozzo® [124] are examples of para-virtualization systems.
- The **Shared Kernel** systems are those systems where the kernel is shared and the user space is partitioned to be used by different sets of applications. An example of shared kernel based virtualization systems is the Linux VServer [125].

Hosted Virtualization Model

The hosted virtualization model allows multiple guest Operating Systems (OS) to be run on the user space of the host OS. Figure 7.3 provides an overview of the hosted virtualization model. As shown on the figure, App1 and App2, and the Guest OS share the user space. Applications running in the guest OS are sandboxed within the guest OS. The applications within the guest OS contact the hardware being redirected through the virtualization layer, thereby reducing the performance of the overall systems.

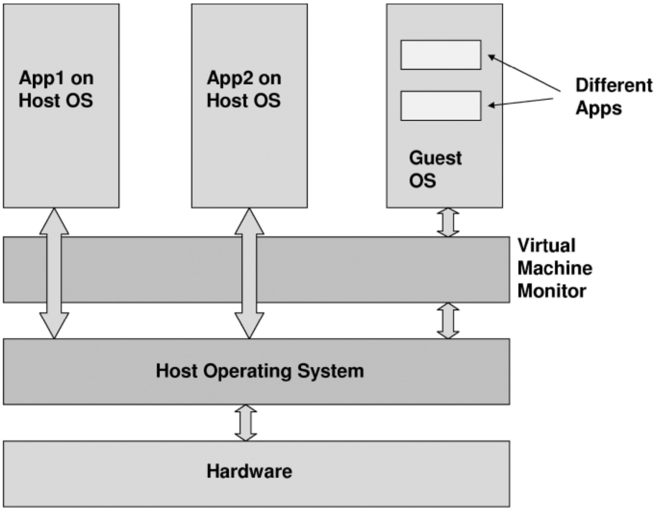


Fig. 7.3. Overview of a hosted virtualization model

VMWare® GSX Server – A Hosted Virtualization Solution: One of the most popular hosted virtualization solutions is VMWare's® GSX Server. Similar to the other hosted virtualization models, VMWare® GSX Server has a host operating system, and guest operating systems which run as applications on the host operating system. VMWare® Workstation's hosted architecture also includes a user-level application (VMApp), a device driver (VMDriver) for the host system, and a virtual machine monitor (VMM) that is created by VMDriver as it loads. Thereafter, an execution context can be of two types, native or virtual. The former context belongs to the host, and the latter belongs to the virtual machine. The VMDriver is responsible for switching this context. I/O initiated by a guest system is trapped in the VMM and forwarded to the VMApp, which executes in the host's context and performs the I/O using the normal system calls. VMware® uses numerous optimizations that reduce various virtualization overheads. In spite of the optimizations, the overheads introduced by the GSX Server can be significantly high, depending on the application. This led to the development of another model of virtualization called the para-virtualization model. However, it is to be noted that the hosted virtualization model in spite of the performance is extremely popular as it provides an easy solution to the virtualization problem.

Para-virtualization System

The hosted virtualization model mentioned before is one of the easiest mechanisms to achieve isolation through virtualization. However, the ease of managing hosted operating systems comes at a price. That price being performance. Since, in a hosted model there are multiple redirections of system calls, the performance suffers. In addition, the virtualization layer must manage all the underlying hardware structures like DMA controllers, page tables, I/O devices, and others, to provide a consistent view to all the operating systems hosted by the model. Whenever the virtualization layer context switches between the different OS images, it first needs to preserve the current states in the hardware structures, which can be used when the execution is resumed again. This managing of the structures puts a huge overhead on the performance of the hosted virtualization models. This overhead is sometimes quite significant.

To counter the problems of the hosted virtualization mentioned above, researchers have developed another virtualization model called the *para-virtualization* model. This model introduces the concept of the idealized hardware interface which completely abstracts the underlying hardware infrastructure. The virtualization layer or the *hypervisor* is embedded in the address space of each guest OS, so whenever the guest OS is required to update a hardware structure, it makes an API call to the hypervisor. Therefore, the hypervisor is able to keep track of all the happenings in the hardware data structures helping it to make optimal decision of updating the structures during context switching. It can also provide run-time specific information to the guest OS-es, enabling them to make better scheduling decisions. Based on these, the para-virtualization solutions have several distinct benefits:

- Performance of the para-virtualization solutions is significantly better than the hosted virtualization model. The hypervisor, since it is embedded with the guest operating systems, results in having fewer redirections. In addition, the information between the guest OS and the underlying hardware abstraction layer is exchanged much faster resulting in better managment of the guest OSes.

- Para-virtualization solutions provide significant benefits in terms of device drivers and device interfaces. Para-virtualization allows the virtualization of device drivers. It helps to provide resource CPU guarantees, and porting OS images across hardware.
- Para-virtualization offers better protection to the hypervisor compared to the hosted virtualization model. Since the hypervisor is run in a different protection domain compared to device drivers, it is protected from bugs and crashes of the device drivers.

In spite of the above benefits, the market share of the para-virtualization solutions are much less compared to the hosted virtualization solutions. Though the performance of para-virtualization solutions is better, they do not work across different platforms. The para-virtualization model requires the hypervisor to be embedded into the address space of the guest OS. Therefore, to achieve this, the guest OS need to be recompiled. This can be applied to the open source operating systems like Linux. However, for closed OS like Windows, para-virtualization solutions are currently not available. However, with chip vendors developing chips which support virtualization like Intel's® Virtualization Technology (VT) [126] and AMD's® Pacifica [127], applicability of the para-virtualization solutions will be greatly enhanced. The experts in the field of virtualization are confident that the para-virtualization techniques will be the future of virtualization.

Xen – A Para-Virtualization Solution: One of the most popular para-virtualization solutions is Xen which was initially developed from University of Cambridge. Currently, Xen is marketed by a company called XenSource® founded by the leader of of the Xen project, Dr. Ian Pratt. Market share of the Xen software is increasing in the virtualization space, where VMWare® has significant presence.

Xen presents all the benefits of the para-virtualization systems. It is fast and has a very low overhead on the overall performance of the system. This is achieved by storing the hardware states in memory and managing them efficiently. Figure 7.4 shows the high level architecture of the Xen para-virtualization system. Because of its open-source nature, and good performance, Xen is even witnessing commercial implementations. One of the biggest impediments in the wide-scale adoption of Xen and other para-virtualization solutions is the lack of virtualization capabilities of the popular IA-32 architecture. As mentioned in [128], the architecture has at least 17 instructions which make the architecture “non-virtualizable.” This leads to the compiling of kernels to make them aware of the Xen virtualization features. However, Intel® is coming up with Intel VT which will introduce virtualization features into the processors along with better management capabilities. The technology will push the adoption of Xen in a greater way.

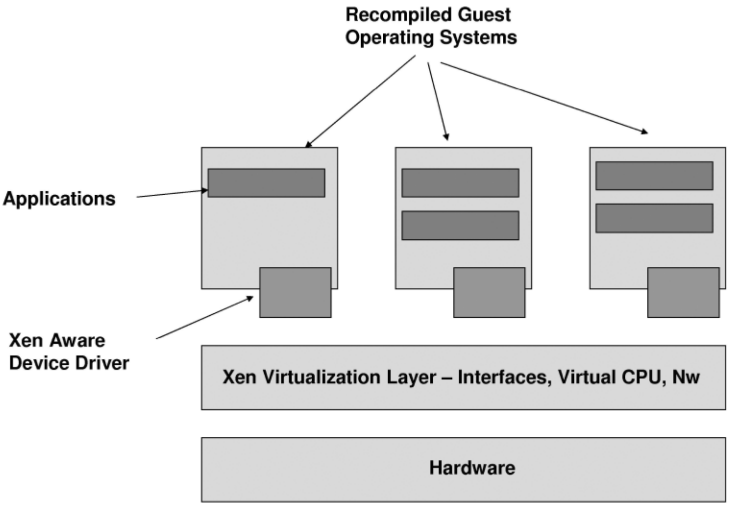

Recompiled Guest Operating Systems
Applications
Xen Aware Device Driver
Xen Virtualization Layer – Interfaces, Virtual CPU, Nw
Hardware

Fig. 7.4. Architecture of Xen

Nova – On Demand Virtual Execution Environments: One of the main challenges in having the virtualization solutions catering to isolation needs of grid systems is to have a policy manager which interacts with the virtualized environment. One of the key requirements of such a policy manager would be to create a virtualized execution environment on-demand based on the policies and incoming job requests. Nova [129] provides such a facility for the grid systems. The goals of Nova are (a) to reduce the time required to get a “working” virtual machine, (b) to ensure that the virtual machine allocated to the grid job has the necessary hardware and software resources, to perform the job, (c) to perform effective clean-up of the virtual machine once the job is complete, and (d) to ensure that the effect of a completed job does not spill over to another future job. Nova addresses the goals by creating, in advance, virtual machines with configurations that consume very little resources which are called "Tiny VM". Nova has been built on top of Xen system. The authors have shown that Nova is able to create virtual machines in the order of a few milliseconds. It is to be noted that the solution is a research in progress and significant effort is needed before it can be deployed effectively in enterprises.

Shared Kernel Systems

The third type of virtualization system is the *shared kernel system.* An example of such a system is the Linux VServer. The basic concept of the Linux VServer and other shared kernel systems is to divide the user space environment into distinctly separate units also called *Virtual Private Servers (VPS)*, in such a way that the processes within each VPS treat them as separate kernels. The shared kernel systems are very efficient compared to the other virtualization technologies. However, the flexibilities are greatly reduced as they tend to work on a single operating system, as all the applications use a shared kernel. For example, the Linux VServer runs exclusively on Linux.

Shared kernel systems are able to achieve the following benefits:

- **Higher Resource Utilization:** One of the biggest advantages of the shared kernel systems is the increase in resource utilization. By proper allocation of resources across the partitions and ability to share common resources across the partitions helps in increasing the utilization levels. The Linux VServer implementation uses token bucket implementation to achieve fairness across the different partitions or contexts.
- **Security:** The shared kernel based virtualization systems have high security as they can isolate the different contexts in an efficient and secure manner.
- **Low Overhead:** The overhead associated with the shared kernel systems is very low as they do not pass through multiple layers unlike the other virtualization systems. As mentioned in [130], the overhead of a Linux VServer system can be as low as 2%.

7.2.3 Flexible Kernel Systems

Some operating systems researchers argue that the performance, flexibility, and extensibility of operating systems are greatly limited by the design, where the interfaces and the implementations of OS abstractions such as inter-process communication and virtual memory are fixed. The need for flexibility and extensibility have been recognized as OS requirements by researchers even in the 1970s [131, 132]. In [131], the authors have advocated the design of open operating systems, where the system provides a variety of facilities, and the user may use, accept, reject, modify the facilities based on permissions and requirements. In many cases, one facility may become a component on which other facilities are built or developed, like files and disk pages. In that case, there is a need to identify smaller components, and make them accessible to the users and other larger components. The development of flexible kernel design provides a whole lot of interesting concepts and design. In this chapter, we will try to cover the three decades of conceptualization, design, and development efforts by identifying two representative solutions in this research area. The first project called Hydra [133] was developed in the 1970s, where the researchers have separated the policies and mechanisms of kernel, and used it as a guiding principle in kernel design. The second work, called *exokernels* [134] developed in MIT, looks at handling resource management at the application layer, thus providing faster transactions, and secure operations.

Project Hydra

One of the earliest examples of a flexible operating system was Hydra, designed in Carnegie Melon University (CMU), way back in 1974. It was an ambitious project designed to exploit the inherent potential of the multiprocessor computing system. The system was designed on C.mmp [135], a proprietary multiprocessor also designed and developed at CMU. The main purpose of the project was to demonstrate the possible development of a flexible operating system, where at the center there is an arbitrary set of facilities, which can be used to create the operating system based on the custom defined policies. Some of the considerations which guided the development of the Hydra system are as follows:

- **Multi-Processor Environment:** The main use case of the Hydra system was the development in a multi-processor environment.
- **Mechanism and Policy Separation:** Another consideration that led to the development of the Hydra project was to separate the

mechanisms and policies. This separation leads to a flexible system design as the system designer can make complex system decisions.

- **Protection:** Hydra develops protection based on capabilities. These capabilities can also be custom defined and are used to provide protection for all entities in the system.
- **Reliability:** The reliability of the system is guaranteed by the C.mmp hardware through the redundancy of critical hardware resources.

As mentioned earlier, the protection of the Hydra system is provided through the mapping of policies and mechanisms. Hydra system develops the protection mechanism by building on top of the underlying core facilities guided by the custom-defined policies. The protection mechanisms are defined through the interaction between *objects*, *capabilities*, and *procedures*. Objects refer to abstract reference of the arbitrary set of resources, capabilities are references to objects, and procedures are the abstract notion of any arbitrary set of operations. Each capability includes information detailing operations that can be performed on the object referenced by the capability. Whenever an operation is attempted on an object, the requestor supplies the capability referencing that object. The kernel examines the rights list of the requestor and prevents the operation when the rights failure occurs. The right has been defined as the set of permissions granting the requestor to supply capability as a parameter to any procedure. Based on the rights, some operations can be denied access to certain objects based on exclusive “kernel” rights.

The Hydra project may not be useful for immediate deployment; however, the concept of separating policies from mechanisms is powerful and has been applied in security systems in different contexts. The next system that we will discuss in this chapter is called an exokernel implemented as Aegis system in MIT. Exokernels extends the Hydra concept by eliminating mechanisms whenever possible through the application level policy handling, which offers high performance and flexibility.

Exokernels

In the mid 1990s, researchers in MIT came up with the design of a flexible operating system architecture, where the management of physical resources is handled at the application layer. The lack of flexibility of the existing operating system architectures to incorporate domain-specific optimizations and implementations motivated the development of the exokernel architecture. Another point that motivated the design is the need to provide more flexibility to the application builders, as the abstraction development methodology followed in traditional operating systems limited the application development to a great extent. The researchers in MIT attempted to solve these problems through application level resource management. The flexible operating system, thus designed, implements the traditional kernel level functionalities like Inter-Process Communication (IPC), Virtual Memory (VM), etc. at the application layer. In this architecture, the exokernel is a minimal kernel which securely multiplexes the underlying hardware resources. Library operating systems, working on top of the exokernel interfaces, implement higher level abstractions. The applications select libraries, or implement their own libraries by simply relinking the application interfaces. Different components of the exokernel system are illustrated in Fig. 7.5. As shown in the figure, the exokernel architecture securely multiplexes and exposes the hardware through low level primitives. The library operating system uses the low level primitives to implement the higher level abstractions, and finally the applications uses these to manage the resources according to need.

The different components of the exokernel systems are:

- **Managing Policies:** The task of managing policies is handled by the library operating system. The policies identify the applications, the share of resources, the way to arbitrate the competing applications, and so on. Different security policies can be embedded into the general policy definitions, as exokernels allow the extensibility in the systems.
- **Secure Bindings:** Another important task of the exokernel system is to securely multiplex resources, and provide protection to applications which have an existing trust relationship. The protection mechanism in secure bindings is created by separating the authorization from the actual use of the resources. This improves the performance of the system to a great extent. The protection checks are carried out by the kernel as simple processes, which are done quickly. The authorizations are carried out at bind time only, allowing the separation of management from protection.
- **Secure Multiplexing:** Secure multiplexing of physical memory is implemented in the exokernels using capabilities which can be authenticated. Whenever a library operating system allocates physical memory page, the exokernel creates a secure binding

for the page recording the owner and the permissions. The owner will then have the rights to access the page based on the permissions. In addition to multiplexing at the physical memory level, there is a need for multiplexing at the network packet/message level. The message multiplexing at software can be provided through packet filtering mechanisms.

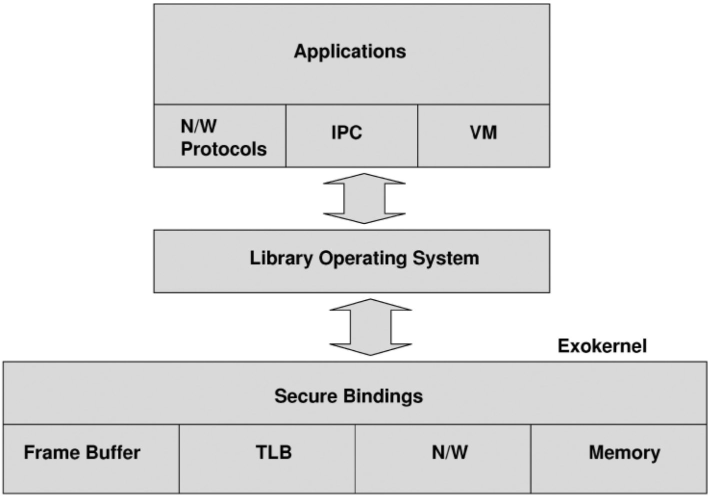


Fig. 7.5. High level view of an exokernel

Researchers have shown that the exokernel approach is able to attain a very high performance. Experiments show that the Aegis (an implementation of the exokernel architecture) is much more efficient that any other traditional operating system. Though the exokernel architecture does not include security benefits directly, there are security benefits which are attained as a byproduct. The benefits are:

- Separation of management and protection guarantees a more flexible and scalable mechanism of providing security. Different security policies can be incorporated easily.
- High performance of the overall system implies that the overhead associated with security is considerably reduced. Therefore, sys-

tem administrators can go for more complicated policies and security mechanisms without sacrificing performance to a great extent.

In spite of the advantages, the exokernel approach is still limited to laboratory implementation and has not seen an industry scale implementation. More research and development efforts need to be undertaken in this direction to make the concept of exokernel a commercially viable solution.

7.2.4 Sandboxing

One of the most popular techniques to achieve isolation is called sandboxing where applications are run in sandboxes to provide isolation and resource accessibility. It is to be noted that many of the solutions mentioned before in this chapter like virtualization has similar features. However, while the main motivation of virtualization solutions are server consolidation and improving resource utilization, sandboxing solutions were primarily designed with isolation in mind. Sandboxing solutions developed over the years can be broadly divided into three main types: through user level monitoring or system call trapping, through loadable kernel modules, and through the creation of user level virtual machines.

User Level Monitoring – Janus

Janus [136] is a user level sandbox developed as part of the research project in University of California, Berkeley. The motivation behind the Janus project was to transparently protect a large legacy system from an untrusted pre-existing application. Therefore, Janus makes an assumption that it does not have any control over the application or the environment on which the application is running. In other words, it cannot get any assistance from the applications, and should make use of the tools provided by the operating system to monitor and provide fine-grained control on the task execution.

Figure 7.6 shows a high level overview of the Janus architecture. At the lowest layer lie the tracing primitives available with the different operating systems like ptrace for Linux and proc for Sun® Solaris. The authors have mentioned that the ptrace primitive had to be modified to accommodate the different features like the abortion of system processes before execution, fine grained control of system process tracing, and so on. The authors called the tracing mechanism ptrace++ indicating that it is an extension of the already available ptrace functionality. Another important component of the Janus architecture is the user adaptation layer which provides a simple interface to the tracing primitives provided by the underlying operating systems.

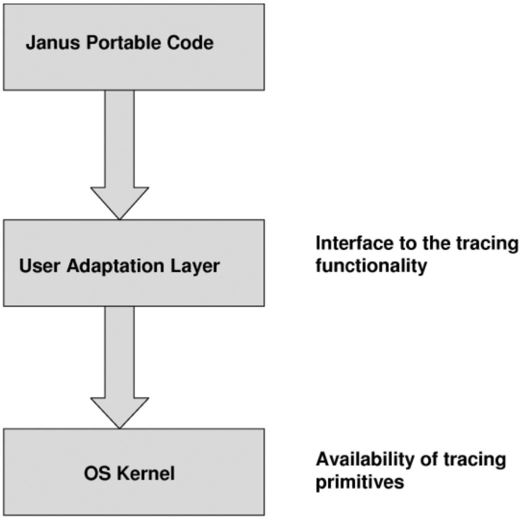

Janus Portable Code
User Adaptation Layer
Interface to the tracing functionality
OS Kernel
Availability of tracing primitives

Fig. 7.6. High level overview of the Janus architecture

Looking at the sandboxing systems based on system level monitoring and system call tracking, a few advantages come to the fore. The predominant advantage of such systems is the ability to provide fine-grained and *flexible sandboxing policies*. Most of these systems allow the system users to provide system specific and business specific policies. The ability to create flexible policies makes these systems distinctly advantageous over the virtualization systems. However, the enhanced flexibility takes its toll on the overall performance of the system. Monitoring the process and system parameters and trapping system calls based on the policies have a significant performance overhead. Another disadvantage of these systems is that they do not provide isolations under all scenarios as the entire context may not be available at the interception point. Sometimes there may be a need to replicate the original system call's implementation because pre and post processing may not suffice. These problems assume significant proportions in closed systems. Taking all these considerations into account, virtualizations provide more robust solutions in spite of their inflexibility in policy definitions.

Loadable Kernel Module – Remus

As mentioned earlier, another mechanism of sandboxing is through the monitor placed at the kernel. This can be achieved without a need to change the kernel source code or recompilation of the kernel, through a loadable kernel module. An example of such a system is Remus [137], developed by researchers in Italy.

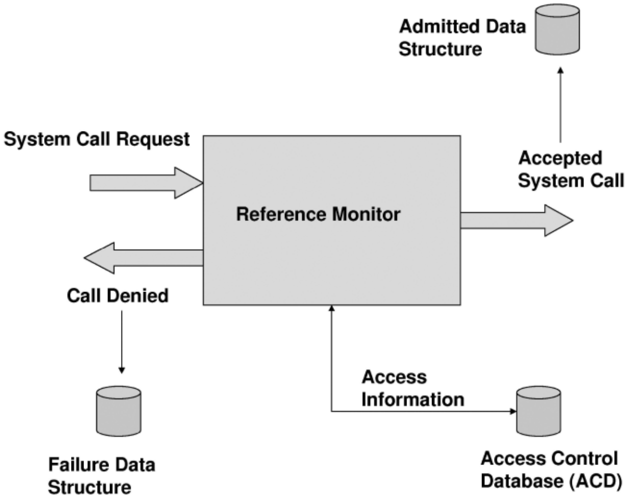


Fig. 7.7. Overview of Remus

The basis of the Remus system is the detailed analysis of the system calls for the UNIX operating system. The analysis includes various threats like penetrating the system with full control, and denial-of-service attacks. The Remus system (Reference Monitor for the UNIX systems) monitors the system calls that may be used for performing malicious activities. Remus intercepts the system calls and allows the system calls to execute only in the case where the invoking process and the value of the arguments comply with the rules kept in the Access Control Database (ACD) within the kernel.

Figure 7.7 shows an overview of the Remus system. The heart of the system is the reference monitor which monitors and controls the system calls and allows only the permissible calls to be executed on the system. The reference monitor consists of two main functions: the *reference function* and the *authorization function*. The reference function is used to make decisions whether to allow or disallow a system call based on the information kept in the Access Control Database (ACD). The ACD consists of rules or conditions that control the system calls and by specific system calls or values of their arguments. Authorization functions, on the other hand, allow the administrators to delete, add, and modify rules and entries in the ACD. In addition to the two sets of functions, a couple of data structures are used mainly for auditing purposes. As shown in the figure, the admitted data structures contain the list of system calls admitted into the system, and the failure data structure contains the list of system calls which have been rejected by the reference function based on the rules stored in ACD.

If loadable kernel modules are compared with the system level monitoring techniques, the former systems have reduced flexibility as the policies have to be embedded into the kernels as loadable modules. Therefore, any change in system policies would require the new policies to be loaded into the kernel which reduces the flexibility of the system. However, the performance of these systems is better than the traditional system level monitoring systems. The robustness of the systems based on the loadable kernel modules is better than the other sandboxing systems.

Virtual Machines – Entropia VM

Perhaps the most common mechanism of providing sandboxing is through the Virtual Machine (VM) technique. This is the technique used in Java Virtual Machines (JVM). A similar technique used in grid is the Entropia approach, called the *Entropia Virtual Machine (EVM)* [138]. EVM has been specifically designed to cater to the desktop grid environment, where there are a large number of desktop clients on which the grid jobs run, in addition to the Entropia server. EVM is part of Entropia Desktop Distributed Computing Grid (DCGrid) environment.

The different layers of the DCGrid are: Physical Node Management (PNM), Job Management (JM), Resource Scheduling (RS), and Entropia Virtual Machine (EVM). PNM is concerned about the physical node in the DCGrid environment. In addition to collecting a huge amount of information regarding CPU utilization, memory, system health, and so on, it also takes care of the reliability issues. The information captured by PNM are then used by the resource scheduler to provide better scheduling. The second layer, JM is responsible for breaking the jobs up into possible smaller subjobs and hands them over to the RS layer. The third layer or RS, distributes the jobs to different execution machines. The last layer or the EVM is responsible for starting the execution of the jobs on the execution host, monitoring the jobs, and provides security for the host by mediating the job's interaction with the host Operating System. We will discuss the different components of EVM. For details regarding the other layers, the readers can refer to [138]. A high level view of the EVM system is provided in Fig. 7.8.

EVM consists of two components: the *desktop controller* and the *sandbox execution layer*. The desktop controller is responsible for launching the processes to run the subjob, and monitoring the running of the subjob on the host desktop. The controller monitors CPU, memory, I/O usage, and other system and process specific information. The sandbox execution layer, on the other hand, provides desktop security through sandboxing and the mechanisms to interface with the desktop controller. The different components which facilitate the working of the sandbox execution layer are:

- **Application Wrapping Technique:** After a user submits an application or job to be executed on the DCGrid, the application is automatically wrapped inside an Entropia Virtual Machine through binary modification. It is achieved through the modification of the import table of the binary, which forces Windows to execute the customized Entropia dll to be executed. This allows the virtualization of address spaces for safe execution of the binaries. Before execution, the binaries are validated through checksums provided with each binary, to ensure the integrity of the binary.
- **Interception Technique:** The interception is the heart of the EVM sandboxing. It is similar to the Virtual Machine Monitor (VMM) described in Sect. 7.2. The interceptor is installed as a Windows Device Driver which intercepts the calls made to specific hardware resources. This technique along with the application wrapping technique provides the host security in DCGrid.

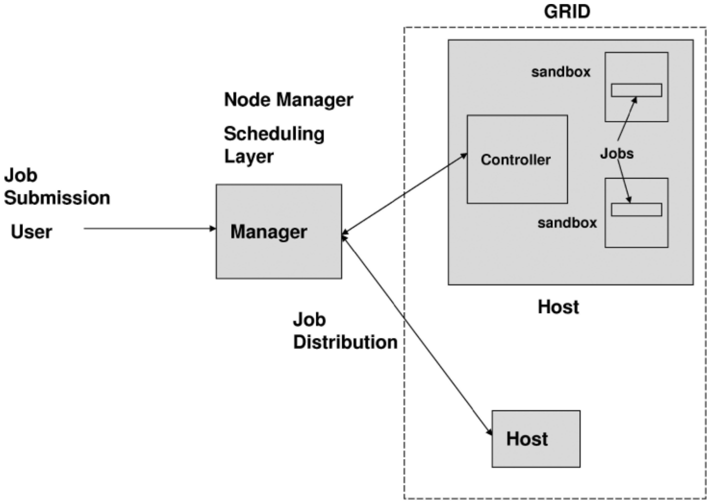


Fig. 7.8. Overview of the Entropia distributed computing grid environment

Paper [138] also talks about the performance implication of such a technique. Based on the evaluations the authors have concluded that a system has an impact of around 6%. Comparing the VM based sandboxing systems with others we find that the flexibility of these systems is greatly reduced because they are closely aligned to specific applications or operating system. For example the Java Virtual Machine or JVM can only be used in a Java based system. The policy level flexibility, however, is strong as user and business level policies can be attached, modified without significant burden unlike the loadable kernel based sandboxes. The performance of the VM based systems are comparable or better than the system level monitoring based sandboxing systems. The robustness of these systems is generally high because they are tuned to a particular category of applications, operating systems, etc.

7.3 Job Starvation Issue

Job starvation refers to a scenario where jobs originating locally are deprived of resources by alien jobs scheduled on the host as part of the grid system. This type of scenario becomes critical in grid systems due to the resource consuming nature of the grid jobs. Therefore, situations may arise where an alien grid job takes up a huge amount of grid resource resulting in a resource starvation for the local jobs. Different solutions pertaining to this are: *advanced reservations* and *priority reduction*. The former solution tries to tackle the problem by pre-allocating a certain amount of resources to the external grid jobs so that the local jobs are never starved of resources. Policies can be enforced in this regard. The latter solution is more of an ad hoc one, where the priorities of the jobs are reduced after a certain amount of time to prevent the starvation. In this chapter, we will briefly talk about the two techniques as scheduling is a discipline by itself and mostly would be outside the scope of the book. Interested readers may look at [139, 140] for more detailed insights into different scheduling policies, mechanisms, and their implementations.

7.3.1 Advanced Reservation Techniques

Under the advanced reservation system [141, 142], a user requests for a set of resources (can be CPU, memory, disk space, etc.) for a specified amount of time for the set of jobs to be run. The resources are booked based on the availability, security, QoS and other metrics. Once the resources are booked, the resource providers honor the contract and have every right to terminate the job once the contract expires. These techniques require schedulers to work hand-in-hand with the resources/hosts providing service to the end users. Under the advanced reservation schemes, users request the grid schedulers/meta-schedulers for resources. The schedulers talk to the different hosts for resources and based on the gathered information reserves resources. The host then agrees to grant access to certain resources and honors the contract. Advanced reservation techniques provide flexibility to the users to set aside some resources for their local use in case the resources are needed. One of the most extensive works done in this area is by researchers from University of Melbourne who have come up with a resource management and scheduling framework called Nimrod-G [143]. It is to be noted that virtualization solution mentioned in the previous section can be used as implementation techniques for enforcing reservations. Since each virtualized compartments can have specific amount of resources available to it, external grid jobs may be run on one of the compartments to provide reservation guarantees in addition to isolations. It is to be noted that an on-demand virtual execution environment on the lines of Nova will be essential to achieve this.

Based on the above discussions, the advanced reservation techniques provide a flexible mechanism to provide resource to the grid from the resource owner or host point of view. However, there are some concerns:

- Advanced reservation schemes require the users to have detailed knowledge about the resource needed by the submitted jobs, which may not be possible all the time.
- The grid schedulers need to implement advanced reservation.
- Though most of schedulers support advanced reservations, the integration of resource and scheduler through advanced reservation is a research problem.

7.3.2 Priority Reduction Techniques

Under these types of techniques, the priorities of the jobs are manipulated to reduce the possibility of starvation. Different techniques have been developed. One such technique called *local priority reduction* [5] reduces the priority of all jobs which are not local to the system. This ensures that the local jobs are never starved of resources. This technique has been employed in the United Devices® GridMP software suite. Other innovative technique have also been developed which tries to reduce the possibility of starvation in the whole system. In the Sun® Grid engine [142] a flexible priority scheme called RRDP has been developed where the priority of the job is determined by the normalized value as a combination of per resource weighting factor, deadline weighting factor, waiting time weighting factor in combination with host or resource policies.

These techniques can work as an ad hoc mechanism to prevent job starvation. These have their advantages as they can be implemented easily and mostly requires changes at the host or the resource owner end. However, the disadvantages are:

- These may result in lower QoS, if QoS is a concern to the users of the grid.

- They may result in unpredictable behavior unless integrated with the advanced reservation schemes.

7.4 Chapter Summary

When a host or a machine enrolls itself to the grid system it is concerned about the reliability of the jobs that would be running. The host level issues can be broadly categorized into data protection issues and job starvation issues. While the former is concerned about protecting the host data from outside jobs, the latter is concerned about protecting the local jobs from starvation. Most of the data protection solutions use isolation as a mechanism to ensure that the local system remains unaffected by the external jobs. Different types of solutions existing are: application layer sandboxing, virtualization, flexible kernels, and sandboxing. As a result, the solution remains only of academic interest without having any commercially viable implementation. On the other hand, the second category of solutions *viz.* the virtualization solutions provides efficient isolation. However, some of the virtualization solutions like the hosted virtualization model come with a performance overhead which for some applications may be significant. The para-virtualization solutions the Xen provide very good performance. However, most of these solutions are available for open operating systems like Linux and currently not available for closed systems like Windows. However, advances in the field of processor level support for virtualization auger well for the para-virtualization systems. There is another point of concern before virtualization systems become default solutions for isolation needs. There is a need for development of policy management mechanisms on the virtualization systems. The third category of solutions is the flexible kernel systems. These are mainly research solutions which require more research and development effort before they can become mainstream. Finally, the sandboxing solutions are there which cater mainly to the sandboxing needs of the users. These solutions are mostly through the monitoring of process and system parameters and trapping system calls. Most of these solutions are flexible in terms of policy management. However, the system level monitoring solutions have performance overhead. The better performance of the loadable kernel modules is offset by the lack of flexibility in terms of policy management. Table 7.1 summarizes the different data protection solutions and related issues.

In the next chapter, we will look at another important component of the grid infrastructure, *viz.* the network.

Table 7.1. Data protection solutions

Solutions	Type	Example	Policy Flexibility	Overhead	Robustness	Comments
Application Level Sandboxing	Same	Proof Carrying Code (PCC)	Low	Low at the recipient	High	Low flexibility, requires vendor support
Hosted Virtualization	Virtualization	VMWare ® GSX Server	Need to be built	Medium to High	High	Relatively High performance overhead
Para-virtualization		Xen	Need to be built	Low	High	Only available in open systems like Linux. A very promising technology.
Shared kernels		VServer	Need to be built	Low to Medium	High	Only available in Linux systems
Flexible Kernels	Same	Hydra, exokernels	Low	Can be tuned as per need	Generally high, can also be tuned	Good concept, requires significant research
System Monitoring	Sandboxing	Janus	High	Medium to High	Medium	High flexibility, low performance
Loadable Kernels		Remus	Low to Medium	Low to Medium	Medium to High	Low on policy flexibility
VM Based		Entropia VM	High	Low to Medium	Medium	Generally for a specific system like Java

8 Grid Network Security

8.1 Introduction

As part of our discussions on grid security, we have talked about security issues pertaining to the different components of the grid stack. However, till now we have ignored the element which cradles the grid infrastructure *viz.* the grid network. In course of this chapter we will discuss the several network security related issues which are relevant in the context of grid computing.

As a discipline, network security is becoming one of the most important areas of research, especially in the context of the Internet [144]. Initially, the focus of the research community had been solely on the performance issues of the Internet. In the last few years, several Internet level security attacks and vulnerabilities has resulted in a surge in the activities in this field. Grid computing, being a distributed system, naturally requires networking infrastructure for its functioning. Therefore, a thorough understanding of the network security related issues is important. In addition, grid adds complexities in terms of heterogeneity and high speed interconnects which add complexities in terms of management and integration with the grid system. In this chapter, we will discuss a few of the issues which are important from the grid perspective. Some of the issues, like secure multicasting and secure sensor grids may not be applicable immediately as the research in those fields are also maturing. However, taking a holistic view, we included the different areas which would be relevant to grid systems.

8.1.1 Grid Network Security Issues

Let us now look at the different issues pertaining to grid networks. The network security assumes importance in grid computing context mainly because of two reasons. Firstly, due to the typical requirements of the grid system, *viz.* the heterogeneity and the high speed networks, several new requirements arise which need to be tackled. Grid network access control and high speed network issues fall into this category. Secondly, there are issues which are challenging security issues in the area of generic networking. These issues need to be tackled due to the distributed nature of the grid systems, issues like secure routing, multicasting, sensor grids fall into this category. All these issues will be discussed in detail in the course of this chapter.

Network Access Control and Isolation

One of the immediate requirements for grid networks is the access control and isolation of the traffic flowing through the network. One of the most important technologies for access control is through firewalls where suspicious traffic is controlled and resources are protected. One of the immediate needs is the integration of grid technologies with firewalls and existing Virtual Private Networks (VPN) technologies. Sections 8.2 and 8.3 provide information about integration of grid with these two technologies.

Secure Routing

Routing is one of the most important components of any networking infrastructure. Routing mechanisms and protocols are designed to route packets through the network in a resource efficient manner. They also take care of network failure, network level congestion, etc. Researchers have looked into the issue designing secure routing protocols which are also relevant in the grid computing context. We look at the secure routing issues in Sect. 8.4.

Secure Multicasting

The main purpose of a grid system is to allow sharing of resources across the networking infrastructure. Multicasting is a technology which allows sending messages to a select set of users and has a direct relevance to grid information dissemination. Multicasting being an efficient mechanism of group communication further adds its value to the grid system. However, the technology does have some security issues where significant research has been carried out in the last few years. In this chapter, we highlight the importance of multicasting and discuss a few research issues and solutions which need to be taken into consideration for designing grid networks. Secure multicasting is dealt with in Sect. 8.5.

Secure Wireless and Sensor Grid

One of the main attractions of grid computing is it pervasive nature. Applications are being developed which brings wireless and sensors networks into the purview of grid systems resulting in a plethora of security issues which need to be analyzed. In this chapter, we also look at the different security issues in sensor networks and promising solutions which can be adopted. Issues related to wireless and sensor networks are looked into in Sect. 8.6.

Security in High Performance Grid Networks

Grid computing has seen the adoption of very high speed interconnects like Infiniband, 10-Gigabit Ethernet and others. Each of these technologies presents a huge opportunity of expanding the grid horizon by targeting high-throughput and high-performance applications. However, there is a need to understand the security pitfalls and vulnerabilities that exist. In this chapter (Sect. 8.7), we highlight some of the work carried out in this area.

8.2 Firewalls

As the name suggests, “firewalls” prevent intruders from getting access to the network by creating a wall or a hindrance to the intruder’s traffic. The concept is similar to the walls built outside the castles or cities in medieval times. Examples of such walls can be found in many European and Asian cities. As mentioned in [145], the term firewall was used as early as in 1764 by T. Lightoler, to describe walls which separated parts of the building most likely to have fire (for example, kitchen) from the rest of the building. The concepts of the network firewalls are also similar. They are meant to protect the network from the rest of the world by looking at the traffic passing through them and making a decision of whether to allow the traffic or not. Figure 8.1 illustrates a firewall in action. It shows that out of the two types of traffic, the firewall allows one to enter into network based on certain set of policies.

We had a detailed discussion about different access controllers in Chap. 5. Readers should have guessed the similarities between the two. Rather firewalls are a special type of access controller, where the resource they are controlling the access to is the network. The main purposes for using the firewalls are:

- **Security:** The most important reason why most companies have firewalls installed is to ensure security in the network. Many companies do port level or packet level filtering to prevent the network from security attacks.
- **Enforce Policies:** Firewalls are one of the easiest means to enforce policies. For example, a company may not allow employees from accessing FTP servers. In that case, the port 21 can be blocked as a company policy. Most of the modern firewalls allow more complex policies to be employed, where evaluation of network packets not only happens at the network or transport level, but also at the application layer.
- **Auditing:** Another important reason why firewalls have become ubiquitous as they can monitor traffic and provide audit trails. These become very good source of information in case of security breach.

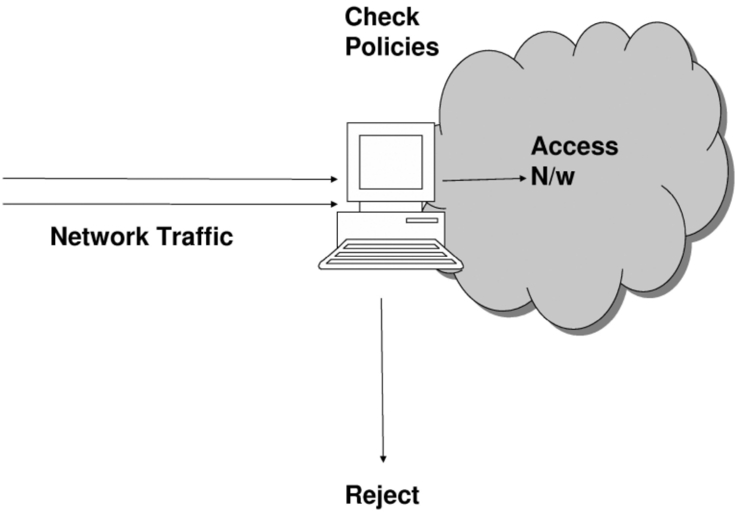


Fig. 8.1. A firewall in action

8.2.1 Different Types of Firewalls

As mentioned in [146], firewalls can be classified into four generations: static packet firewalls, circuit-level firewalls, application level firewalls, and dynamic packet filters.

Static Packet Firewalls

The *static packet firewall* is the first generation firewall technology which works at the network and transport layer. A set of simple rules or policies can be applied based on the port from which the packet is coming, the IP address of the packet, or similar information. The rules are simple and based on denying or permitting the packets. Though these firewalls are simple to implement, they have several disadvantages. Firstly, the rules exclude application level or state information which restricts their applicability. Secondly, they assume that the adversary is outside the network and cannot prevent internal attacks.

Circuit Level Firewalls

A *circuit level firewall* is a second-generation firewall technology that uses state information to validate the packet. It is able to determine whether the packet is a connection request packet or a data packet belonging to a particular connection. Several transport layer protocols like TCP follow a three-way handshaking mechanism for session or connection establishment. For the purpose of session validation, a circuit level firewall is able to examine each connection setup to ensure that a legitimate handshake has taken place for session establishment. In addition, data packets are not forwarded until the handshake is complete. The firewall maintains a table of valid connections and allows network packets containing data to pass through when network packet information matches an entry in the virtual circuit table. Similar to the static packet firewall, this type of firewall also cannot restrict packets based on higher layer information.

Application Level Firewalls

An *application level firewall* is a third-generation firewall technology that can evaluate network packets for valid data even at the application layer before allowing a connection. It examines the data in all network packets at the application layer and maintains complete connection state and sequencing information. In addition, an application layer firewall can validate other security items that only appear within the application layer data, such as user passwords and service requests. This type of firewall introduces a significant amount of overhead as every packet needs to be evaluated across the networking stack.

Dynamic Packet Filtering

A dynamic packet filter firewall is a fourth-generation firewall technology that allows modification of the security rule base dynamically. This type of technology is most useful for providing limited support for different connectionless protocols like the UDP transport protocol. This type of firewall is able to associate virtual connections for the connectionless UDP protocol. This type of firewall technology is an improvement over the static packet filtering mechanism.

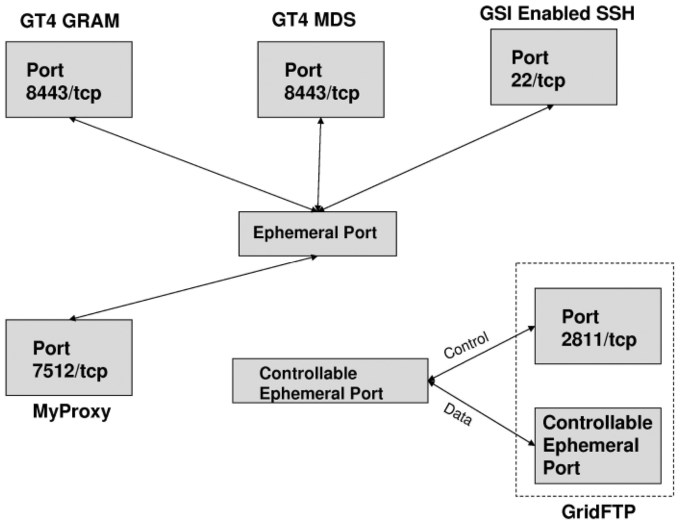


Fig. 8.2. Firewall requirements for Globus

8.2.2 Firewalls and Grid – Issues

Currently most research and development activities in grid computing takes place for the e-sciences community. The community is big and the research challenges are enormous. However, when the grid moves to enterprises several interesting and critical challenges will be witnessed. Some of the challenges and possible efforts have been highlighted in the previous chapters. Another big challenge is the integration with the firewall technologies. Most of the enterprises employ some amount of firewalls and packet filtering and efforts need to be taken to solve the problem of easy integration with the existing firewalls.

Globus and Firewall

Figure 8.2 shows the firewall requirements for different components of Globus. In the figure, a controllable ephemeral port describes a port which is selected by the Globus Toolkit, which is constrained by a configurable limit. On the other hand, an *ephemeral port* describes a non-deterministic port assigned by the system in the range less than 1024. The requirements of the different components are described as follows:

- **GSI:** GSI involves the authentication, confidentiality, integrity, and secure delegation modules of Globus. The request should originate from an ephemeral port and similar to ssh configuration, the server listens to port 22.
- **GRAM:** GRAM is the resource management module of Globus. In the GT4 GRAM, connections are initiated by the client from an ephemeral port. To initiate and control jobs, all traffic goes through a single hosting environment defined by port 8443/tcp. For GT3, this port is 8080/tcp.
- **MDS:** MDS is the monitoring service of Globus. Similar to GRAM, connections are initiated by the client from an ephemeral port and all traffic goes through a single hosting environment defined by port 8443/tcp. As in GRAM GT3, for MDS GT3 this port is 8080/tcp.
- **MyProxy:** As mentioned in Chap. 9, MyProxy is a credential storage service for X.509 credentials. MyProxy connections are authenticated and secured with GSI and are normally from ephemeral ports on the client to 7512/tcp on the server.
- **GridFTP:** Similar to any FTP service, GridFTP also requires two different channels: control and data channels. The control connection is established from a *controllable ephemeral port* on the client to the well-known static port of 2811/tcp on the server. In the case

of a single data channel, the connection is established from a *controllable ephemeral port* on the client to a *controllable ephemeral port* on the server. In the case of third-party transfers (a client controlling a file transfer between two servers), this connection may be from a server to another server. In the case of multiple parallel data channels, the direction of the connection establishment is dependant on the direction of data flow – the connection will be in the same direction the data flow.

Adaptive Firewall for the Grid (AGF)

The Adaptive Firewall for the Grid (AGF) [147] is a project done at Technical University of Denmark (DTU). The main motivation behind the work is the observation that to meet the grid firewall requirements, the administrators need to open several well-known ports, and a range of ephemeral ports for incoming connections. This can be dangerous as adversaries may be able to sneak into the system through the open ports. The AGF system develops a mechanism so that the firewall can adaptively open and close ports based on service requests. The firewall will open the ports when it receives authenticated requests. Moreover, the firewall will close the ports when there are no service activities on those ports.

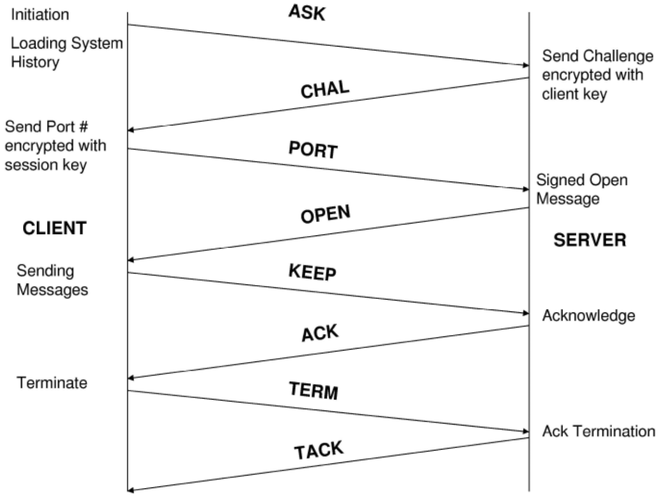


Fig. 8.3. Overview of AGF messages

Following are the different messages exchanged by the client and the server which is illustrated in Figure 8.3.

- *ASK* is sent from client and is the first message for a session. The message indicates that the client wants to open a port.
- Once the server receives the ASK message from the client, the server sends the *CHAL* message to the client. This contains a challenge to the client, encrypted with the client's primary key. It also includes the session key to be used.
- The third message is the *PORT* message which is sent from the client and includes the expected port number that the client wants to open.
- On receiving the PORT message, an *OPEN* message is sent from server indicating the firewall could be opened for such a port.
- Then, a *KEEP* message is sent from the client, confirming that the client wants to open the firewall. The server then opens the firewall.
- *ACK* is sent from server, indicating the firewall has opened, and the client should continue to send the keep-alive message.
- *KEEP* and ACK are continuously sent between client and server, until termination of the connection takes place.
- *TERM* is sent from client, when the client wants to terminate the firewall. The server closes the firewall when such message is received.
- "TACK" is sent from server, acknowledging that the server has terminated the firewall. All the previous messages (from PORT to TACK) are encrypted by the shared session key.

8.2.3 Firewalls and Web Services

Let us now focus on some of the issues and solutions in integrating firewalls with Web services.

Flexibility

As mentioned in [148], firewalls place some constraints and reduce the ease for the home users especially in case of asynchronous message processing. This makes an assumption that that SOAP server/listener need to be installed in the home machine, and the machine over the Internet would be able to send message through the home user's firewall. This decreases the flexibility of the design which led to the concept of WS-Polling [148], which works in the principle of emails and SMTP servers.

Effectiveness

With the growth of Web services and XML technologies, the effectiveness of most of the third-generation firewalls (application level firewalls) is under question. There is a need to parse the XML messages and understand the contents of the messages before a policy decision can be made. Hence sophisticated techniques are needed. To cater to these types of requirements, XML firewalls [149, 150, 151] have come into vogue. These are a new generation technology, which operate above the conventional application layer unlike conventional firewalls that operate on the network layer. XML firewalls have the capability of examining an incoming SOAP request, and taking an appropriate action based on the message content. Such content inspection is vital to prevent malicious as well as DoS attacks. Further, XML firewalls can offer nonrepudiation mechanisms by providing audit trails of all service accesses.

Coordination

Peer-to-peer interactions between Web services which are behind firewalls introduce problems as the end-points are inaccessible. Several solutions have been proposed to address this issue. The solution [152] attempts at solving by implementing a mechanism similar to a post office mailbox. A Web service client with no endpoint creates a mailbox and then uses this mailbox address when it needs to receive messages. When the client is ready, it can check the mailbox service (Post Office) for new messages and download them for processing.

8.3 Virtual Private Networks (VPN)

To carry out business related activities, it has become absolutely essential for employees, contractors, and business managers to access confidential resources and communicating them across geography. It has become quite common for business executives to log-on and access resources using laptops while traveling or when they are in client or business locations. Since the communications generally take place over the public network, confidentiality, authentication, and integrity are very important. One prominent communication service which provides these and allows access of resources anywhere and anytime is called Virtual Private Networks (VPN).

Before the advent and popularity of VPN technologies, private networks were created using permanent links between corporate sites. VPN technologies extend this concept by providing *virtual* networks that are dynamic and connection setup can be provided according to organizational needs. Unlike traditional corporate networks, VPNs do not maintain permanent links between end points. Rather the connection is torn down as soon as it is not required resulting in bandwidth savings.

VPN technologies are cost effective alternatives to completely private networks which allow different parties to come together and share resources in a secure manner.

8.3.1 VPNs and Grid – Types of VPNs

There are mainly two different types of VPN technologies. These are Layer 2 VPN service and Layer 3 VPN service.

Layer 2 VPN Service (L2VPN)

In L2VPNs, the provider extends layer 2 services to the customer sites. A key property of L2VPNs is that the provider is unaware of Layer 3-specific (Network Layer) VPN information. The customer and the provider do not exchange any routing information with each other. Forwarding decisions in the provider network are based solely on Layer 2 (Data Link Layer) information such as MAC address, ATM VC identifier, MPLS label, and port number. Currently, two different approaches to L2VPNs are described in the literature, Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS) [153]. The major difference between the two is that the VPWS provides VPN service between one site and another while VPLS provides a service across multiple sites. The VPWS approach can be regarded as a generalized version of the traditional leased line service, in which the sites are connected in a partial or full mesh. The VPLS approach emulates a LAN environment where a site automatically gains connectivity to all the other sites attached to the same emulated LAN.

Layer 3 VPN Service (L3VPN)

In L3VPNs, the provider offers layer 3 (Network Layer) connectivity, typically Internet Protocol (IP), between the different customer sites. At present, there are two dominating L3VPN approaches, *BGP/MPLS VPN* [154] and *Virtual Router (VR)* [155]. Both approaches concentrate the VPN functionality at the edge of the provider network (provider edge or

PE nodes) and hide VPN-specific information from the provider core nodes, to improve scalability. In the BGP/MPLS VPN approach, a routing context is represented as a separate routing and forwarding table in the PE. Each PE node runs a single instance of a BGP variant called Multiprotocol BGP (MPBGP) [156] for VPN route distribution across the core network. PE nodes use MPLS labels to keep VPN traffic isolated and transmit packets across the core network in tunnels. The tunnels are not necessarily MPLS tunnels, they can be of any type, such as IPSec (see Chap. 2). If a tunnel type other than MPLS is used, the only nodes that need to know about MPLS are the PEs. Any routing protocol can run between the Customer Edge (CE) nodes and the PEs, but in practice the customer must use the routing protocol chosen by the provider. In the VR approach, PE nodes have one VR instance running for each VPN context. A VR emulates a physical router and functions exactly like one. VRs belonging to the same VPN are connected to each other via tunnels across the core network.

8.3.2 VPNs and Grid – Issues

If grid computing has to become an important part of any enterprise's infrastructure, there is a need to integrate with the VPN technologies which have become a norm with most enterprises for secure access to the internal resources. Let us now discuss some of the issues in integrating VPNs with Grid technologies.

Manageability

VPNs and grids represent two diametrically opposite paradigms. In a typical grid computing environment, different entities share resources. VPNs, on the other hand, are point-to-point security solutions between two entities. In order to use a VPN over each connection between a user and a resource node, a potentially enormous number of VPNs will be needed, with associated key management challenges for each. This will result in a huge manageability cost for the enterprise. Even a simple example of having one grid node, scheduler, and a few resources, with VPN connections between each of them is not at all feasible.

Performance

Like any other security solutions, VPN will introduce additional overheads which will reduce the overall throughput of the grid systems. Several research works have addressed this issue. In the next subsection, we will discuss Hose, which talks about managing VPN connections and improving throughput in a scalable manner.

Setup

Another issue that hinders the integration of VPN and grid technologies is the requirement of manual configuration required at each VPN. In a grid computing environment, flexibility is one of the key drivers. Nodes are added or deleted on demand based on the utilization of the systems. This is very difficult to achieve in a VPN setup. Added to this is the issue of trust management, which would be really difficult to manage and maintain. It is to be understood that VPNs do not provide end-to-end security, rather provide security at the network or the data link layer (Layer 3 or Layer 2 security). Grids, on the other hand, require security at the message level. Therefore, integration is needed which is always through manual setup, and hence not scalable.

8.3.3 VPNs and Grid – Some Solutions

Several research efforts have been undertaken in combining VPN and grid services. In this chapter, we will discuss in brief about two such solutions: Hose and On-demand grid support system.

Hose – A Resource Management Solution

The Hose service model [157] is an effort to provide flexible resource management in a VPN environment. Proposed by researchers from AT&T® Research, the Hose service model is characterized by aggregate traffic from a set of end-points to another in a VPN. The hose service model is a flexible alternative to the customer pipe service model, where a customer buys a set of fixed allocations (customer-pipes) from the service provider. In this model, the customers specify the incoming and outgoing traffic aggregated over the different sites in the VPN system.

Figure 8.4 shows a traditional VPN setup where proper provisioning of bandwidth is required to satisfy the Service Level Agreement (SLA) for each customer. Taking the same example for a hose model, each site would be provisioned by the aggregated amount of traffic coming in and going out of the site. Following are the advantages of the Hose model:

- **Flexibility:** The Hose model allows the flexibility of clubbing together traffic having similar QoS requirements. Overall, it provides more flexibility in terms of resource allocation and utilization.
- **On demand resource:** This type of model fits nicely with the grid vision as resources could be adjusted on demand.

In spite of the flexibility provided by this model, one of the main disadvantages of this type of model is the lack of QoS guarantees that it can provide. Since the resources can be shared, the absolute guarantees are hard to provide which became a bottleneck for such a system to be accepted widely.

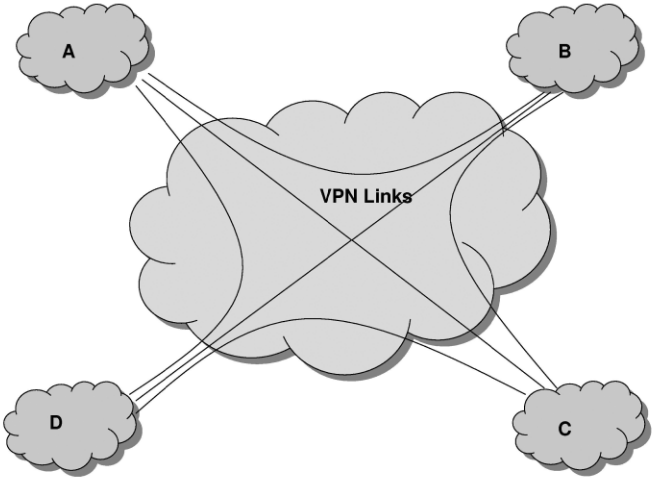


Fig. 8.4. A traditional VPN setup

On-Demand VPN Support for the Grid

In [158], the authors have proposed a network resource abstraction for resource discovery of on-demand VPN. The main contribution of the work lies in the abstraction of the information provided so that the VPN resources can be discovered. The proposed abstraction is implemented and integrated with Globus MDS, version 2. The two main components of the abstractions are:

- **Path Element (PE):** The different elements of the VPN service are abstracted into the concept of Path Element (PE). The PE provides unidirectional connectivity between two network nodes. A network node can represent a single device like an end-system, a router or a switch or a network domain like an autonomous system, IP network, or even a LAN. PE is the generalized class from which technology dependant classes like the DiffServPathElement and the LSPPathElement classes are derived.
- **Path Discovery:** The process of path discovery is carried out by simple match making based on some *Service Attributes.* Different types of service attributes are *Service Types*, which indicates the type of service (a premium service for example); *Time,* which indicates the amount of reservation time; *Application Type*, and *Service Properties.*

8.4 Secure Routing

Routing tables are used to route packets over any network especially the Internet. Routing protocols like distance vector, link state, and path vector protocols have been designed to create routing tables through the exchange of routing packets. Routing table "poisoning" is a type of attack on the routing protocols where the routing updates are maliciously modified by the adversaries resulting in creation of wrong routing tables. A simple example of routing table "poisoning" leading to DoS attack has been described in Chap. 6.

8.4.1 Impacts of Routing Table "Poisoning"

Routing table "poisoning" can have impacts like suboptimal routing, congestion, partition, overwhelmed host, looping, and illegal access to data.

Suboptimal Routing

With the emergence of the Internet as a means of supporting soft real-time applications, optimality in routing assumes significant importance. Routing table poisoning attacks can result in suboptimal routing that can affect real-time applications. Similarly, in a grid scenario also this type of attack may lead to suboptimal routing resulting in a QoS violation.

Congestion

Routing table "poisoning" can lead to artificial congestion if packets are forwarded to only certain portions of the network. Artificial congestion, thus created, cannot be solved by traditional congestion control mechanisms.

Partition

The "poisoning" attack may result in the creation of artificial partitions in the network. This can become a significant problem since hosts residing in one partition will be unable to communicate with hosts residing in the other partition.

Overwhelmed Host

Routing table poisoning may be used as a weapon for DoS attacks. If a router sends updates that result in concentration of packets to one or more selected servers, the servers can be taken out of service because of huge amounts of traffic. This type of DoS attack is more potent as the attacker is not spoofing identity, and is thus impossible to detect by the detection techniques mentioned in Chap. 6.

Looping

The creation of triangle routing caused due to packet mistreatment attacks could also be simulated through improper updation of the routing table. Loops thus formed may result in packets getting dropped and hence lowering of the overall network throughput.

Access to Data

Adversaries may gain illegal access to data through the routing table poisoning attack. This may lead to adversaries snooping packets which were not supposed to pass through that part of the network.

8.4.2 Different Routing Protocols

Routing protocols can be broadly categorized into three main categories: distance vector, link state, and path vector routing protocols.

Distance Vector

In this set of protocols, the nodes in the network create a vector of shortest paths distances to all the other nodes in the network. This distance vector information is exchanged between the nodes. After receiving the distance vector information from its neighbors, each node calculates its own distance vector. One point to note about these protocols is that, no node has the full topology information and depends on its neighbors for creating its routing tables. It has been shown that several problems like the Count to Infinity problem can be a result of not having the full topology information. Routing Information Protocol (RIP) [159] is an example of distance vector protocol.

Link State

In link state protocols, each node sends its connectivity information to all other nodes in the network. Based on the information received from all other nodes, each node computes the shortest path tree by applying the Bellman Ford algorithm. Unlike the distance vector protocol, each node participating in the link state protocol has the full topology information. As a result, link state protocols are inherently robust. Open Shortest Path Forwarding (OSPF) [160] is an example of the link state protocol.

Path Vector

This protocol is a variation of the distance vector. In this protocol, each node sends the full shortest path information of all the nodes in the network to its neighbors. It has been shown that problems associated with standard distance vector protocols can be avoided in the path vector protocol. Border Gateway Protocol (BGP) [161] is an example of the path vector protocol.

8.4.3 Routing Attacks and Countermeasures

Routing table poisoning can be broadly categorized into (a) link and (b) router attacks. Link attacks, unlike the router attacks, are similar in case of both link state and distance vector protocols.

Link Attacks - Interruption

Routing information can be intercepted by an adversary, and the information can be stopped from propagating further. However, interruption is not effective in practice. The reason for this is that, in the current Internet scenario there is generally more than one path between any two nodes, since the average degree of each node is quite high (around 3.7). Therefore, even if an adversary stops a routing update from propagating, the victim may still be able to obtain the information from other sources. Most routing protocols employ robust updates between neighbors [159, 160], by using acknowledgments. Link attacks are detected in those cases. However, if links are interrupted selectively, it is possible to have unsynchronized routing tables throughout the network. The after-effects of such routing tables are looping and denial-of-service. Unsynchronized routing tables can also be created if a router drops the updates, but sends an acknowledgment. The problem of router dropping routing updates selectively has not been studied in the literature.

Link Attack – Modification/Fabrication

Routing information packets can be modified/fabricated by an adversary who has access to a link in the network. As solutions for this problem, digital signatures are generally employed. In case of digital signatures, the routing updates increase by the size of the signature (typically between 128 to 1024 bits). This is a viable solution in link state routing protocols, since the LSAs are transmitted infrequently. This is also proposed as a solution for distance vector protocols. Distance vector protocols suffer from excessive bandwidth consumption as the distance vectors are exchanged quite frequently. Therefore, the addition of extra overhead in the form of a digital signature has been looked upon by the research community with concern. Efforts have been undertaken to reduce the overhead through the use of efficient digital signatures [162]. Another problem with this approach is that it relies on the existence a public key infrastructure (PKI) for its functioning [163]. In the absence of a PKI, the proposed solutions are not viable.

Link Attack – Replication

Routing table "poisoning" can also be in the form of replication of old messages, where a malicious adversary gets hold of routing updates and replays them later. This type of attacks cannot be solved using digital signature schemes, because the updates are valid, only they are time shifted. As a solution to this problem, sequence information are generally used.

Sequence information can be in the form of sequence numbers or timestamps. An update is accepted as a valid update if the sequence number in the packet is greater than or equal to the sequence number of the previously received update from the same router.

Router Attacks – Link State

A router can be compromised, making it *malicious* in nature. Router attacks differ in their execution depending on the nature of the routing protocol. In case of link state routing protocol, a router sends information about its neighbors. Hence, a malicious router can send incorrect updates about its neighbors, or remain silent if the link state of the neighbor has actually changed. A router attack can be *proactive* or *inactive* in nature. In case of proactive router attack, the malicious router can add a fictitious link, delete an already existing link, or change the cost of a link proactively. In case of inactive router attacks, a router ignores a change in link state of its neighbors. The solutions proposed for router attacks in link state protocols can be categorized into two types: *intrusion detection* and *protocol-driven*. The use of intrusion detection techniques have been suggested as a mechanism to detect router attacks [164]. In these techniques, a centralized attack analyzer module detects attacks based on some possible alarm events sequences. Using an attack analyzer module in the Internet scenario is not a scalable solution. In a protocol-driven solution, the detection capability is embedded in the link state protocol itself. In [165], Secure Link State Protocol (SLIP) has been proposed, where attack detection capability has been incorporated in the routing protocol itself. A router does not believe an update, unless it receives a "confirmation" link state update from the node supporting the questionable link. However, the solution is not complete as it works only in a symmetric network where both nodes supporting a link can identify the change in the link state. It also makes an assumption that no malicious collusion exist in the network.

Router Attacks – Distance Vector

Unlike the link state, in the case of distance vector protocols, routers can send wrong and potentially dangerous updates regarding any nodes in the network, since the nodes do not have the full network topology. In distance vector protocols, if a malicious router creates a wrong distance vector and sends it all its neighbors, the neighbors accept the update since there is no way to validate it. As the router itself is malicious, standard techniques like digital signatures do not work. In [166], the authors have proposed a validation scheme through the addition of predecessor information in the distance vector update. Several other variations of this algorithm [167] have also being proposed. However, most of these solutions work under some assumptions. More research is needed before these solutions can be adopted in practice.

8.5 Multicasting

The proliferation of group applications associated with the growing concern for secure group communication drives the need for efficient and secure solutions for group communication services [168 - 172]. Multicasting is an effective mechanism for supporting group communication. In a multicast communication, each sender transmits only one copy of each message that is replicated within the network and delivered to multiple receivers. For this reason, multicasting typically requires less total bandwidth than separately unicasting messages to each receiver. Since most of communications occurring on a grid imply many participants that can be geographically spread over the entire planet, multicast protocols can provide an efficient way of handling the data transfer. Several multicast systems have been developed for grid based applications [173]. Since there are lots of security issues in multicasting where still research is being carried out, most of the grid based implementations do not have them in place. In this section, we provide a brief overview of existing general multicast security issues and solutions.

8.5.1 Secure Multicasting

Secure multicasting, like any other fields of security, is based on the principle of *confidentiality* and *integrity*. These solutions cannot be trivially extended to multicasting scenario for multiple reasons:

- Unlike unicasting, in multicasting, data has to be shared among multiple group members.
- Group members may join/leave the multicast session at any time.
- Nonmembers must be prevented from collaborating (colluding) to recover the session key.

The issues that differentiate secure multicasting from any other modes of secure communication can be broadly categorized into *member authentication*, *key management* and *packet authentication*. An interaction between the different modules of secure multicasting is shown in Fig. 8.5. Whenever a user (A in the figure) is accepted into the multicast group, a new key is generated (in the rekeying module) and distributed (in the key distribution module). The source authentication module comes into the picture when the multicast group receives packets from the sender S. The sender may be part of the group or may be outside the group. When a member leaves the multicast group, rekeying and key distribution modules are invoked. The three issues along with their proposed solutions are listed in Fig. 8.5.

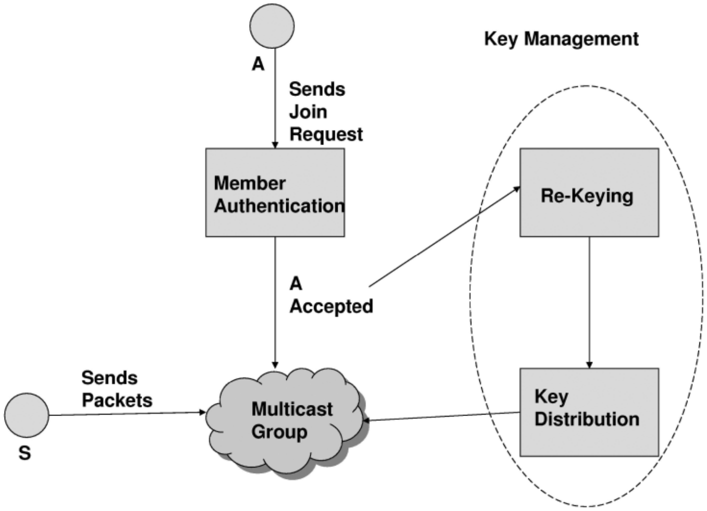


Fig. 8.5. Overview of the different issues in secure multicasting

Member Authentication

Group management involves the fundamental functionality of admission control of the group members. In secure multicasting, only valid users should be able to access data. For example, in a live lecture session only the registered users should be able to listen to the lectures. Member authentication has been traditionally performed in multicasting using a centralized architecture. Recently some systems have been proposed which authenticate members in a hierarchical fashion. Scalability and complexity in implementation are the main selection criteria.

- In **centralized member authentication**, a centralized controller is used for authentication purposes. It is simple to implement. However, it introduces a single point of failure. An example of such schemes is the Core Based Tree (CBT) mechanisms.
- In a **hierarchical member authentication** scheme, there are multiple controllers arranged in a hierarchical fashion. The member joins to the nearest controller. Iolus architecture [174] introduces the hierarchical member authentication approach. These systems are more scalable; however controllers need to be deployed in each multicast node.
- In case of **distributed authentication techniques**, the authentication capability is distributed throughout the network. In [175], the authors describe a mechanism where the authorization server provides access tokens to the group members and access control lists to the routers. These types of schemes require additional intelligence at the router level.

Key Management

After the establishment of multicast session, management of keys needs to be handled. Management of keys involve: distribution of keys in a scalable and secure manner to all the group members, and rekeying when members join/leave the multicast group. Distribution of keys remains an interesting and challenging problem. This is quite different from a unicast scenario, where a key is to be shared by the sender and the receiver only. In multicasting, keys need to be shared among the group members in a scalable manner.

- **Core based techniques** were discussed by [176]. In this scheme the core of the multicast group is also responsible for distribution of keys.
- In the **hierarchical key distribution scheme**, the distribution is split up among different entities in a hierarchical fashion. Two common hierarchical key distribution schemes are [174], and KHIP [177].
- In the last few years, a new technique [178, 179] has been proposed which allows the use of single group data key for data transmission (as in core based schemes), as well as having scalable add and delete operations (as in hierarchical schemes). These techniques are referred to as **tree based techniques** or key graph techniques. The main idea behind these techniques is to have a single server or core, and to have the server distribute subgroup keys in

addition to the individual user and group keys. To balance the cost of addition and deletion, the keys are arranged in logical hierarchy (instead of physical hierarchy as in Iolus), with the root key being the root and the individual user keys being the leaves. The subgroup keys then correspond to the intermediate nodes of this conceptual tree.

Rekeying is needed as keys need to be changed to prevent the users from accessing them after the user leaves a multicast group.

- **Individual Rekeying**: In this technique, new keys are created as soon as a member joins/leaves a multicast group to provide join/leave secrecy.
- **Periodic Batch Rekeying:** In this type of rekeying, requests are collected for a certain amount of time (called the rekey interval), and then rekeying is carried out in a batch. The batch rekeying techniques have been explained and analyzed in [180].

Source Authentication

The source authentication problem is different in multicasting than in traditional unicasting, as the authentication has to be carried out in a scalable manner, and there is a possibility of group members colluding.

- **Individual Signature:** The simplest way to deal with the authentication problem is to sign each and every packet. This is not a practical scheme as this is computationally very expensive. However, the scheme does not introduce any extra delay into the authentication mechanism, *i.e.* the latency is low. The scheme also is not vulnerable to collusion, as each and every packet is signed.
- **Single MAC Schemes:** Message Authentication Codes (MAC) are also traditionally used for authentication. A MAC creates an output from a shared secret key and the message itself. The output, the signature of the packet, is then appended to the packet itself for transport across the network. MACs are generally faster to compute than the public key signatures; they also require that every receiver has access to a shared key. This is different from the public key approach, where any user can have access to the key. These schemes are more efficient than the individual signature scheme. Any receiver can pose as a sender by signing a message of its own with the shared key.

- **Stream Signing:** A stream can be defined as a potentially very long (infinite) sequence of bits that a sender sends to a receiver [181]. The key aspect of the stream is that the receiver must take data as received, and process it as soon as possible. In [181], the authors assumed that it is possible for the sender to embed authentication information into the stream, and the receiver has a small buffer where it can authenticate received bits. The stream is divided into blocks, and the authentication information is embedded into the stream. The information from the i^{th} block is used to authenticate the $(i+1)^{th}$ block. The signer needs to sign only the first block, this signature will propagate through the rest of the stream through authentication information. The above mechanism requires high efficiency and packet loss cannot be tolerated. Therefore, there must exist an underlying reliable transport protocol like TCP.
- **Chaining:** To increase the efficiency of the stream signing protocols, in [182] efficient schemes have been proposed which remove the deficiencies of the stream signing problems. To accomplish these requirements two chaining schemes have been proposed to sign and verify multiple packets, denoted as blocks, in a single operation. Since all the packets in a block are signed and verified in only one operation, the procedure can be amortized over the entire block, making the overall rates much faster than the other methods. The basic function of both these methods is to compute the block digest of each block as part of the authentication information. The authentication information consists of the signed digest of the block and some addition chaining information so the receiver may verify where it belongs in the block. This is a very efficient approach. However, unequal authentication information may lead to bursty traffic.

8.6 Sensor Grids

With the growth of the data-centric nature of grid computing applications, newer grid applications are being developed which requires interaction with *sensors* and *actuators* for tapping into the resource specific information. Some recent works in this area have been captured as part of our discussions on monitoring systems in Chap. 11. With the emergence of *sensor networking technologies* as a major area of research and development, researchers have been able to integrate the tiny sensors and actuators with general purpose computing elements. These networks typically consist of hundreds or thousands of self-organizing, low-power, low-cost wireless nodes deployed en masse to monitor and affect the environment. Sensor networks have found applications in various fields, viz. medical and patient care, military, supply chain, etc. [183]. Researchers are working on integrating grid systems with the sensor networks. One such system is the Hourglass model developed in Harvard [184]. Integrating sensor networks into the grid infrastructure poses challenges in terms of routing, aggregation, and querying, diverse sensor network data. Since the sensor networks can be dispersed geographically, the challenge of routing becomes quite critical. In addition, since the sensors have low computation capabilities, and requires mechanisms to conserve power, the ability to aggregate and query information in a sensor network is a challenge. Security is another great challenge in a sensor networking infrastructure. Integration of sensor networks in a grid requires mechanisms to address the security challenges posed. Since the grid based sensor networks have limited implementations, we provide an overview of security challenges and some solutions in generic sensor networks.

8.6.1 Security in Sensor Networks – Issues

Sensor networks can be of mainly two types: centralized and distributed. The difference between the two lies in the way in which data is aggregated across the sensor nodes. In the first type, sensor networks often have one or more points of centralized control called base stations or sinks. A base station is typically a gateway to another network, a powerful data processing or storage center, or an access point for human interface. The sensors collect information and pass it on to the base stations for further processing. Typically, base stations have more processing capabilities and battery lifetime compared to the individual sensors. In the distributed sensors, the data is stored in a distributed manner in individual sensors. However, most of the work done assumes a centralized architecture, where the sensor nodes establish a routing forest with the base station as the root.

Sensor networks typically consist of small sensors or motes which limit the amount of processing that a sensor node can carry out. Most of these sensors have limited battery power and hence need to be in a sleep mode for most of the time to conserve energy. Sensor networks also are generally deployed in places where bandwidth availability may be very limited. All these constraints limit the sensor network designers from implementing expensive cryptographic mechanisms. Since bandwidth is very expensive in most sensor network implementations, therefore special care must be taken in ensuring that too much bandwidth is not wasted in adding security measures limiting the sizes of the messages exchanged.

Let us now discuss some of the attacks possible in a sensor networking infrastructure. Interception, node hijacking, sybil, sinkhole, and wormhole are some of the attacks which can make the sensor networks vulnerable. The attacks can be performed by a *mote class* or a *laptop class attacker*. The former class of attackers has the same capabilities as that of the sensor nodes, while the latter class of attackers has more capabilities than the sensor nodes.

Interception

Perhaps the easiest means of attack in a sensor networking environment is eavesdropping or information gathering in a passive manner. A laptop type attacker can easily intercept the data streams passing between different sensor nodes or between sensor nodes and the base station. The passive nature of the attacker can hide the information about the attack and hence the identity of the attacker. A more malicious adversary can actually change the contents of the information to cause confusion. However, content corruption can lead to much easier detection. Encryption techniques can be used to mitigate this type of attack.

Node Hijacking

There are two modes of hijacking a node: a node can be compromised and secret information in that node can be obtained. In such a case, there is a need to exclude the compromised node. The second type of hijacking is where a new node is introduced into the network. The node not only introduces spurious and sometimes potentially dangerous information, it also can consume a lot of network bandwidth which is a scarce commodity.

Sybil Attack

In this type of attack, an attacker poses multiple identities to other nodes in the sensor network. The attack is dangerous especially in the context of routing in sensor networks. Assume that a node A is posing as n additional nodes A1, A2, …, An. When routing a packet within the network, one of the fictitious nodes may be chosen as the next hop resulting in unauthorized data access, additional latency, resource wastage, looping, and sometimes even network partitioning. If the attacker is an external attacker, some type of authentication may reduce the probability of such an attack. This type of attack is very difficult to detect and prevent especially when the attacker is a laptop type attacker and is an internal member of the network.

Sinkhole Attack

In this type of attack, the attacker acts as if all the traffic in the network is intended towards the attacker. Sinkhole attacks typically work by making a compromised node look especially attractive to surrounding nodes with respect to the routing algorithm. Typically, routing algorithms work by the principle of least cost path. If the attacker or adversary can advertise that the shortest path to all the nodes in the network is through it, then all the packets can traverse through it. It is to be noted that such an attack is possible in traditional Internet also. However, the effect is much reduced because of the packet by packet authentication of routing updates required for generating the routing tables. Sinkhole attacks become an important problem in sensor networks because it is a constrained bandwidth environment and it is extremely expensive to even sign the Hello messages exchanged during route establishment.

Wormhole Attack

Another interesting attack is the wormhole attack [185] where an adversary tunnels messages received in one part of the network over a low latency link and replays them in a different part. The simplest instance of this attack is a single node situated between two other nodes forwarding messages between the two of them. In a typical attack scenario, wormhole attacks involve two distant malicious nodes colluding to understate their distance from each other by relaying packets along an out-of-bound channel available only to the attacker. An adversary situated close to a base station may be able to completely disrupt routing by creating a well-placed wormhole. An adversary could convince nodes who would normally be multiple hops from a base station that they are only one or two hops away via the wormhole. This can create a sinkhole: since the adversary on the other side of the wormhole can artificially provide a high quality route to the base station, potentially all traffic in the surrounding area will be drawn through the adversary if alternate routes are significantly less attractive.

8.6.2 Existing Solutions

Several sensor network security solutions have been proposed in the literature. Most of them have been implemented in the lab settings and have limited deployment. We are presenting two of the most widely cited solutions in this area: Security Protocols from Sensor Networks (SPINS), and TinySec.

Security Protocols for Sensor Networks (SPINS)

SPINS [186] is a suite of security mechanisms proposed from the University of California, Berkeley. SPINS has two basic components, SNEP and μTESLA. While SNEP provides confidentiality, authentication, and data freshness, μTESLA provides authenticated broadcast for severely resource constrained environment.

- **SNEP:** SNEP uses encryption to achieve confidentiality and message authentication code (MAC) to achieve two-party authentication and data integrity. In addition to confidentiality SNEP also achieves semantic security, which ensures that the eavesdropper will not be able to infer anything about the message, even if it can get hold of multiple encryptions of the same information. The basic technique to achieve this is randomization: Before encrypting the message with a chaining encryption function the sender precedes the message with a random bit string (also called the *initialization vector*). This prevents the attacker from inferring the plaintext of encrypted messages if it knows plaintext-ciphertext pairs encrypted with the same key. To avoid adding the additional transmission overhead of these extra bits, SNEP uses a shared counter between the sender and the receiver for the block cipher in counter mode (CTR). The communicating parties share the counter and increment it after each block.
- **μTESLA:** μTESLA is a mechanism to authenticate broadcast messages. The heart of the mechanism is the concept of key chains where a chain of one-way keys are computed through a function so that the $(i+1)^{th}$ key is dependent on the i^{th} key. Whenever the base station is sending a packet, it computes a MAC based on one of the keys of the key chain. The receiver sensor nodes stores the packet in the buffer waiting for the key to be disclosed. The base key of the key chain is disclosed so that the other keys can be obtained and the message can be authenticated based on the MAC. Though the mechanisms is efficient, it requires a loose synchronization

between the base station and the receiver sensor nodes as each key corresponds to a particular time period.

TinySec

TinySec is another security architecture [187] for the wireless sensor networks from University of California, Berkeley. TinySec is a lightweight, generic security package that can be integrated into sensor network applications. It is incorporated into the official TinyOS release. The important point that differentiates the TinySec approach with any other security mechanism is its link layer security characteristics. The authors in [187] try to analyze the difference between link layer and end-to-end security. The authors have concluded that the use of link layer compared to end-to-end security is an effective mechanism for sensor network security. The reasons are two-fold. Firstly, since most of the communication is between the sensor nodes and the base station, unnecessary bandwidth is being wasted for having end-to-end security. Moreover, sensor nodes require contents and message suppression to reduce the message overhead, and hence end-to-end security may not be an effective mechanism. Secondly, if message integrity is checked only at the end node, a low-bandwidth sensor network is vulnerable to denial-of-service attacks as messages travel throughout the network. It is to be noted that no detailed analysis have been done in the literature about the applicability of link-level security vis-à-vis end-to-end security in sensor networks.

TinySec provides the basic security properties of message authentication and integrity (using MAC), message confidentiality (through encryption), semantic security (through an Initialization Vector) and replay protection. TinySec supports two different security options: authenticated encryption (TinySec-AE) and authentication only (TinySec-Auth). With authenticated encryption, TinySec encrypts the data payload and authenticates the packet with a MAC. The MAC is computed over the encrypted data and the packet header. In authentication only mode, TinySec authenticates the entire packet with a MAC, but the data payload is not encrypted. In terms of keying mechanisms, TinySec allows multiple types of mechanisms. It allows a network-wide single key based mechanism, a key per node-pair, or a group of neighboring nodes sharing a TinySec key.

8.7 High Performance Interconnects

Performance is one of the main criteria which drives the adoption of different high speed interconnect technologies to create very high performance clusters. In this section, we will talk about the two most popular high speed interconnects used by the cluster community, namely the 10-GB Ethernet and Infiniband.

8.7.1 10-Gigabit Ethernet

One of the most popular high speed interconnect technology available is the 10-Gigabit Ethernet. The 10 Gigabit Ethernet standard extends the IEEE 802.3ae standard protocols to a wire speed of 10 Gbps and expands the Ethernet application space to include WAN-compatible links. Under the Open Systems Interconnection (OSI) model, Ethernet is fundamentally a MAC Layer protocol. 10 Gigabit Ethernet retains the key Ethernet architecture, including the Media Access Control (MAC) protocol, the Ethernet frame format, and the minimum and maximum frame size. 10-Gigabit Ethernet sticks to the basic Ethernet architecture and differs by only one key ingredient. Since 10 Gigabit Ethernet is a full-duplex only technology, it does not need the carrier-sensing multiple-access with collision detection (CSMA/CD) protocol used in other Ethernet technologies. In every other respect, 10 Gigabit Ethernet matches the original Ethernet model. Since 10-Gigabit Ethernet is compatible with the Ethernet architecture, therefore, applications can be seamlessly migrated to this infrastructure. In addition to 10Gigabit Ethernet's advantage with respect to compatibility with legacy infrastructures, Balaji et. al [188] show that it also delivers performance that is comparable to traditional high-speed network technologies such as Infiniband and Myrinet in a system-area network environment to support clusters and that 10Gigabit Ethernet is particularly well-suited for socket-based applications.

8.7.2 Infiniband Architecture (IBA)

Infiniband is a switched fabric based interconnect architecture operating at a base speed of 2.5 Gbps to 10 Gbps in each direction. Instead of a shared bus architecture, Infiniband uses a switched fabric architecture. A point-to-point switch fabric means that every link has exactly one device connected at each end of the link. Thus the loading and termination characteristics are well controlled and also it can provide better scalability and fault-tolerance. Infiniband is one of the most popular interconnect technologies especially for high-speed, high-performance clusters. Several case studies can be found where Infiniband is used in grid based deployment.

8.7.3 Some High Performance Security Solutions

One of the major issues in implementing security solutions in a high-speed networking environment is performance. One of the earliest attempts to identify the attacks and solutions in a high-performance distributed environment is provided by Dimitrov and Gleeson [189]. The authors presented security enhancement methods in three levels: network host interfaces, SANs, and protocols for interconnecting many SANs. Their approach can be a good systematic guideline for enhancing the security of cluster systems based on the Myrinet or Virtual Interface Architecture (VIA). Another security solution that talks specifically for the Infiniband architecture is presented in [190]. A security solution which requires firewall/intrusion detection, encryption/decryption, message authentication, distributed denial of service (DDoS) attack protection, etc., it results in a significant overhead which significantly reduces the performance. Cyber Security Processor (CYSEP) [191] is a security solution implemented in an Application Specific Security Circuit (ASIC) which provides very good performance at high speed. In this chapter, we will discuss about the security enhancements for IBA and the CYSEP solution.

Security Enhancements for IBA

In [190], the authors have mentioned that Denial of Service (DoS) attacks are perhaps one of the most critical challenges that need to be addressed in an IBA based network. IBA specifies a mechanism, referred to as *partitioning*, for grouping nodes to limit access control [12]. Partitioning enables several nodes to exclusively share some resources, forbidding other nodes not in the same partition to access them. However, an attacker on a compromised InfiniBand node can easily trigger a DoS attack by flooding packets with random partition keys in the InfiniBand network. Destination nodes will block those packets because they do not have legitimate partition keys. In the mean while, the packets have already traversed through the network, incurring a significant delay to other legitimate traffic. IBA allows the use of five different types of keys for isolation and protection. Since the keys are available as plain text in the messages, the key level information can be easily obtained by the hacker. The authors propose to reduce the effect of the problem by two different types of key management:

partition level key management and *queue pair-level key management.* In the former management mechanism, each partition has a key associated with it. The key is not created through a Key Exchange protocol; rather the Subnet Manager (SM) creates the key. For a more granular control, a key is associated with each queue pair.

Cyber Security processor (CYSEP)

CYSEP is a high speed security module for high speed networks. The CYSEP [191] supports, at wire-speed, four major functions, namely, firewall/intrusion detection, encryption/decryption, message authentication, and distributed denial of service (DDoS) attack protection at the speed of 10 Gbps or higher. The Firewall and Intrusion Detection Engine (FIDE) prevents attacks and filters unwanted content at the edge of a network. It includes the functions of finite automata based signature detection and packet classification. The encryption/decryption engine implements the message confidentiality primitives necessary to establish VPN over the public Internet. The authentication and authorization engine implements the message integrity primitive necessary to establish VPNs. The DDoS protection engine uses the packet-score scheme, which estimates the legitimacy of a suspicious packet according to the score assigned. The authors have claimed that the throughput of the system to go beyond 70 Gbps in the case of application specific integrated circuit implementation.

8.8 Chapter Summary

The network is one of the most important components of any infrastructure and grid is no exception. Different network security issues are: Issues related to network access control and isolation, secure routing, multicasting, wireless/sensor grids, and high performance interconnects. The first issue is of immediate concern, where efforts are needed in integrating the grid solutions with existing enterprise firewalls and VPNs. Most of the current solutions include manual intervention or customizations to integrate with firewalls. Several research projects like Adaptive Grid Firewall (AGF) are looking at this issue. Solutions like Hose look at using VPNs for resource management purposes. The second issue of secure routing is important and several research efforts have been undertaken. Both third and fourth issues are long term security issues which include secure multicasting over the grid and secure sensor networks. These have exceptional research potentials. Finally, high performance interconnects introduce security challenges and solutions like Secure IBA and CYSEP and efforts in that direction. Table 8.1

summarizes the different network issues and solutions discussed in this chapter. With this we come to the end of the infrastructure related issues. In Chap. 9 - 11, we will look at the management related issues starting with the credential management systems.

Table 8.1. Summarizing the different network issues

Types	Issues	Solutions	Comments
Network Access Control & Isolation	Firewalls – Flexibility, effectiveness, and coordination Issues	AGF, XML Firewalls	More research and development efforts are needed in automation front
	VPNs – Manageability, Performance, and setup issues	Hose, On-demand VPN	Similar to Firewalls, more work is needed for automated setup and deployment
Routing	Link Attacks	Signature based	Most of the common problems have solutions, performance is the key
	Router Attacks	Consistency based, Intrusion detection based	Source related attacks require more research attention
Multicast	Member Authentication	Centralized, hierarchical, and distributed techniques	Hierarchical and distributed schemes are scalable. However, they require more intelligence.
	Key Management	Core-based, hierarchical, and tree-based techniques	Tree based techniques are scalable, require more research
	Source Authentication	Individual, single MAC, stream-based, chaining	Stream signing and chaining are scalable. However, they introduce restrictions in terms of transport protocol and unequal information size.
Sensor Networks	Node Hijacking, Sybil Attacks, Wormhole attacks	SPINS and TinySec	More research is needed in this area
High Perform Networks	Performance related	IBA Security, CYSEP	More performance to security trade-offs are needed

9 Grid Credential Management Systems

9.1 Introduction

When I reflect upon my activities of the day I find that I have used multiple credentials to access resources of different forms. I used my company identity card to enter the office premises, entered the password to enter into the office network, used the smart card to access the high-security lab, used the Personal Identification Numbers (PIN) to access my ATM account, and used my passport to get a US visa, and this was just one day. Different identity checks were required by different systems, and my identities were in different forms which I either carried in my head or as a card or a paper. I am surely not an exception; every one of us is doing the same, maintaining multiple credentials to access some form of resource. This has really become pronounced with the growth of information technologies, where there are multitudes of system interfaces which require some sort of user authentication. As a result individuals possess multiple digital identities and credentials, many of which are short lived.

At this point, one may be concerned about the relationship between identities and credentials. Identity of an individual user is unique; however it may be manifested in different ways to disparate systems through user credentials. For example, my identity credential to the US consulate is my passport, while to the company network is the combination of network's user id and password. Therefore, when we talk of managing different user identities, it is actually the user identification credentials we are talking about. However, credentials go beyond just identifying the user. Credentials may authorize a user to access certain resource or can be used as a proof of authentication. Credentials can be short-lived, for example identity cards or passwords which expire when the individual leaves the company, or after a fixed amount of time as the case may be. Other examples of short time credentials are the tickets issued in busses for a short ride. Therefore, the individuals manage their credentials by a combination of paper, cards, and own memory as I did today. Secure management of user credentials is a very important challenge. Identity theft has topped the list of complaints to the US Federal Trade commission in 2002, accounting for 43% of all the complaints [192]. Therefore, identity and user credential management is surely a very important problem and several research and development efforts are undertaken in this direction. In a grid system also, the management of credential assumes enormous proportion because of the heterogeneity of resources, policies, and possible authentication and other security mechanisms across the grid. In this chapter, we will talk about credential management issues and solution in the grid computing environment. Before discussing the grid specific issues and solutions, we will talk about the generic properties and systems used in this space.

9.1.1 Types of Credentials

Let us now discuss the different types of credentials generally used in digital systems. Credentials are of three main types: identity credentials, authentication credentials, and authorization credentials.

Identity Credentials

This type of credentials is used to uniquely identify a particular user. The use of identity credentials is quite prevalent in the physical world also. There are numerous examples when one has to prove one's identity. Foreign travels, applying for credit cards, buying liquor, are few examples when proving one's identities becomes mandatory. Passports, driving license, or some other government issued photo identity cards are generally used for this purpose. Identity credentials prove one's identity as it is vouched for by a *trusted third party*. Therefore, even though a person may not be trusted, the identity proves that there exists a trusted third party who vouches for the person's identity. In the digital world also in many cases proving a user's identity becomes important. Some form of digital certificates signed by a certificate authority is used to prove the identity of the users.

Authentication Credentials

In addition to identity credentials, another form of credentials typically used is called the authentication credentials. At this point, one may wonder about the utility of the authentication credential, when the identity credentials actually can be used to authenticate a user. One way to differentiate between identity credentials and authentication credentials lies in the longevity of the credentials. Identity credentials are generally long-term while authentication credentials have lesser longevity. Another important distinguishing factor is that identity credentials are generic in nature while authentication credentials generally have system specific and other policies like temporal policies ingrained in them. The authenticating systems use the identity credentials to make the authentication decision before issuing the authentication credentials. For example, visa issued by different governments can be considered to be an authentication credential. The visa is issued by the government official after checking the identity credentials which include passport and other supporting documents. In computing systems, there may be multiple systems which require authentications. Authentication credentials become useful when a user gets authenticated from one system and uses the credential to access the other systems also.

Authorization Credentials

The third type of credentials typically used is called the authorization credentials. These credentials are issued to authorize users to access certain resources. Common examples of such authorization credentials are bus, train, or flight tickets which authorize users to access the respective transport for traveling purposes. Similar credentials are issued to access computing resources also. A detailed description of grid authorization systems is provided in Chap. 5.

9.1.2 Characteristics of Credential Management Systems

The Secure Available Credential (SACRED) working group in Internet Engineering Work Force (IETF) [193, 194] is concerned with the secure use and management of credentials in a roaming or desktop environment using desk tops, laptops, mobile phones, PDA, etc. The main motivation of the group is to develop solutions to support user mobility, use of the same credentials from different network devices, and secure storage of credentials. The group has developed a few RFCs concerning the requirements for securely available credentials, and a framework to achieve that. The requirements are briefly discussed below.

- **Credential Transmission:** The first requirement is that of credential protection during transmission in a networked environment. The RFC [194] mandates the protection of credentials during network transmission using a double layered encryption mechanism through the exchange of a shared or session key between the client and the server. Furthermore, the RFC mandates that the credential transfer protocol should ensure that all the transmitted credentials are authenticated in some way.

- **Credential Storage:** The RFC mandates that the credentials should not be in the clear when stored. The credentials should be defined as an *opaque* data object that can be used by the network device. Clients should be able to recover the credentials from the opaque objects. The credential format should provide a privacy and integrity protection.
- **Heterogeneity:** The RFC mandates that different credential types like X.509, PGP, etc. should be supported and also should allow the use of different cryptographic protocols. This requirement is especially important in a grid computing context as different systems may support different types of credentials and protocols.

Though the requirements provided by the RFCs operate at the high level, they provide guidelines for the credential repository which enables the design of the credential management system.

Desired Characteristics of Credential Management Systems

Based on the requirements provided in the previous section, it is clear that the Credential Management (CM) system should provide secure transmission of credentials, secure storage of credentials, and should cater to different types of systems and mechanisms. Let us now look at the different characteristics that a credential management system would require.

1. **Initiation:** Every CMS should provide mechanisms so that users can obtain the initial credentials from the CMS system. The CM system should provide the required credential after authenticating the user. The authentication can be based on multiple different mechanisms like password based, certificate based, or some other mechanisms.
2. **Secure Storage:** As mandated by the SACRED RFC [193], the long term credentials or the private keys should be stored in the CM systems in a secure manner, preferably encrypted. This is a very important requirement, as the compromise of the long term credential would lead to disastrous consequences.
3. **Accessibility:** This is more related to the utility of the CM system. The CM system should be able to provide credentials when the user needs them. Proper access control mechanisms need to be provided when the credentials are accessed.

4. **Renewal:** Most credentials have a specific expiration time. The CM system should be able to handle renewal of expired credentials.
5. **Translation:** This is important if there are multiple systems having different authentication and security mechanisms. The credentials used in one domain or realm may have to be translated into credentials in other domain which should be handled by the CM system.
6. **Delegation:** As mentioned in Chap. 4, delegation is really important from the grid perspective. CM systems should be able to delegate specific rights to others on the user's behalf.
7. **Control:** Monitoring and auditing of the credential usage is very important because it not only provides a handle to credential compromise, it can be used for pricing if required. Therefore, CM systems should be able to monitor and audit the credentials provided to the users.
8. **Revocation:** Finally, the CM system should provide mechanisms to revoke credentials in case of user compromise.

9.1.3 Different Credential Management Systems

In the previous section we have listed the requirements and characteristics of a credential management system. Based on the characteristics provided in the previous section, the CM systems can be broadly categorized into credential repositories and credential federation systems. As the name suggests, the **credential repositories** or credential storage systems are concerned about securely storing the credentials, generating new credentials on demand, and sometimes generating proxy credentials on user's behalf for delegation purposes. **Credential federation systems** or credential share systems are responsible for sharing the credentials across different domains or realms. Examples of credential storage systems are smartcards, MyProxy etc., and examples of credential share systems are the Liberty Project, VCMan, etc. We will discuss the different systems in subsequent chapters.

A high level overview of the two systems is provided in Fig. 9.1. As illustrated in the figure, unlike the centralized credential management systems, the federated systems are distributed resulting in no single point of failure, however leading to more complexity in having to deal with heterogeneous authentication and policies.

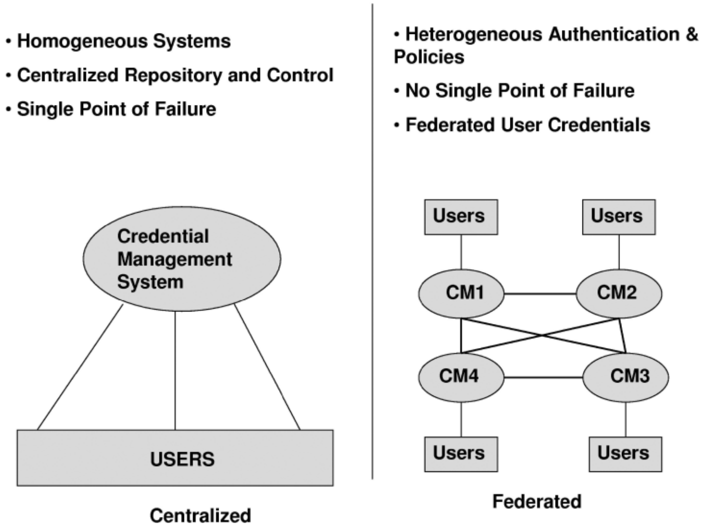

• Homogeneous Systems
• Centralized Repository and Control
• Single Point of Failure
Credential Management System
USERS
Centralized
• Heterogeneous Authentication & Policies
• No Single Point of Failure
• Federated User Credentials
Users
Users
CM1
CM2
CM4
CM3
Users
Users
Federated

Fig. 9.1. Centralized vs. federated credential management system

9.1.4 Centralized Vs. Federated Credential Management

When I enter my organization premises I use my user id and password to access most of the documents and resources of the organization. Here, my credentials (user id and passwords) are used to access the resources through a centralized credential management system. Therefore, in a centralized credential management system there is a need for all the characteristics of a CMS mentioned in the Sect. 9.1 other than translation. The reason is that the system is homogeneous as all authentication mechanisms, policies, etc. are managed centrally. However, centralized credential management systems do not serve all the purposes. For example, I am visiting a city in Switzerland on a vacation. There are two Web sites called xxx.com and yyy.com where the former is responsible for getting me the cheapest fare and the latter is responsible for arranging my stay, arranging for airport drops, sight-seeing, and others. Since I have two different accounts or credentials in each of the Web sites, therefore I would have to log on to the two sites and provide them separate information. Instead, if the flight information can be shared between the xxx.com and yyy.com in spite of my having different credentials in each Web site, it would be extremely convenient for me. This is an example of federated credential management. Therefore, in a federated credential management system the characteristic that comes to the forefront is trust among the different sites or entities sharing the credentials. Therefore, credentials are exchanged among the entities that trust each other. It is to be noted that the credential repositories and federated credential management systems are not competing technologies, rather they are complementary in nature.

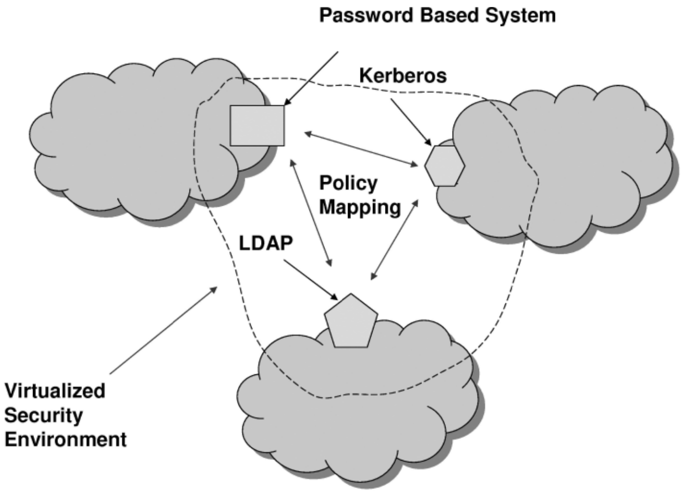


Fig. 9.2. A federated grid system

From the example mentioned above, it may seem that federated credential management may be useful for the identity management across different Web sites. However its applicability to grid computing may not be easily understood. Most of the grid systems that have been implemented employ centralized control of the resources. However, when enterprises come together to share resources, there may be a need to integrate the disparate authentication mechanisms, policies, etc. across the different enterprises. Let us take an example illustrated in Fig. 9.2. In this figure, there are three enterprises having different authentication mechanisms like Kerberos, password based, and X.509 certificate based. The enterprises have come together to share the resources, however they do not want to give away the control on the resources totally. Therefore, there is a need for integration of the policies across the different systems based on mutual understanding and trust and hence there is a need to manage heterogeneous credentials across the different systems. Some of the features that are needed in federated credential management systems are:

- **Repository of Heterogeneous Credentials:** Unlike the centralized credential management systems, a federated credential management system requires the storage of heterogeneous credentials.
- **Credential Transfer:** There is a need for transfer of credentials across the different systems based on trust in a secure manner.
- **Credential Translation:** There is a need for mapping between different credentials so that users can login and submit jobs with a different credential from the credential of the domain or the site executing the job.

9.2 Credential Repositories

The credential storage systems are designed so that the responsibility of storing the credentials securely is outsourced from the user to these systems, and the users can get the credentials anytime on demand.

9.2.1 Smart Cards

The smart card, an intelligent token, is a credit card sized plastic card embedded with an integrated circuit chip. It provides not only memory capacity, but computational capability as well. The self-containment of smart card makes it resistant to attack as it does not need to depend upon potentially vulnerable external resources. Because of this characteristic, smart cards are often used in different applications which require strong security protection and authentication. Smart cards can be thought of as a user managed portable credential storage, similar to a car key. The private key of the user can be embedded in the hardware of the smart card, so the only thing the user needs to worry about is the safety of the card itself. The smart cards therefore act as the identification card for the user storing the user's credentials, which can be distributed beforehand. More sophisticated cards mapping the user's biometric information in the card are also being developed. More details of the hardware and software architecture of the smartcards are available in [195].

Let us now try to analyze the applicability of smart cards in the grid scenario.

- **Security:** Smart cards are probably the safest way to store the user's credentials. There are two possible ways to compromise smart cards. Firstly, as all the key material of a smart card is stored in the electrically erasable programmable read only memory (EEPROM), and due to the fact that EEPROM write operations can be affected by unusual voltages and temperatures, information can be trapped by raising or dropping the supplied voltage to the microcontroller. In [196], several examples of attacking the smart card microcontroller by adjusting the voltage are provided. Secondly, the cards can be subjected to acid attack or the card can be stolen or destroyed. The second point thus raises an important concern. Since the users need to store the cards, and there may be multiple credentials associated with the user, therefore storing multiple smart cards increase the storage problem for the users. The problem of storing multiple credentials can be substantially reduced if the credential sharing techniques are employed in conjunction. Another way of addressing the problem is to go for virtual smart cards described later in this section.
- **Usability:** In addition to the users' woes of managing and storing multiple smart cards, there is a cost associated with smart cards. Though the cost of the smart cards has come down significantly, they still cost around $20 - 50 which may increase the significant overhead to the grid infrastructure.

9.2.2 Virtual Smart Cards

The term virtual smart card was first coined in [197] by Sandhu et al. They made a distinction between virtual smart card and a virtual soft token. In the former, the private key is never brought to the client system, similar to a smart card where the private key never leaves the card. In the case of virtual soft tokens, the user can retrieve the private key in any system of user's choice. The virtual soft tokens consist of two components, the key and the password, which are combined to produce the final encrypted version of the token. The simplest way to produce the encrypted key will be to encrypt the key with the user password, so that without knowing the password the key cannot be used. Unfortunately this scheme is susceptible to dictionary attacks. An attacker who has access to the encrypted private key can verify guesses for the password by decrypting the private key with the guess and verifying success or failure with respect to the known public key.

Therefore, the virtual smart card schemes combine the advantages of the smart cards and the online CAs.

- The virtual smart cards provide a *cheaper alternative* to the traditional smart cards. Since no infrastructural support is required, the deployment of such a technology in a grid computing scenario provides a cheaper alternative to the smart card solution. Moreover, since the user credentials are stored in a secure centralized place, user concern regarding credential compromise is reduced to a great extent. It also allows the user to store multiple credentials.
- Virtual smart cards are much more flexible than the credential management handled by online CAs. Virtual smart cards, at least theoretically, do not prohibit the storage of multiple credentials of different types. Therefore, Kerberos credentials along with X.509 certificates can be stored. Similarly, the repository may not be restricted to one CA, and manage credentials from multiple CAs.

9.2.3 MyProxy Online Credential Repository

MyProxy Toolkit [198] was developed in University of Illinois, Urbana Champagne (UIUC) and was developed to meet the credential management requirement of the grid community. It is a very popular grid credential management system. It has been used in major grids including NEESgrid, TeraGrid, EU DataGrid, and the NASA information power grid.

Figure 9.3 shows an overview of the MyProxy online credential repository system. It is based on the client server technology. The MyProxy system is the implementation of the Virtual Soft Token system proposed in [197], where the X.509 proxy certificates are used to store and retrieve user credentials without having to expose the private key. During the enrollment phase, the long lived user credentials are stored in the MyProxy repository, whose typical lifetime ranges from weeks to years. The users fetch the short term credentials or proxies (with lifetime set to a week or less) from the MyProxy server so that the long term credentials are safe. To achieve the above, the client establishes a TCP connection to the server and initiates the TLS handshake protocol as shown in the figure. The server must authenticate with the client using its own certificate, the client may also authenticate with the server. However, this step is optional for clients which do not possess X.509 credentials. According to [197], systems similar to MyProxy system are stronger than physical soft token systems, but they are vulnerable to dictionary attacks. The reason is that the private key is exposed to the server and can be compromised. However, it is to be noted that though such an attack is theoretically feasible, it can be greatly limited due to the use of SASL and hardware secured MyProxy solutions [199].

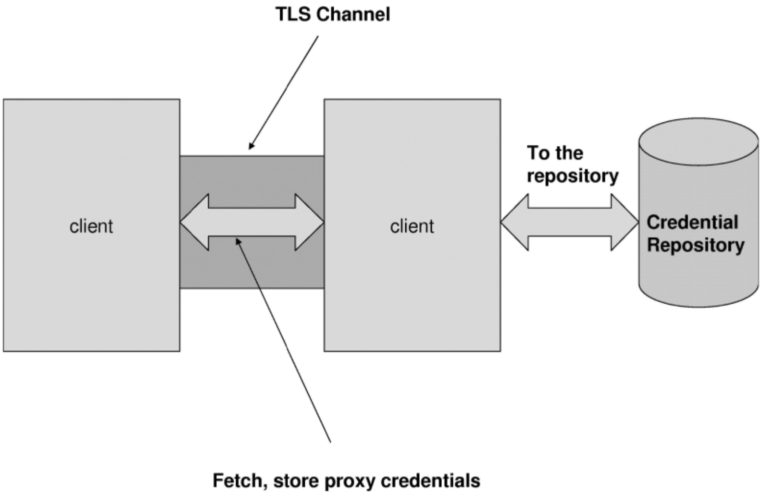


Fig. 9.3. Overview of the MyProxy system

Following are some of the important features of the MyProxy credential manager:

- **Proxy Certificates:** Proxy certificates are used by the MyProxy clients to retrieve credentials from the repository without having to export the private keys from the repository. Proxy credentials are derived from X.509 end entity certificates, and signed by corresponding end entity private key, to provide restricted proxy and delegation. The proxy credentials represent short term credentials which are derived from the long term credentials like the private keys. In the Globus system, the proxy credentials are stored unencrypted in the user's end system, to be used for single sign on purposes. MyProxy credential repository can be used to retrieve the proxy credentials at the start of the session.

- **Delegation:** MyProxy credential repository allows the process of delegation. This can be done through the proxy certificates. To delegate responsibility to some other host, the client generates a public/private key pair and sends a certificate request to the server. The server signs the request with the private key and sends the certificate chain back to the client. The server also sets the validity period of the new proxy certificates as much less than the original credentials. The client can also use the MyProxy credential repository system to store the delegated proxy credentials.
- **Access Control:** MyProxy allows the server administrators to have different access control mechanisms to control the retrieval and storage of the credentials. MyProxy allows the integration with SASL, allowing MyProxy to use local site authentication mechanisms. Therefore, policies can be set so that users with Kerberos tickets can access the MyProxy credentials without having to enter any other password.
- **Secure Storage:** MyProxy server encrypts the private keys using the user-chosen password by Triple DES in CBC mode. MyProxy does not store the password in the repository, as this would help the adversary to get the password in case of server compromise. Rather, the MyProxy uses the password sent by the client to sign the proxy certificate. Figure 9.4 illustrates this process.

In [199], the authors have proposed a hardware secured MyProxy solution, where cryptographic co-processors are used for secret storage of private keys. The co-processor (IBM® 4758 cryptographic co-processor) not only protects the user's private key from the adversaries, but also prevents the access of those keys by the administrators. The advantages of such a mechanism are three-fold:

- Since the keys are stored and generated only in the cryptographic co-processor, therefore neither the end-user nor the administrator can retrieve the key resulting in strong security.
- The keys can be generated faster due to the use of hardware random number generator.

- Since a cryptographic co-processor is used, the host does not require extensive security. Hence, machines of lesser configurations can be used.

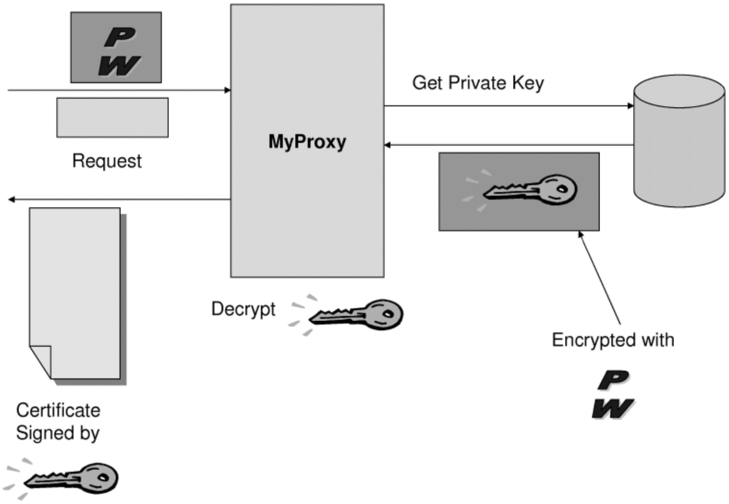


Fig. 9.4. Overview of the MyProxy storage

9.3 Federated Credential Management Systems

Let us now discuss some federated credential management systems which are currently available. They can be broadly categorized into two main types: specific and generic. The specific solutions aim at creating a federated solution for a specific platform or protocol. Examples of such systems are Virtualized Credential Manager (VCMan) and KX.509. The former tries to solve the inter-operability issue of CAS (refer to Chap. 5) by extending CAS to provide inter-operability, while KX.509 is a protocol of inter-operability between X.509 and Kerberos credentials. Though these solutions solve a niche problem, however the generic problem needs to be tackled as well. Liberty framework from the Liberty Alliance is an attempt in that direction. It is a framework for sharing attribute information in a distributed manner across entities in a trusted domain called the Circle of Trust (COT). Another solution to tackle the problem is Shibboleth.

9.3.1 Virtualized Credential Manager (VCMan)

VCMan [200] is an architecture to provide inter-domain virtualized credential and policy management. A high-level view of the VCMan architecture is provided in Fig. 9.5.

VCMan consists of three parts: Local Policy Manager (LPMan), Inter-domain Policy Manager (IPMan), and Inter-domain Credential Manager (ICMan).

Local Policy Manager (LPMan)

LPMan maps the policies between the users/resources and the Community Authorization Service (CAS). This functionality can be provided using a static or a dynamic mechanism. Currently the CAS implementation of the Globus Toolkit provides static functionality. Please refer to the CAS description in Chap. 5 for more details.

Inter-domain Policy Manager (IPMan)

This functionality allows the mapping of the policies across different domains. This mapping facility may involve a handshake mechanism between domains for policy-level understanding.

- ***Mapping of Policies:*** This part of the IPMan policy manager maps the policies of different domains. Each CAS server of the domain exposes the security policies of the domain based on the WS-Policy specifications (see the appendix). The policies are stored in local database of each CAS representing a domain. Domains have flexibility in assigning policies. For example, a domain D1 may decide on having a user based policy while a domain D2 may decide on having a role based policy model. In a user based policy model, a user is directly mapped to a resource, while in a role based policy model, the user is mapped to a role which in turn is mapped to a resource. For inter-domain policies, a particular domain may also employ user based, role based on some other customized policy model. A domain D1 may specify that certain user X from Domain D2 has read access to resource R. Otherwise; it can specify that only administrators (some role) of domain D2 can access the resource. In addition, it can also specify that all domains having X.509 based authentication mechanism can access resource R only in read only mode.

- ***Policy Exchange:*** When the CAS of domain D2 (CAS2) comes to the CAS of another domain D1 (CAS1), policy exchange between the domains takes place.

Inter-domain Credential Manager (ICMan)

Once the policies have been exchanged and established between the different domains through the respective CAS-es, the actual mapping and management of credentials are required. The user is oblivious of the actual authentication mechanism that is being used. The ICMan performs a two-pronged action: (i) Mapping the credentials of the current domain to that of the remote domain. This action is carried out at the CAS level. (ii) Managing the credentials. This includes regeneration of credentials once they expire.

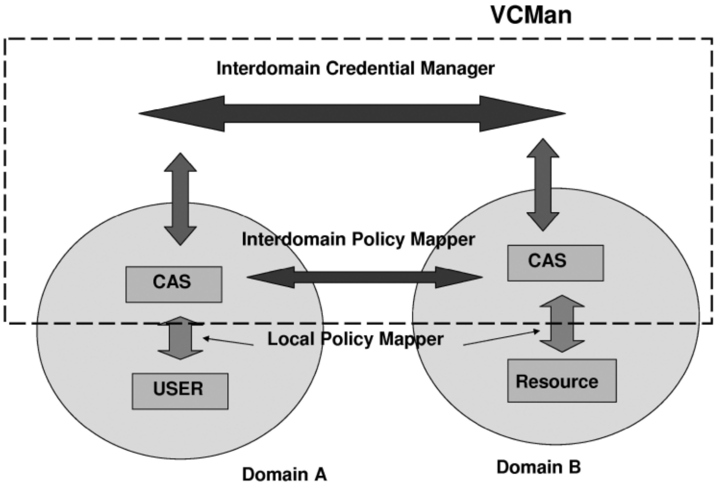


Fig. 9.5. Overview of the VCMan credential manager

9.3.2 KX.509

KX.509 is a protocol which allows the workstations to acquire a temporary X.509 certificate on behalf of a user based on the possession of the Kerberos ticket of the user. The advantage of the KX.509 protocol is that it allows users to access applications or resources which accept X.509 certificate, though the user authenticates using the Kerberos system.

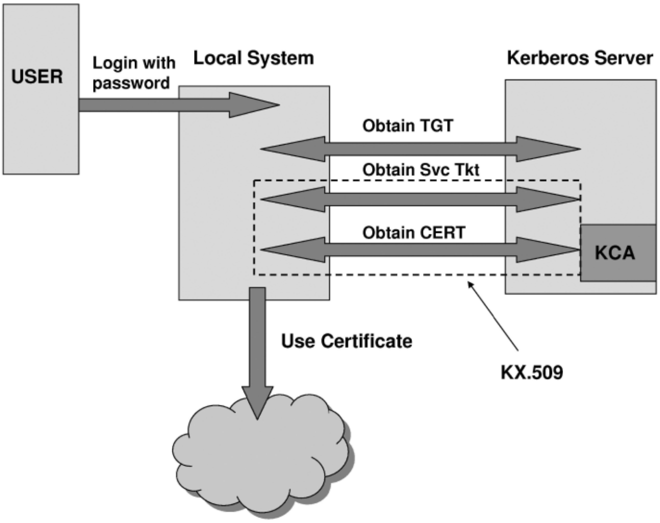


Fig. 9.6. Overview of KX.509 system

Figure 9.6 shows the overview of the KX.509 system. In the KX.509 system, the user logs in to the local system using the password. The local system obtains the Ticket Granting Ticket (TGT) from the Kerberos server. Once the TGT is obtained, the KX.509 protocol starts. KX.509 has the following steps:

- KX.509 generates the public/private key pair.
- KX.509 obtains the Kerberos service ticket for the Kerberized Certificate Authority (KCA) and sends the public half of the key pair.
- The KCA decrypts the service ticket, and uses the session key to send back a short lived certificate which can be used to access the resources requiring the X.509 certificate.

9.3.3 Liberty Alliance for Federated Identity

The Liberty Alliance [201] is a world-wide consortium of 150 companies who have come together to create the framework of federated identity. The solution put forward by the consortium is a framework to create federated identity. The framework provides mechanism standards for sharing attribute information across a trusted set of entities called the Circle of Trust (COT). The framework defines two types of entities: The Service Providers (SP) and the Identity Providers (IP). SPs are responsible for providing some service to the user, while identity providers are responsible for authenticating the users. One user may have multiple identities with different SPs. Once the user is authenticated by an IP, the user can access any service from the SPs and the sessions will be transferred between the SPs. It is to be noted that there is no globally unique identity for the user. The user may possess different identities with different SPs as long as they are linked through the Circle of Trust constraint. The Liberty Alliance architecture is illustrated in Fig. 9.7.

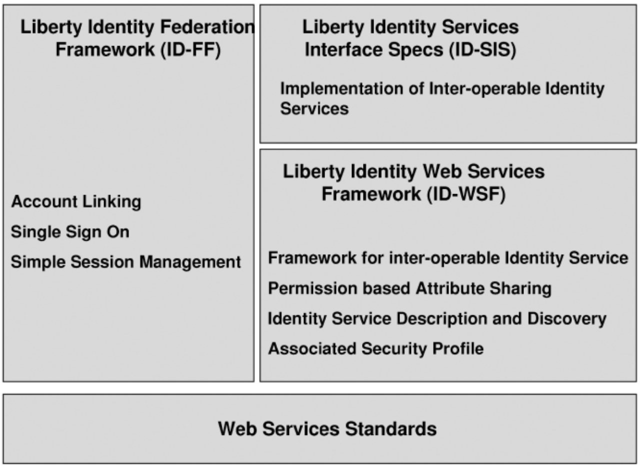

Liberty Identity Federation Framework (ID-FF)
Account Linking
Single Sign On
Simple Session Management
Liberty Identity Services Interface Specs (ID-SIS)
Implementation of Inter-operable Identity Services
Liberty Identity Web Services Framework (ID-WSF)
Framework for inter-operable Identity Service
Permission based Attribute Sharing
Identity Service Description and Discovery
Associated Security Profile
Web Services Standards

Fig. 9.7. Architecture of the Liberty Alliance

Liberty Alliance Specifications

The technology and standards work at the Liberty Alliance has focused on three sets of specifications, *viz.* ID-FF, ID-WSF, and ID-SIS frameworks.

- **Identity Federation Framework (ID-FF):** This framework mainly handles the account federation and single sign on aspects. The different components of the ID-FF framework include account linking, where multiple identities across different SPs or sites are linked together. It also includes single sign on and session management components which allow the user to seamlessly sign-on to Liberty enabled sites and manage sessions across sites respectively. Components to maintain user anonymity and real-time exchange of meta-data also form part of the ID-FF framework.
- **Identity Web Services Framework (ID-WSF):** This framework handles the Web services discovery and transactions between Web services based on identity. For example, a user A may be entitled to more information regarding a Web service than user B based on their identities. Specifically it includes permission based attribute sharing, identity based service discovery, security profiles, identity services template, SOAP based binding, and others.
- **Identity Services interface Specification (ID-SIS):** This is a collection of specifications for inter-operable services to be built on top of the Liberty's ID-WSF. These might include services such as registration, contact book, calendar, geo-location, presence, or alerts.

9.3.4 Shibboleth Identity Federation

Shibboleth [201] is a federated identity management system based on open source software developed by the Internet2 consortium members, with assistance from the National Science Foundation. Internet2 is a consortium of US universities working in partnership with industry and government to develop and deploy advanced network applications and technologies. Shibboleth is essentially a transport mechanism built on top of an institution's existing architecture that allows organizations to exchange information about their users in a secure and privacy-preserving manner. The purpose of the exchange is typically to determine if a person using a web browser has the permissions to access content or a service from a content provider based on information such as being a member of an institution or a particular class. The system preserves privacy in that it leads with this information, not with the identity of the user, and allows users (or their institution) to determine whether to provide extra information about themselves. Shibboleth has been developed as an open architecture and as an open source implementation; it is standards-based so that information that is exchanged between organizations can interoperate with that from other solutions. For the purpose of inter-operability, Shibboleth uses SAML and SOAP. The basic concepts contained within Shibboleth include:

- **Framework based on Clubs:** Shibboleth uses *clubs* to specify a set of parties who have agreed to a common set of policies. Sites can choose to join multiple clubs. This moves the trust framework beyond bi-lateral agreements, while providing flexibility when different situations require different policy sets.
- **Federated Administration:** The origin site provides attributed assertions about the user to the target site. Based on the trust relationships between the origin and target sites, user can be identified. Origin sites are responsible for authenticating their users using any authentication mechanisms.
- **Access Control**: Access control decisions are made using those assertions. The collection of assertions might also include identity assertions.

As can be surmised from the discussions above, the purpose of both Shibboleth and Liberty framework are the same, to provide federated credential management. While Shibboleth is an academic project aiming at creating a federated identity spanning a few hundred institutions, Liberty framework aims at a management of tens of millions of accounts spanning over a million enterprises. Given the problems each system typically attempts to solve, the main difference lies in the way the credentials are accessed and stored. Shibboleth assumes that the credentials or assertions are stored and provided by the origin sites. Liberty framework makes no such assumptions as the credentials may be distributed in nature. Another significant difference is that Liberty framework works on any wireless devices and specifications are provided on how the different wireless protocols and devices can be leveraged. However, it is to be noted that the flexibility provided by the Liberty framework comes at a cost of added complexity and concerns about the credential privacy due to its distributed nature. Since open source implementation of Shibboleth exists, many systems (even commercial) have used it for federated identity management. We will provide an example of GridShib where Shibboleth is integrated with the Globus toolkit.

Integration of Globus with Shibboleth

The management of federated identity is very useful in grid context. Therefore, several researchers have looked at integrating Globus with the Shibboleth system. One such initiative is informally called GridShib [202], whose objective is to provide mechanisms whereby a Grid service can obtain from the Shibboleth service the specific user attributes that it is authorized to obtain. There are several scenarios where such a system could potentially be used. One such scenario is where the user tries to access a shared grid resource gets authenticated using GSI in its own organization or campus, and the attributes are then pulled from the origin campus. The GSI-Shibboleth integration consists of five main components:

- **Assertion Transmission**: Transmission of assertions is required from the Shibboleth service to the Grid software and ultimately to the Grid runtime authorization decision-making component.
- **Attribute Authority:** Since Grid resources serve users from multiple organizations, a mechanism is needed to determine which organization's Shibboleth service is authoritative for a particular user.
- **Distributed Attribute Administration**: Sometimes it becomes necessary for projects or systems outside the domain to operate the Shibboleth service on its portion of the attribute space.
- **Pseudonymous Interaction**: This helps in extending to Grids the pseudonymous interaction provided by Shibboleth.
- **Authorization**: A mechanism is needed in Globus which can take advantage of the attributes.

9.4 Chapter Summary

Credentials are important in grid systems as they are used for accessing the Grid resources. Therefore, there are needs for mechanisms to securely store, access, and manage credentials in grid systems. Credential Management (CM) systems are precisely meant for this purpose. Credential management systems can be divided into two main categories: credential repositories and credential federation systems. As the name suggests, the credential repositories or credential storage systems are concerned about securely storing the credentials, generating new credentials on demand, and sometimes generating proxy credentials on user's behalf for delegation purposes. Credential federation systems or credential share systems are responsible for sharing the credentials across different domains or realms.

Examples of credential storage systems are smartcards, MyProxy etc. and examples of credential share systems are the Liberty Project, VCMan, etc.

Table 9.1. Summarizing the different schemes

Scheme	Type	Identity	Credential Type	Security	Comments
Smart Cards	Repository	Single	Keys	Very Secure	Cost and usability may be limiting
Virtual Smart Cards	Repository	Single	Keys	Very Secure	Very secure Can work on top of repositories like MyProxy
MyProxy	Repository	Multiple	Keys	Secure	Susceptible to dictionary attacks Requires mult credential support
VCMan	Federation mechanism	X.509 and Kerberos	Tokens	Secure	Requires CAS Only support X.509 and Kerberos
KX.509	Federation Protocol	X.509 andKerberos	Tokens	Secure	Limited to X.509 and Kerberos
Liberty Alliance	Federation Framework	Multiple	Multiple	Secure	Extensive framework
Shibboleth	Federation System	Multiple	Multiple	Secure	Framework and Open source implementation

Table 9.1 summarizes the different schemes. As can be observed from the table, smart cards are very secure; however cost can be a hindrance in its widespread adoption. Research and development efforts are needed for creating virtual smart card technologies on top of existing technologies like MyProxy. MyProxy credential manager, though a good effort, is susceptible to dictionary attacks. VCMan and KX.509 have limited use at this moment as they only support X.509 and Kerberos. VCMan also requires CAS support. The Liberty framework is an important industry effort to create a common framework for identity management and research is needed for integrating that framework with existing grid based systems.

10 Managing Trust in the Grid

10.1 Introduction

In our everyday life, we come across different types of people, situations, events, and environments. The interactions between the individuals depend on implicit understanding of relationship across society. This implicit understanding between individuals is based on the confidence that the individuals can gather and emanate during the relationships, the personal and professional connections bonding the individuals, societies, cultures, and a host of other factors. When I meet a new car mechanic, I feel confident to give him the responsibilities of repairing my car based on my personal interactions or hunch about his abilities, his credentials, the company he represents, and a host of other factors. In other words, my decision to ask for his help depends upon amount of *trust* I can put to the claims that he is making. Trust pervades through different strata of human society and extends beyond the human-human relationships. For example, trust can act as a means of developing relationships between electronic gadgets like computers with human beings. Trust is a complicated concept, and the ability to generate, understand, and build relationships based on trust varies from individual to individual, situations to situations, society to society, and environment to environment. For example, I may trust the car mechanic to repair my car. However, I may not have enough trust in him repairing my computer and so on. Each individual, in general, carry some amount of prejudices based on past experiences or history which are generally used to determine the trustworthiness of a person the individual is interacting with. In this chapter we will discuss the different aspects of managing trust in grid systems.

10.1.1 Definition of Trust

Trust is a fascinating subject, and social scientists have researched into the concept and developed theories around it. One of the most popular definitions of trust was coined by Deutsch [204] which states that, "*(a) an individual is confronted with an ambiguous path, a path that can lead to an event perceived to be beneficial or to an event perceived to be harmful; (b) he perceives that the occurrence of these events is contingent on the behavior of another person; and (c) he perceives the strength of a harmful event to be greater than the strength of a beneficial event. If he chooses to take an ambiguous path with such properties, he makes a trusting choice; else he makes a distrustful choice*." This definition was further extended again by Deutsch later in 1973 [205], where he defined trust as *confidence that an individual will find, what is desired from another rather than what is feared.* The coupling of confidence and trust finds its way in the definition of Webster dictionary as a *confident dependence on the character, ability, strength, or truth of something or someone.*

All these definitions of trust fall into the domain of social settings and social interactions. However, in the domain of digital systems also the concept of trust is becoming more and more relevant. This is because of the increased importance of distributed and autonomous systems. There is a need for developing trust among the different systems which interact among each other. Therefore, we are witnessing a renewed interest of research related to trust in distributed systems. One of the most popular definitions of trust that has also been adopted by computer scientists is the one coined by Diego Gambetta [206]. He defines trust as "*a particular level of the subjective probability with which an agent assesses that another agent or group of agents will perform a particular action, both before he can monitor such action (or independently of his capacity ever to be able to monitor it) and in a context in which it affects his own action*." Gambetta introduced the concept of using values for trust and also defended the existence of competition among cooperating agents. A recent definition of trust has been put forth by Grandison and Sloman [207] who define trust as "*the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context*."

As the above definitions show, trust is a multidimensional quantity, and can be viewed as conditionally transitive [208]. By this we mean, that if there are three individuals A, B, and C, and if individual A trusts B, and B trusts C, then A will trust C based on certain policies and conditions. Generally, researchers have defined trust graphs to determine the trust relationships between different entities. We provide more details about trust graphs and relationships in subsequent sections. In addition to the term trust, there are several concepts and terms which are used. They are:

- **Trustworthiness** An entity's trustworthiness is an indicator of the quality of the entity's services. It is often used to predict the future behavior of the entity. Intuitively, if an entity is trustworthy, it is likely that the entity will provide good services in future transactions. In most trust models, the domain of trustworthiness is assumed to be a [0,1] model. However, trustworthiness can be a continuous function also. It just increases the complexity of the system manifold.
- **Reputation:** Reputation indicates the general perception of a system's trustworthiness as perceived by other entities. We have provided more detailed description and its relationship with trust in the subsequent section.
- **Feedback:** A piece of feedback is a statement issued by the client about the quality of a service provided by a server in a single transaction. In general, feedback may be multidimensional, reflecting the client's evaluation on a variety of aspects of a service, e.g., price, product quality, and timeliness of delivery. Many of the times, for simplicity, feedback is assumed to be one-dimensional in nature.
- **Opinion:** An opinion is a user's general impression about a server. It is derived from its feedback on all the transactions that are conducted with the server. Similar to feedback, opinion also can be multidimensional in nature.

10.1.2 Reputation and Trust

Closely related to trust is the concept called *reputation*. Reputation as defined in Merriam-Webster is "*Overall quality or character as seen or judged by people in general.*" The concepts of reputation and trust are therefore closely related. Reputation is the metric generally used by individuals based on word-of-mouth, or past history to determine the trustworthiness of a person or things. The inter-relation between trust and reputation is widely exploited in day-to-day activities. Students looking for higher educations rely on institute rankings to determine the quality of the institute, or in other words the rankings of the institute determine the trust the students can put on the quality of education the institute will provide. In this case, the reputation is determined by the rank of the institute. However, this is not at all straightforward in all cases. When I read an article on the Internet, I base my judgment on a host of different criteria: the credentials of the authors who had written the article, the conference or journal where the article has been published, and so on. Therefore, determining the reputation of some system or individual may be based on a host of metrics and policies, and is a challenging research topic. Computer scientists have started to take a considerable amount of interest on the reputation of systems especially for distributed systems and applications. Abdul-Rehman et al. [209] defines reputation as "*an expectation about an individual's behavior based on information about or observations of its past behavior.*" In online communities, where an individual may have much less information to determine the trustworthiness of others, their reputation information is typically used to determine the extent to which they can be trusted. Similarly, in a distributed grid environment, where the grid nodes join or leave the grid, there the reputation information can be used to determine the trustworthiness of the nodes.

10.1.3 Categories of Trust Functions

Defining trust metrics is important, and researchers have developed trust functions for distributed systems. It is to be noted that most of the research done in this area are related to the fields of e-commerce and peer-to-peer computing. However, most of the concepts apply generally to distributed systems and therefore can be adapted to grid computing also. The following categorizes the trust functions in different categories [210].

Subjective vs. Objective

An entity's trustworthiness is often related to the quality of services it provides to others. If the quality of a service can be objectively measured, then an entity's trustworthiness for that service is called *objective trust*. For example, suppose a website provides specification information of automobiles. The quality (or accuracy) of such information can be easily and indisputably checked against the official data released by manufacturers. For some other services, their quality cannot be objectively measured. For example, given a movie review from a website, different people may have different opinions about its quality. It largely depends on each individual's taste and other subjective factors. In this situation, it is only meaningful to discuss the trustworthiness of the entity from the specific source's point of view. This type of trust can be classified as *subjective trust*.

Now the question arises about the relevance of these definitions in the realm of grid computing. The reason is that a grid computing node would rarely review a movie and provide subjective opinions. However, the definitions are still relevant. Let us take the following example. There are four grid nodes A, B, C, and D. The nodes A, B, and C try to make create a trust decision regarding D based on the information collected from D in a periodic manner. Though all the three nodes are collecting the same information, they are constrained by the times at which the samples are taken and hence the decisions made by the nodes are highly subjective.

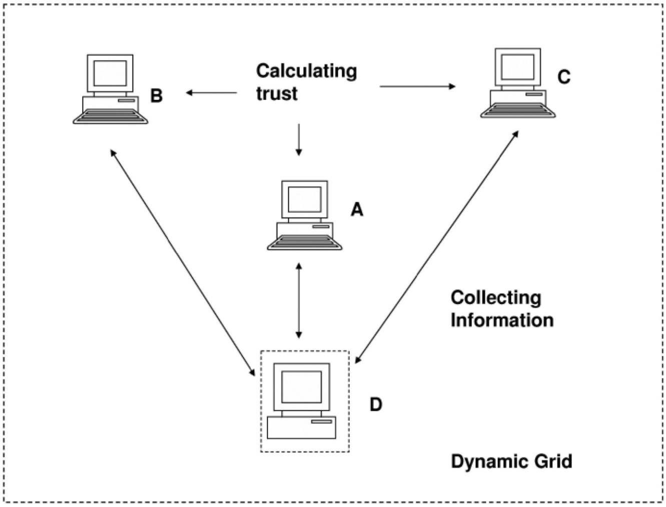


Fig. 10.1. Example of trust in a dynamic grid

Transaction based vs. Opinion Based

Some trust models rely on the information of *individual transactions* to infer an entity's trustworthiness, while others only request opinion information. The former is called transaction based, while the latter is called *opinion based*. In the grid system mentioned in Fig. 10.1, information can be gathered based on number of successful transactions (transaction-based) or based on the opinions of the other nodes in the grid (opinion-based).

Complete vs. Localized Information

Trust functions can also be classified according to the way information is collected. Some trust functions [211, 212] assume that every entity has the same access to all the transaction or opinion information. In other words, to apply a trust function, a complete transaction or opinion graph is a must. Such trust functions can be classified as *global trust functions*. Another approach is to adopt a localized search process. Typically, it is assumed that an entity has several neighbors, who may or may not have interactions with the entity before. If Alice wants to evaluate Bob's trustworthiness, she will broadcast to her neighbors the requests for Bob's transaction/opinion information. This process continues until her neighbors have returned sufficient information for Alice to make a trust decision. To achieve better performance, information collection is usually a controlled flooding process. Therefore, the trust function is applied on a subgraph of the complete trust graph. Since each entity chooses their neighbors freely, different trust evaluation sources may construct different subgraphs. Trust functions of this kind can be classified as *localized trust functions*. Intuitively, for a localized trust function, each entity typically has access to different information. A localized trust function is thus also subjective [213].

Rank based vs. Threshold based

For most trust functions, its returned trustworthiness can be interpreted as an approximation of some of the properties of a system. For example, if the trustworthiness of an automobile website is 0.8, we may think that approximately 80% of the information provided by the website is accurate. For such trust functions, it is appropriate to predefine a threshold of trustworthiness to make trust decisions. For example, if a website's trustworthiness is over 0.9, then we trust information from that website. Thus, such functions can be categorized as *threshold-based*. In some other trust functions, the calculated trustworthiness of a single entity alone does not convey much information. It becomes meaningful only when it is compared with the trustworthiness of other entities. In some sense, such trust functions return the relative ranking of an entity. Such functions are called *rank-based functions*.

Characteristics of different trust functions are summarized and tabulated in Table 10.1. As can be inferred from the table, subjective, transaction based, and global functions are less vulnerable to malicious updates.

Table 10.1. Characteristics of different trust functions

Type of Trust Fn	Message Overhead	Storage Overhead	Security	Comments
Subjective & Objective	Can vary	Can vary	Subjective functions are vulnerable to malicious updates compared to the Objective functions	Subjective functions should be used only in cases where objective information is unavailable
Transaction based & Opinion Based	High	High	Transaction based functions are less vulnerable than Opinion based functions	Transaction based provide more accurate information with more overhead
Global & Local	High	High	Global functions are less Vulnerable than local functions	Global trust provides high accuracy with high overhead
Rank based & Threshold Based	Can vary but generally high	Can vary but generally high	Similar in security performance	When inference is based on relative performance a rank based function is better

10.2 Trust Management Systems

Till now, we have talked about trust functions and reputations. Now, let us concentrate on Trust Management Systems (TMS) responsible for managing trust in a distributed environment. TMS systems can be divided into two main types: policy-based TMS and reputation-based TMS.

- **Policy-based TMS:** In policy-based systems, the different entities or components constituting the system, exchange and manage credentials to establish the trust relationships which are further refined based on certain predefined policies. The primary goal of such systems is to enable access control by verifying credentials and restricting access to credentials based predefined policies. Different credential management systems described in Chap. 9 fall under this category. Since we have dealt with this type of systems in detail in Chapter 9, in this chapter we would mostly restrict ourselves to the reputation-based systems only. We will however, discuss about the policy based TrustBuilder system.

- **Reputation-based TMS:** The second category of TMS systems, or the reputation-based TMS, provides a mechanism by which a system requesting a resource evaluates the trust of the system providing the resource. The trust values can be a function of the global and local reputation of the systems along with the different policies. In this chapter, we will describe a few reputation-based TMS systems in detail.

Management of trust within a TMS system includes negotiating of trust when a new member joins the distributed environment, storage of the trust metrics, and distribution of trust metrics. More details about the life cycle of the TMS system is provided in the next subsection.

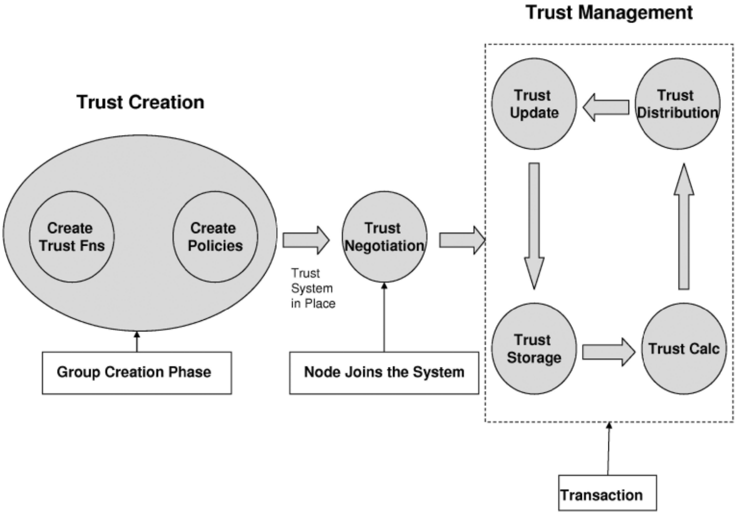


Fig. 10.2. Life cycle of a trust management system

10.2.1 Life Cycle of Trust Management Systems

Figure 10.2 shows the event view of the life cycle of a trust management system. The life cycle is composed of mainly three different phases: *trust creation phase*, *trust negotiation phase*, and *trust management phase*. The trust creation phase generally is done before any trusted group is formed, and it includes mechanisms to develop trust functions and trust policies. Trust negotiation, on the other hand, is activated when a new untrusted system joins the current distributed system or group. The third phase, or the trust management phase, is responsible for recalculating the trust values based on the transaction information, distribution or exchange of trust related information, updating and storing the trust information in a centralized or in a distributed manner.

Trust Creation Phase

This phase actually takes place before any transactions and is responsible for setting up the trust functions and the policies that will be used by the trust management system. The first step in the trust creation phase is to determine the type of the trust management system, whether it would be policy-based or it would be reputation-based. Once the type of TMS system is decided; the policies are defined and created. The next important step in this phase is to determine the trust function. As mentioned earlier, trust functions can be of several categories: objective or subjective, transaction-based or opinion-based, complete or localized, and threshold-based or rank-based. The choice of a trust function depends on the type of application the TMS is catering to. A high level comparison between the different trust functions are provided in Table 10.1.

Trust Negotiation Phase

The second phase in the life cycle of the TMS is the trust negotiation phase which is started when a new entity or node joins the system. Figure 10.3 shows a very high level overview of any trust negotiation phase. The phase essentially consists of three different steps: request, policy exchange, and credential exchange. At the heart of the trust negotiation lie the policies and the policy language acceptable to both the parties. Several policy languages have been developed as part of the distributed trust management solutions [214]. The different steps in the trust negotiation phase are:

- **Request:** This step identifies the client and the type of service the client wants from the system. This step can succeed a key establishment phase, where the session key can be established by the two parties.

- **Policy Exchange:** This step exchanges the policies between the new entity and the system. The policies can be expressed in policy languages like the PeerTrust [215]. At this step, the trust computation of the new node can also be evaluated based on the system's trust function.
- **Credential Exchange:** In this phase, secure exchange of credentials like keys, certificates, etc. take place. Proper security measures need to be taken to ensure the secure exchange of credentials.

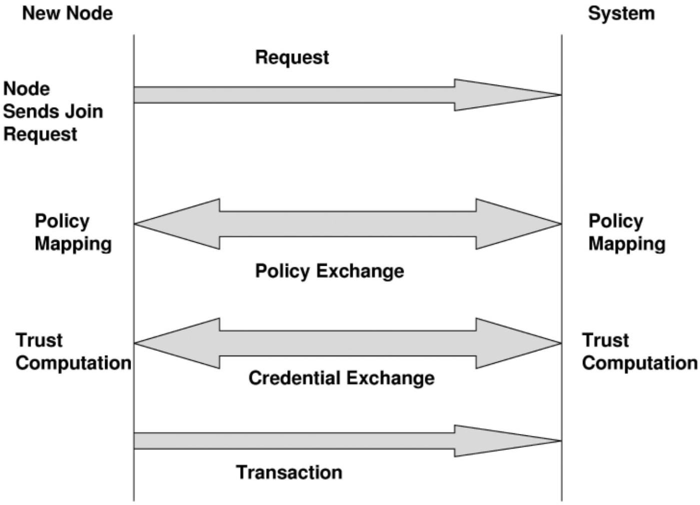


Fig. 10.3. Typical trust negotiation phase

Trust Management Phase

After the trust negotiation phase comes the trust management phase which is concerned with the general running of the distributed system. The different activities that are part of this phase are:

- **Trust Computation:** In this step, the trust value is computed based on the decided trust function.

- **Trust Distribution:** This step includes the secure distribution of trust information to other nodes in the distributed system. Since a secure distribution is a necessity, all the principles of security, *viz.* confidentiality, authentication, integrity, and non-repudiation need to be maintained. This step also requires keeping in mind the type of trust function in use and the number of nodes where the information needs to be broadcasted.
- **Trust Storage:** The trust information needs to be securely stored. The credential repositories like the MyProxy can be used for this purpose.
- **Trust Update:** Updating the trust needs to be carried out either in an event by event basis or in a timely manner. Event based trust update can happen after a set of transactions or when the trust value or opinion crosses a threshold.

10.2.2 Characteristics of Trust Management Systems

In this section, we will talk about some key characteristics of a trust management system.

Scalability

When we talk about distributed systems, scalability assumes paramount importance. Same holds true for TMS on a grid also. The scalability of a TMS refers to the ability of the system to scale with the increasing number of nodes or entities in the system. The system should scale in terms of messages which are exchanged between the nodes, the information stored at each node, and the computation carried out at each node.

- **Message Overhead:** This indicates the number of messages and the sizes of messages that are exchanged to create and manage the trust information. High message overhead indicates a high bandwidth cost. To ensure scalability, the message overhead should not increase significantly with the number of nodes in the system. The TMS systems which scale well do not exchange all the information to all the nodes in the system. Rather, they exchange with a select set of members or only transfer aggregated information.
- **Storage Overhead:** The TMS should also scale in terms of amount of information stored to compute the trust information. It is possible that a TMS system may require nodes to store trust

data about other nodes in the system. This trust data may be as detailed as each transaction detail that has taken place in that node. Depending upon the type of information stored, it is possible that the TMS system may require a significant amount of storage to make meaningful trust decisions. Therefore, storage considerations are also important in deciding the TMS systems.

- **Computational Overheads:** Sometimes the trust computations are too complex and compute-intensive for the nodes with less computing power to do. Therefore, when one is architecting a TMS system, one needs to keep the infrastructure in mind also.

Reliability

Reliability is another important characteristic that needs to be taken into account when designing a TMS system. Reliability can be achieved at the message level and the node level.

- **Message Level Reliability:** Here the information exchanged between the different nodes is reliable in nature. This can be achieved by two ways, spatial and temporal redundancy. In case of spatial redundancy, some extra information is put in the messages like CRC, hashes, etc. In case of temporal redundancy, positive or negative acknowledgments are used to achieve reliability. TCP at the transport layer can be used to achieve temporal redundancy.
- **Node Level Reliability:** One of the main characteristics of a decentralized system is its constantly changing topology. This is typically due to the transient nature of nodes constituting the topology. The nodes may enter, leave or be disconnected from the system at any time. Fault tolerance in this context represents the ability of the trust model to adapt to this transient nature of the system. When peers enter or leave the system, not only do new trust relationships need to be formed but trust values and transaction information may also need to be replicated across peers to ensure availability and reliability of trust data.

Security

Perhaps the most important characteristics of a TMS system are the security characteristics of the system. Designed TMS systems must understand the security characteristics and vulnerabilities associated with the system.

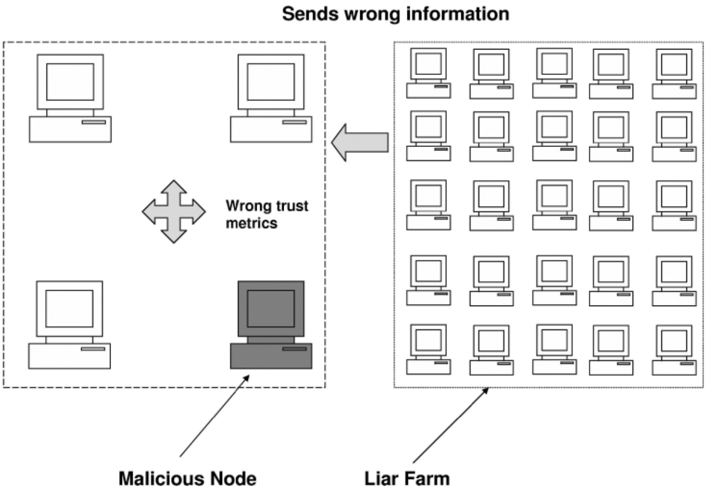


Fig. 10.4. Liar farms creating sybil attack

The security vulnerabilities that a TMS needs to take care are:

- **Fabrication/Modification:** The different nodes or entities of the TMS can fabricate or modify the trust data resulting in an artificial reduction or increase of trust values of different nodes. This may be dangerous for systems which build the trust based on information obtained from different sources.
- **Masquerade:** In this type of attack the adversary node impersonates or masquerades as someone else and sends information which is generally wrong. Some form of authentication through digital signatures or certificates eliminates this problem.
- **Anonymity:** In some cases anonymity can be an important security criterion [216]. Anonymity guarantees that the nodes remain anonymous to one another and are unaware of what the other node is doing. Several solutions, especially in the domain of peer-to-peer systems, are available which tackle anonymity [217, 218]. However, no anonymity solution has been applied in the context of TMS.

- **Collusion:** Until now, we have only considered a single malicious node in the system. However, if there are multiple malicious nodes working in harmony towards the same malicious purpose, the effect can be really dangerous. The nodes can collectively send bad information about one node in the system, or send very good reputation information about one of the malicious nodes, forcing the node to be used in most of the transactions.
- **Sybil Attack:** Collusion attack can be carried out by only a single node by using a sybil attack or liar farms. If a proper identity management system is not in place then a malicious node can assume multiple identities. Once multiple identities have been established, then the node and its shadows can launch deadly attacks, as they have the critical mass to actually change the trust values of the system. A typical example of launching a sybil attack is illustrated in Fig. 10.4. In the example, the malicious node assumes multiple identities and sends wrong information to the other nodes resulting in the creation of wrong trust metrics.

10.3 Reputation-Based Trust Management Systems

We have discussed different aspects of TMS systems. We will now discuss some TMS systems which have either been implemented or are in their advanced stage of research and development.

10.3.1 PeerTrust – A P2P Trust Management System

Perhaps the most widely known trust management system is the PeerTrust [219]. This was developed in Georgia Tech with Peer-to-Peer based electronic applications in mind.

Trust Metric

In PeerTrust, the trust metrics are derived from a combination of parameters:

- The feedback that a node or peer receives from other peers in terms of satisfaction. This value of satisfaction can be an average of the number of successful transactions over a period of time.
- The total number of transactions that the peer has with other peers which would indicate the nature of misbehavior of a peer.

- The factor of credibility for the source of the feedback. This indicates how much importance needs to be given to the information provided by the source.
- The transaction context, which discriminates between the mission critical or important transactions from that of mundane or noncritical ones.
- The community context, which takes into account the vulnerabilities and characteristics that can be associated with the community as a whole.

The metric consists of two parts. The first part is a weighted average of the amount of satisfaction a peer receives for each transaction. The weight takes into account the credibility of feedback source to counter dishonest feedback, and transaction context to capture the transaction-dependent characteristics. The second part of the metric adjusts the first part by an increase or decrease of the trust value based on community-specific characteristics and situations.

Architecture of PeerTrust

PeerTrust does not use a centralized database for storing the trust information. Rather, the trust information is stored in a distributed manner over the network. Each peer or a node in the network has a *trust manager* that is responsible for feedback submission and trust evaluation, a small database that stores a portion of the global trust data, and a *data locator* for placement and location of trust data over the network. The feedback submission and trust evaluation by the trust manager is performed in two steps:

1. The node manager submits feedback to the network with the help of the data locator. The data locator contains the information about the different information stored at different nodes or peers and uses this information to route the data to appropriate peers for storage.
2. Node manager is also responsible for evaluating the trustworthiness of a particular peer. This task is performed in two steps. It first collects trust data about the target peer from the network through the data locator and then computes the trust value using the trust function defined for the system.

The trust data can be distributed in the different nodes or peers based on any existing data management mechanisms like CHORD [22], CAN [23], P-Grid [220], Pastry [221], etc. which use the Distributed Hash Table

(DHT) mechanism of distributing the data across the network. Figure 10.5 shows the high level architecture of the PeerTrust system. An example is also shown, where the data is stored according to the P-Grid DHT architecture.

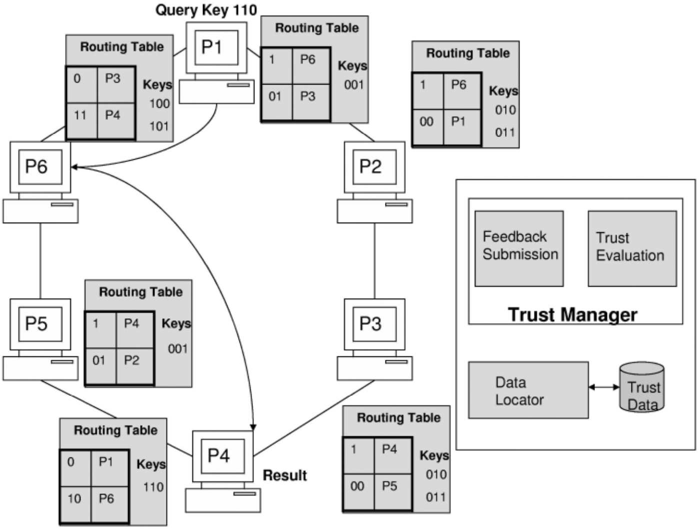


Fig. 10.5. Architecture of the PeerTrust system

Let us now discuss some of the characteristics of the PeerTrust trust management system. Specifically, we will talk about trust computation, dynamic peer management, security, reliability, and peer selection processes with the PeerTrust system.

- **Trust Computation:** The trust computation is based on the computation of trust data collected from different nodes or peers. PeerTrust develops two different mechanisms of trust computation and evaluation, *Dynamic Trust Computation (DTC)* and *Approximate Trust Computation (ATC)*. The first computes trust based on fresh trust data collected at runtime from different peers. The algorithm requires the peers to recursively evaluate the other peers' trust values as the credibility factor, in order to compute the different trust values. Therefore, the process is really expensive and cannot be used in practice, especially if the number of nodes or peers is significantly high. The ATC mechanism, on the other hand, computes trust metrics based on cached information and hence is much less expensive compared to the DTC scheme. The cached information removes the recursive computation of trust values.

- **Dynamic Peer Personality:** Since the PeerTrust mechanism is completely dependant on the dynamic calculation of trust values based on reputation as maintained by different peers, a malicious peer having very good reputation can start behaving maliciously without reducing its trust value significantly. To prevent this type of dynamic behavior, a window is used over which the trust computation takes place. This gives more importance to the latest transactions.
- **Security and Reliability:** The PeerTrust system uses a Public Key Infrastructure (PKI) based security mechanism for secure exchange of trust data. Each peer, when they are sending, signs the trust information so that the authenticity of the trust update can be verified by the receiving peer. The information exchanged are also encrypted, preventing a malicious node from getting sense out of the routed packet. Reliability is achieved by replicating the key information in different nodes. A peer-to-peer based distributed architecture for data storing scales well. One of the comments one can make about this architecture is the dependence on the PKI architecture and lack of trust negotiation mechanism.
- **Peer Selection:** A key objective of the trust based peer selection scheme is to select a set of peers that can perform a set of tasks in a most trusted manner. To achieve this, PeerTrust employs a threshold-based peer selection mechanisms. Rules or policies can be created where a trusted peer can be defined. For example, a policy can be defined where a peer is trusted if the trust metric is greater than a threshold.

10.3.2 XenoTrust Trust Management System

Another interesting project that develops a distributed trust and reputation management architecture is called XenoTrust [222] which is built on the XenoServer Open Platform [223]. The platform was developed at the University of Cambridge. The platform consists of three main components: XenoServer, XenoCorp, and XenoServer Information Services (XIS). XenoServers provide services to the client like hosting the client tasks in exchange of money. XenoCorp provides authentication, auditing, charging, and payment services. Each XenoServer periodically reports its status and the XIS is used for storing the XenoServer status updates. A high level overview of the XenoServer is provided in Fig. 10.6.

The XenoTrust architecture introduces two levels approach of managing trust, called *authoritative* and *reputation based*. *Authoritative trust* is a boolean property established between a XenoCorp and the clients and servers that register with it. *Reputation-based trust,* on the other hand, is a discrete continuous property which quantifies, in a particular setting, the trustworthiness that one component ascribes to another. It is distributed and highly subjective, in the sense that each entity has its own, independent view of others' reputations. Let us now discuss some of the aspects of the XenoTrust Trust Management System.

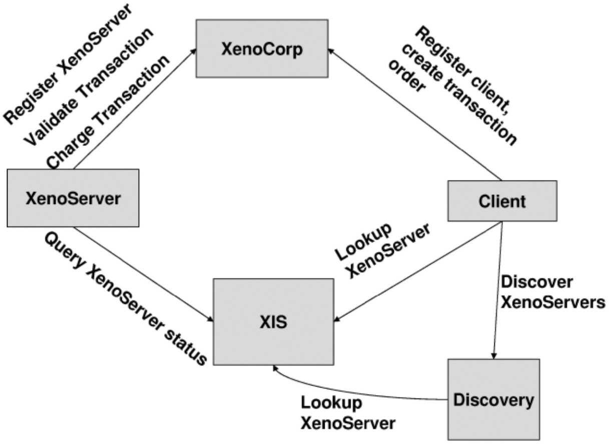


Fig. 10.6. Overview of XenoServer open platform

Reputation Representation and Exchange

Reputation values of different components are stored in the XenoTrust system in a centralized manner. The reputation information are stored in the form of a vector which indicates the individual experiences of a particular peer with other peers in the system. Different components of behavior like performance, honesty, etc. are reflected in the reputation vector through fields called tokens. The different peers or components of the system exchange *statements* with the XenoTrust system which is nothing but a tuple containing information about the advertiser, subject, token, values, and the timestamp. To maintain authenticity the information is signed.

Reputation Retrieval

The computation of reputation vectors is done at the XenoTrust itself through aggregation of reputation information. The XenoTrust system supports complex rulesets for combining the results. For information retrieval from XenoTrust, in addition to employing a query based pull mechanism where the components or peers specifically ask for information from the Xenotrust system, XenoTrust also employs a push-based architecture. When participants of the XenoTrust system deploy rule sets, they can either use an event-based mechanism or simple poll. In the former case, XenoTrust notifies the participants whenever significant change happens, based on individual subscription. The latter allows the participants to query XenoTrust using the deployed rule sets.

Security

Let us now discuss the security implications of the XenoTrust system.

- XenoTrust employs a simple authentication mechanism to prevent forgery. However, it does not specify anything more, and the users are free to combine different negotiation and security mechanisms on top of XenoTrust.
- Similar to other reputation systems, XenoTrust is susceptible to negative reputation information. Sybil attack is also possible, but expensive as it requires participants to acquire new identities and provide information that significantly changes the overall trust values of the system.
- Replay attacks are prevented using timestamps to maintain the freshness of the trust updates.

10.3.3 NICE Trust Management System

The NICE framework [224], developed at the University of Maryland, is a platform for implementing cooperative applications over the Internet, which can be defined as a set of applications that allocate a subset of resources, typically processing, bandwidth, and storage, for use by other nodes or peers in the application. Therefore, grid computing is naturally an application for the NICE trust management framework.

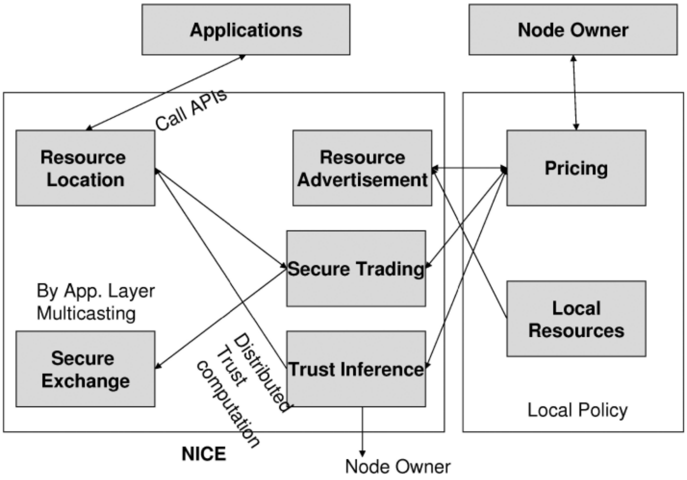


Fig. 10.7. Overview of the NICE framework

Figure 10.7 shows the high level architecture of the NICE framework. As shown in the figure, applications locate the resources using APIs of the NICE framework. The different secure trading, bartering, and redeeming protocols are implemented within the NICE framework and are not exposed as APIs. The different nodes exchange information based on policies through an application layer multicasting [225] based signaling protocol. One of the unique features of the NICE system is a distributed trust evaluation scheme, which identifies robust cooperative group, and is able to isolate and hence nullify the effects of malicious nodes. Following are some of the key components of the NICE trust management framework.

Policies in NICE

One of the basic assumptions of NICE is that malicious users can consume resources so that the valid users can be prevented. The default policies in NICE are used so that the resource consumption can be limited by cliques of malicious users. There are two different policies to ensure limited usage of resources: trust-based pricing and trust based trading limits.

- **Trust-Based Pricing:** In this type of policy, resources are priced proportional to mutually perceived trust. This policy is motivated by the trading system and hence the name. Transacting or trading with a peer having significantly low trust will induce more risk for a peer, which is reflected on the subsequent price of transaction.
- **Trust-based Trading Limit:** In this policy, instead of varying the price of the resource, the policy varies the amount of resources. This policy assures that when trading with a principal with relatively low trust, a peer is bounded by the amount of resources that a peer can lose.

Trust Metrics and Evaluation

Another interesting feature of NICE is the distributed trust evaluation mechanism. Whenever there is a transaction between two different nodes or peers, the peer receiving the service signs a cookie showing that the peer generating that service has successfully completed the transaction. Therefore, each node maintains a trust relationship and trust value based on the transaction it has received. It is to be noted that the node generating transaction can store the signed cookies to prove its trustworthiness sometime later. NICE framework defines two metrics of trust between two nodes: *strongest path* and *weighted sum of strongest disjoint path*. The former is the trust value of the strongest path, and the latter is the weighted average of all the disjoint paths. Since the centralized and flooding based mechanisms are both inefficient in terms of computation and bandwidth wastage respectively, the implementers have used an intelligent forwarding scheme for the distributed implementation of metrics computation.

Security, Reliability, Scalability

Different security features of NICE are as follows:

- **Authentication:** In the NICE framework, each and every node signs the cookies that they transact and send signed updates. It is to be noted that there is no need for the key associated with NICE to be registered with any central authority. Each user can generate its own key.
- **Sybil Attacks:** Whenever each user can create its own persona, there is a possibility of sybil attack in case of malicious users. To prevent such an attack a PKI needs to be integrated with the

NICE system. However, one of the positives of the NICE architecture is the capability of the system to isolate malicious users through cooperative which can reduce the effectiveness of sybil attacks and other denial-of-service attacks.

- **Reliability:** The ability of NICE to form robust groups makes it reliable in nature. However, NICE does not implicitly use a reliable storage and fault tolerant mechanisms to make the storage and information retrieval reliable.
- **Scalability:** The distributed storage of information and probabilistic and intelligent flooding makes the protocol highly scalable. The underlying signaling protocol based on application layer multicasting increases the scalability, though may reduce the performance of the system as the signals go through multiple layers over the physical infrastructure.

10.3.4 Secure Grid Outsourcing (SeGO) System

The Secure Grid Outsourcing (SeGO) system [226, 227], conceptualized and developed at the University of Southern California, is developed for secure scheduling a large number of autonomous and indivisible jobs to grid sites. A unique feature of the work is that the authors use a fuzzy inference approach to binding security in trusted grid computing environment. The authors define the trust metrics based on the *site reputation* and *defense capability* of a resource. The first characteristic is the behavior attribute of a site and is composed of four parameters related to jobs behaviors like: Prior job execution success rate, cumulative site utilization, job turnaround time, and job slowdown ratio. The second criterion is based on defense capabilities of a site namely, the IVS related capabilities, Antivirus capabilities, firewall capabilities, and secure job execution capabilities. Fuzzy logic is used to integrate the different capabilities to finally develop the trust metric. Figure 10.8 shows a high level view of the SeGO architecture.

SeGO introduces fuzzy logic based trust integration model. The approach makes two basic assumptions:

- All resources have prior agreement for participating in grid operations. Therefore, SeGO does not look at trust negotiation, and trusted node join and leave mechanisms.
- The sites report their information honestly. Therefore, the possibility of malicious resources is discounted.

Based on the above assumptions, let us now discuss the different components of the SeGO system.

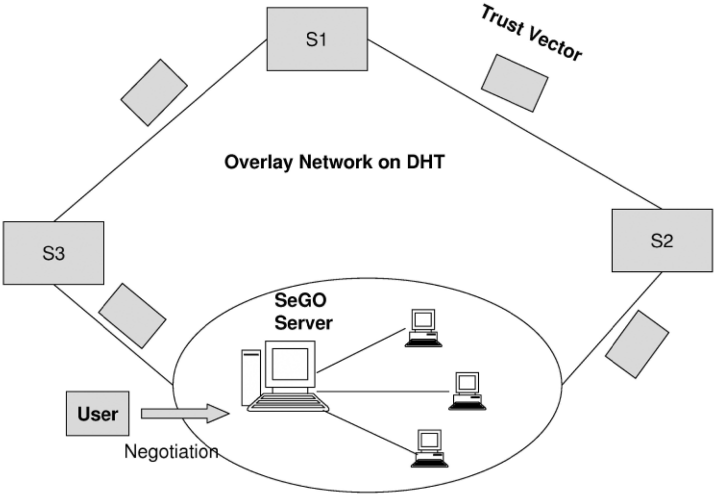


Fig. 10.8. SeGO fuzzy trust integration over grid

SeGO Trust Model

The SeGO model introduces a hierarchical trust model, specifically designed for distributed security enforcement in computational Grids. This model has two levels of trust inference: the lower level fuzzy inference system collects all input parameters from a single site, thus called the *intra-site level*. The output of the intra-site level provides the inputs to the upper level. The upper level collects inputs from all resource sites, thus called the *inter-site level*. There are two fuzzy inference systems applied in the intra-site level. One evaluates the self-defense capability and the other one evaluates the site reputation. Each site reports its assessed self-defense capability to all other sites. There is only one fuzzy inference system at the inter-site level, which collects inputs from intra-site levels, and infers the site trust indices to form the trust vector for each site.

Trust Integration and Updates

SeGO scheme does not explicitly deal with the storage of trust information. A DHT based approach similar to PeerTrust can be used to store the trust information in a distributed manner. SeGO can also be integrated with a VPN-based direct transfer of trust information over the grid. Each site has a SeGO agent installed on a SeGO server which is used for authenticating the user and initial negotiation. The SeGO agent has a *resource manager* and a *trust manager*. Resource manager is used for monitoring of resource status, and the trust manager is used for calculating the trust index. The scheme also allows each site to reassess the reputation of some other site through a manner similar to Time to Live (TTL) used in the networks. After counting the execution of sufficient number of jobs, a site A reassesses the reputation of another site B. A new trust index is computed as a function of the old trust index and the received information.

10.4 Policy-Based Trust Management Systems

While the reputation-based TMS are mainly concerned with developing a trust metric based on the different components of the performance of the system, policy-based systems are mainly concerned with developing a policy based trust language and developing trust negotiation mechanisms. In this section, we will discuss about some of the mechanisms in this regard.

10.4.1 PeerTrust Trust Negotiation

At the outset, let us mention that the PeerTrust policy mechanism is different from PeerTrust reputation model discussed previously. These are two different projects from two different universities. PeerTrust Policy is a collaboration project between the University of Chicago, Urbana Champagne and the University of Hannover in Germany. The main application of the PeerTrust policy mechanism is the semantic Web and the researchers have developed a simple and expressive policy language for trust negotiation. The two basic components of a PeerTrust policy system is the PeerTrust policy language and the PeerTrust automated trust negotiation.

PeerTrust Policy Language

The PeerTrust policy language is quite extensive. It provides well-defined semantics, ability to express complex conditions, sensitive policies, and delegation.

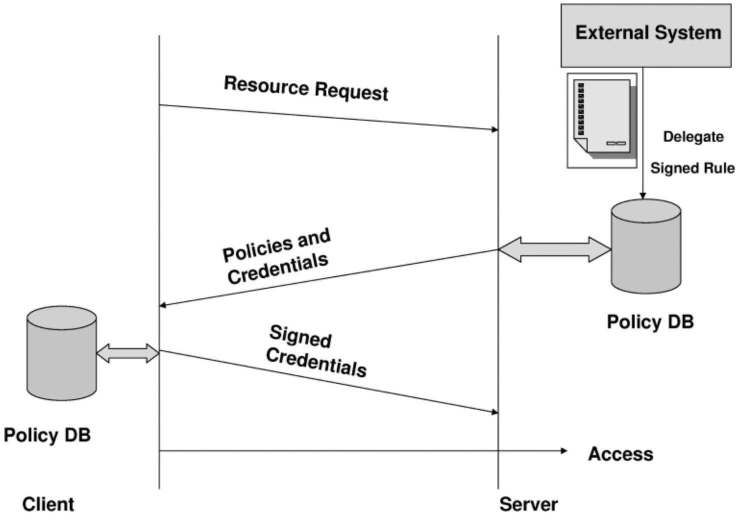


Fig. 10.9. Negotiation in PeerTrust

We will provide a brief overview of the syntax and semantics of the PeerTrust system. More details can be found in [219].

- Each peer defines a policy for each of its resources, in the form of a set of definite Horn clause rules. These and any other rules that the peer defines on its own are its *local* rules. A peer may also have copies of rules defined by other peers, and it may use these rules in its proofs in certain situations. Definite Horn clauses are the basis for logic programs [228], which have been used as the basis for the rule layer of the semantic Web and specified in the RuleML effort [229, 230] as well as in the recent OWL Rules Draft [231]. Each rule is signed based on the issuer's digital signature to verify the authenticity.
- The semantics of the PeerTrust language is an extension of that of SD3 [232]. For each authority argument that has not been specified explicitly in a rule or literal, '@ Self' is added. The policy is determined by a forward chaining nondeterministic fixpoint computation process. In this process, at each step, a nondeterministically chosen peer can have three actions: Firstly, it can either apply

one of the rules. Secondly, it can send a literal or rule in its knowledge base with a certain context. Thirdly, it can receive a context-free signed rule or literal from another party.

PeerTrust Automated Trust Negotiation

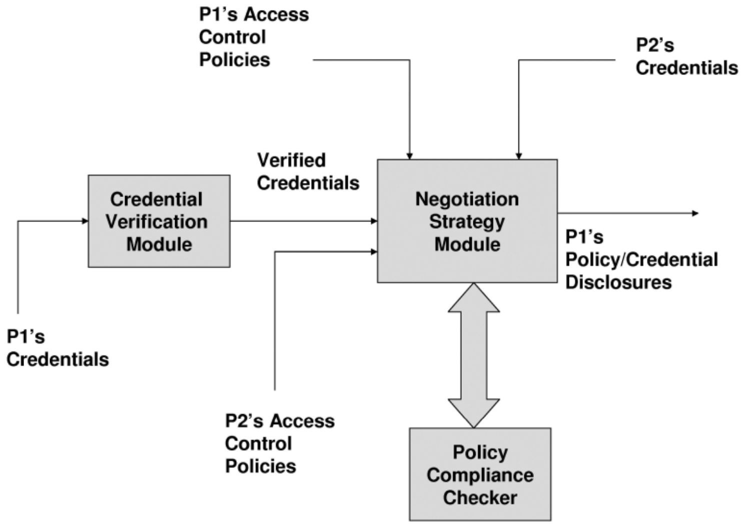
Figure 10.9 shows the high level automated trust negotiation in the PeerTrust policy management system. The server on receiving client's request regarding the resource can look at its policy database, which is formed by set policy rules and signed rules delegated by external sources. Based on the set of rules, the server can ask for certain credentials from the client. The client also may have some policies based on credentials disclosure which are embedded in the disclosed credentials.

10.4.2 TrustBuilder

TrustBuilder [233] is a trust negotiation project done through collaboration by researchers in the University of Chicago, Urbana Champagne, and the Brigham Young University. Unlike the PeerTrust policy mechanism, which develops a policy language for trust negotiation, TrustBuilder develops an infrastructure for trust negotiation for open systems. The TrustBuilder system has deployment of trust negotiation on TLS, SMTP, POP, ssh, and HTTPs.

TrustBuilder basically allows strangers to access sensitive data and services over the Internet. In TrustBuilder protocol and architecture, the negotiating parties establish trust between themselves by negotiating trust in a need-to-know manner. In this way, all the credentials are not disclosed to the either party. To ensure correct inter-operation between the different parties, TrustBuilder has identified four conditions:

- Both the parties should use the same protocol which defines the messaging order, and information contained in the messages.
- Whenever one party's strategy recommends the disclosure of a resource, the credentials previously disclosed by the other party must satisfy the resource's policy.
- If the policies of two parties allow a successful negotiation, there should be sequence of credential disclosure steps.
- There should always be a termination strategy in case the negotiation is unsuccessful.



TrustBuilder Architecture
P1's Access Control Policies
P2's Credentials
Verified Credentials
Credential Verification Module
Negotiation Strategy Module
P1's Policy/Credential Disclosures
P1's Credentials
P2's Access Control Policies
Policy Compliance Checker

Fig. 10.10. TrustBuilder security agent architecture

- **Negotiation:** Figure 10.11 shows the high level architecture of the security agent which negotiates on a party's behalf to mediate a stranger's access to a party's local resource. The figure shows how P2's security agent identifies the amount of credentials to be disclosed to P1. P2's security agent uses a policy compliance checker to determine which of P2's policies are satisfied by P1's disclosed credentials so that none of P2's local resources is disclosed to P1's agent before the resource's policy has been satisfied. P2's agent also uses a compliance checker to determine which of P2's credentials satisfy P1's disclosed policies. When P2's security agent receives a credential from P1's agent, the credential verification module performs a validity check, including signature verification, revocation check, and credential chain discovery when necessary. The verification module also handles P1's agent's demands that P2 demonstrate the possession of a private key that matches a certified public key.

- **Ubiquity**: TrustBuilder also allows for ubiquitous trust negotiation which works across multiple protocols like TLS, by leveraging the TLS rehandshake facility [234] RMI, SOAP, HTTP, and other protocols.

10.4.3 Trust Negotiation for the Grid

TrustBuilder is an interesting piece of work in the domain of policy-based trust negotiations. Researchers have integrated the work with the grid schedulers to develop a grid-based trust negotiation mechanism [235]. Grid-based trust negotiation architecture combines the TrustBuilder for underlying trust infrastructure, and PeerTrust for the language for automated trust negotiation.

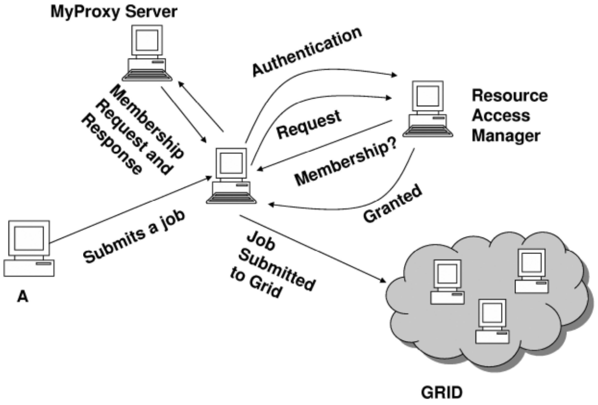


Fig. 10.11. Trust negotiation over the grid

Figure 10.11 shows the high level architecture of trust negotiation in the grid. In the figure, a user A wants to submit a job onto the Grid infrastructure. The grid uses MyProxy for credential management, and a scheduler for submitting the job. Let us go through the different steps involved:

1. User A sets up a TLS session with the resource access manager.
2. Once the session is established, a TLS rehandshake is established using the TLS rehandshake protocol defined in [234]. A also sends the policy written in RT [236] or PeerTrust language

[219]. Based on the policy, the access manager sends the policy and the certificates, if required to A.

3. If the policy accepts a proxy certificate chain then that is obtained from the MyProxy server. Based on all the information, the job is granted access to the Grid resource.

The above example shows how the TrustBuilder agents can be used for policy checking and negotiating strategies.

10.5 Comparing the Trust Management Systems

We had previously mentioned the different desirable characteristics of the trust management systems. This section tries to understand how the different systems perform with respect to the characteristics mentioned before.

10.5.1 Generic Understanding of Trust Management Systems

Let us now look at how the different trust management systems perform with respect to individual characteristics. Table 10.2 shows the mapping between the different trust management systems and the characteristics.

Scalability

Reputation-based trust management systems generally involve exchange of trust metrics among the different entities or trust nodes. Policy-based systems, on the other hand, require a policy database and information to make the trust related decisions. Therefore, with respect to message overhead, most of the reputation-based systems perform below par compared to the policy-based systems. However, the amount of message overhead depends on the specific type of trust function used. Among the different reputation-based systems, PeerTrust, and SeGO use global trust functions and hence require a significant amount of message exchange to determine the trust metrics. Therefore, it is not surprising that both of them have high message overhead. Since NICE is based on local information, the message overhead is controlled in this case. One exception to the general trend is the XenoTrust system which has a centralized information base and has low message overhead. policy-based systems, mostly, has low message overhead as the message exchange is minimal. The computation overhead associated with the different systems depends on the type of trust function, metrics, and policies used. Reputation functions used in XenoTrust, PeerTrust, and SeGO are quite complex and require significant computational resources, especially since the metrics need to be computed based on global information. NICE, on the other hand, require a path-based trust metric computed on local information and hence is less expensive. The main component of latency in policy-based systems is due to searching the policy database rather than computing complicated trust functions.

In terms of computation overhead, however, the reputation-based trust management systems are mostly better than their policy-based counterparts. The reason is that, reputation-based systems, can use a distributed implementation of the reputation information by using concepts like Distributed Hash Tables (DHT). PeerTrust and SeGo use DHTs to reduce the storage. However, it should be noted that distributed storage increases the message overhead and overall system latency. Most of the policy-based systems use a centralized database to store the policy information and hence generally require higher storage.

Reliability

Reliability achieved in most systems in through replication of information. Since XenoTrust assumes that the server maintains the entire information base, it has a single point of failure. However, replication of the information can improve the overall reliability of the system. Policy based systems also require some amount of replication to provide better reliability. Systems using DHTs to store information like PeerTrust and SeGo have the advantage in this respect, as selective information can be replicated across the DHT.

Security

Security is one of the areas where the policy-based systems score more than the reputation-based systems. Reputation-based systems are always prone to collusion attacks as the reputation information computed depends on the information sent by the other nodes in the system. Therefore, strategically placed adversaries can collude and create corrupted information. All the reputation-based systems are prone to this type of attack. Some of the systems like XenoTrust, NICE, and SeGo are prone to sybil attack also where a node creates several virtual nodes to create confusion and corruption. PeerTrust prevents this type of attack by having a PKI infrastructure in place.

Table 10.2. Comparing the different trust management systems

Params	PeerTrust	XenoTrust	NICE	SeGO	PeerTrust Policy	TrustBuilder	Policy Grid
Trust Fns	Global	Global	Local	Global	Policy based	Policy based	Policy based
Centralized	No	Yes	No	Can be	Yes	Yes	Yes
Storage	Low	High	Low	Can be Low	High	High	High
Computation	High	High	Low	High	Medium	Medium	Medium
Reliability	High	Replication possible	High	Can be high	Replication possible	Replication possible	Replication possible
Bandwidth	High	Low	Medium	Can be High	Low	Low	Low
Authentication	Using PKI	Signature based	Signature based	No	Can be included	Yes	Using GSI
Collusion	Possible	High	Possible	Very High	No	No	No
Sybil Attack	No	High	Possible	Very High	No	No	No
Trust Negotiation	No	No	No	Can be included	Yes through PeerTrust Language	Mainly a trust negotiation scheme	Integrating TrustBuilder with Grid

10.5.2 Applicability of the Trust Management Systems

We have looked at different trust management systems. One may ask the question about which of them can be readily deployed. Among the two general categories of TMS, policy-based TMS is of immediate need rather than reputation-based TMS. There are a couple of reasons for this observation. Firstly, the reputation-based TMS systems are still in evolution and developed in research labs, and would require a few years before maturing. Secondly, the vision of having a dynamic grid system with grid nodes joining and leaving the system is not likely to be realized in the next few years. However, our feeling is that enterprises would be requiring some type of trust management system in the next few years to cater for the demand of multiple enterprise grids. Then some form of policy-based TMS would evolve, most likely from an evolved version of a CAS or VOMS type of authorization system with more advanced integrated policies. In the next five years, we expect to find the development of grid based trust language and research and deployment of reputation-based TMS existing with the policy based authorization systems.

10.6 Chapter Summary

In an extremely dynamic system, where members join and leave, managing and maintaining trust is extremely important. A dynamic and large grid system is also no different. The life cycle of a typical Trust Management System (TMS) consists of trust creation, trust negotiation, and trust management. The trust creation phase generally is done before any trusted group is formed, and it includes mechanisms to develop trust functions and trust policies. Trust negotiation, on the other hand, is activated when a new untrusted system joins the current distributed system or group. The third phase, or the trust management phase, is responsible for recalculating the trust values based on the transaction information, distribution or exchange of trust related information, updating and storing the trust information in a centralized or in a distributed manner. The different characteristics of a good trust management system are scalability, reliability, and security. A TMS needs to be scalable in terms of message and storage overhead, must have message level and node level reliability, and be secure against different attacks like fabrication, masquerade, anonymity, and collusion. The different TMS include reputation-based TMS and policy-based TMS. The different TMS systems discussed in this chapter are PeerTrust, XenoTrust, NICE, and SeGO. The different policy-based systems discussed in this chapter are PeerTrust policy, TrustBuilder, and policy grid. The different systems have been discussed in detail and contrasted. As can be observed from Table 10.1, reputation-based systems like XenoTrust, NICE, SeGO are vulnerable to collusion due to the cooperative nature of the protocols involved. Policy-based systems, on the other hand, though immune from collusion attacks require more storage and are less flexible in managing trust. Significant research efforts are needed in this area. In the next chapter, the different grid based monitoring systems will be discussed.

11 Grid Monitoring

11.1 Introduction

Sharing of resources and collaboration among the different participating entities form the basis of grid computing. In a grid computing environment jobs are submitted to the heterogeneous grid infrastructure by users who may be in different departments or even organizations. Collaboration of users takes place in a grid community through the formation of a Virtual Organization. In many cases, usage of resources needs to be tracked or *monitored*. Monitoring of resources is needed because of primarily two reasons. Firstly, different organizations or departments can be charged based on their usage. Secondly, resource related information can be logged for auditing or compliance purposes. In addition to tracking and logging, there may be a need to integrate with the variety of different systems like scheduling, replication, etc. which may be able to use the monitoring information for better performance. In this chapter, we provide an overview of different distributed monitoring tools and techniques that can be applied to the grid computing scenario.

It would be wrong to assume that monitoring is essential in only the digital domain. Examples of distributed monitoring are very common in day-to-day life also. Market research surveys are examples of monitoring of human behavior with respect to a certain product or a brand. In these surveys, different field researchers ask several questions based on a common questionnaire to the consumers. The field researchers also take a lot of pain in gathering the information time, period, and the profile of the consumers they interview. There is a team of coordinators whose responsibilities include collation of the responses gathered by the field researchers and making meaningful assessment based on the information. If we contrast this scenario with a face-to-face interview with a celebrity, in the latter case more flexibility is given to the interviewer in framing the questions, and collation of the information from the answers provided to the questions. This clearly shows the difference between centralized monitoring systems and the distributed one. While market research is an example of a distributed monitoring system, interviewing a celebrity is like monitoring a single system. Distributed monitoring systems are more complex to design as they introduce synchronization and scalability problems. Market researchers try to solve this problem by not only synchronizing in terms of time, period, and demography of the consumers, but also having a set questionnaire. These steps try to remove individual biases in getting consumer feedbacks. Similarly, when one is designing distributed monitoring systems, there is a need to gather information in a specified format, and synchronize the information. Sometimes it may be required to determine the freshness of gathered data based on the timestamp associated with the data. Hence, there may be a need for synchronizing time across the distributed system. Another complexity introduced in a distributed monitoring system is scalability. This can be handled in creating a hierarchical model where the information is aggregated at each point.

11.1.1 Stages of Monitoring

Since grid systems are typically distributed in nature, we will focus our attention on distributed monitoring. Monitoring of distributed systems has been the focus of research for the distributed computing community for many years. Several distributed monitoring architectures have been defined and analyzed in papers [237, 238]. Distributed systems are composed of independent components. For example, a grid infrastructure is composed of servers, desktops, and the network connecting the different components. The different stages of distributed monitoring are data collection, data processing, data distribution, and data presentation.

Data Collection

The first stage in distributed monitoring is to gather or collect data from different components through *sensors*. In the area of market researchers, this activity is carried out by the field researchers. Similar to the field researchers, the sensors gather specific information like the bandwidth usage, CPU and memory usage, health of the node, and so on. There are two ways in which the data can be gathered by the sensors namely *active* and *passive monitoring*. In the former case, the sensors actually introduce load or messages in the distributed system to gather the information. As a general rule, one end (the *probe*) generates a specific traffic pattern, while the other (the *target*) cooperates by returning some sort of feedback; the ping tool is a well known example. Passive monitoring, on the other hand, gather the information by looking at the link, CPU utilization, memory, without actually introducing any message or load in the system. Once the data have been collected, the sensors report the data in a specific format or structure to the other layers. The gathered data can be static is nature like the network topology, configurations of machines, or dynamic like the utilization of the system, system load, network available bandwidth, and so on. Moreover, the data can be collected in a time-based or in an event-based manner. In the former case, data is collected in a periodic manner, while in the latter, data is collected only when a specified event takes place.

Data Processing

This step involves getting information out of collected data. This step is generally application-specific and may take place during any stage of the monitoring process. Typical examples include filtering according to some predefined criteria, or summarizing a group of events (i.e., computing the average). This step is essential as sensors may gather different types of data from the components. However, an application may not be concerned about all the gathered data and may only be concerned about certain information, or in some cases aggregated information. This step is similar to the coordinators gathering the data supplied by the field researchers and summarizing and packaging them in meaningful information. Due to the hierarchical nature of many distributed monitoring systems, there is a need to process and filter data at every step of the hierarchy.

Data Transmission

This step involves the transmission of collected and processed data to the different entities interested. Transmission involves sending the data in a format understood by other parties over a transmission medium for example the network. Therefore, this step is also concerned with securely transferring the data over the network. Depending on the nature of the data, different authentication, encryption, and integrity mechanisms can be applied to secure the transmitted data.

Data Storage

There may be a need for storage of gathered or processed data for future references. For example, administrators may be interested in the average utilization of a grid system for auditing purposes. Depending on the criticality of the gathered and processed information, proper security measures need to be taken.

Data Presentation

The CEOs of the company sponsoring the market analysis may not be interested in the huge amount of varied data that has been collected. Rather, they would mostly be interested in reports that provide the answers to the questions they seek. Similarly, different distributed monitoring schemes present the data in forms of abstractions so that the data can be comprehended by the end users in a specified manner. The results may be presented in a standard format so that it can be fed to different systems for visualization or analysis purposes.

11.1.2 Requirements of Distributed Monitoring System

During the design of a distributed monitoring system, one should be primarily concerned about the scalability of the systems. In addition, one should also look at the scalability, flexibility, portability, robustness, and security requirements.

Scalability

As mentioned earlier, scalability is one of the most important criteria of a distributed monitoring system. The monitoring system should be able to scale with growing number of resources and users. Therefore, the monitoring system must be designed with scalability in mind using a hierarchical model. Moreover, the probes and sensors used in gathering the data from the system should not introduce an excessive amount of load in the system. One way to achieve this is through a nonintrusive way of gathering information.

Flexibility

Another important requirement of any distributed monitoring system is its flexibility. It should be flexible enough to accommodate different data formats and schemas. It should also not only be flexible enough to handle static and dynamic requests, but also application specific policies regarding measurements can also be added. It should also be flexible enough to be extended to different event types, protocols, and standards.

Portability

One of the main characteristic of the grid infrastructure is its heterogeneity. Therefore, the distributed monitoring system suitable for grid systems should be portable i.e., it should be able to work across the different systems and platforms.

Robustness

The system should be robust to infrastructure failures like node and network failures of various types. As systems scale in the number of nodes, failures become both inevitable and commonplace. The system should be able to localize such failures so that the system continues to operate and delivers useful service in the presence of failures and the effect is minimized.

Security

Depending on the nature of the data, care must be taken to secure the stored, processed, and transmitted data. Security principles like authentication, confidentiality, integrity must be maintained in all transactions involving secure data.

11.2 Grid Monitoring Architecture (GMA)

The Grid Monitoring Architecture (GMA) has been put together by the Global Grid Forum (GGF) as a recommendation for a grid monitoring system. It is to be noted that it is not a standard and has been put forward for further discussions and implementations and if possible getting results. Figure 11.1 shows the components of the Grid Monitoring Architecture (GMA). The different components of the GMA are sensors, producers, consumers, and the directory service.

- **Sensors:** Sensors are the source of the monitoring data and are responsible for gathering data from the distributed grid system. They are also responsible for generating events from the measured data. The architecture can be further simplified by placing the sensors in the consumers or users as mentioned in [239]. However, the architecture implicitly assumes that sensors are the logical block in the overall GMA structure [240].

- **Producers:** In GMA, the producers are responsible for registering themselves to the registry or the directory service and also provide information about the type and structure of information they want to be made available to the grid. The producers are also some form of sensor managers by making available the amount and type of data that would be useful to the consumers.
- **Directory Service or the Registry:** The directory service (also known as the registry) is responsible for publishing the event type and the corresponding producers. Consumers can contact the service to find out the type of information available and locate the producers who can provide the information.
- **Consumers:** These are the users of the monitoring data. The consumers query the directory service and get the information about the producers. Once this information is obtained, the consumers contact the producers directly to get the information.

GMA defines three types of interactions between producers and consumers.

- **Publish/subscribe** refers to a three-phase interaction consisting of a subscription for a specific event type, a stream of events from a producer to a consumer, and a termination of the subscription. Both the establishment and the termination of a subscription can be initiated by any of the two parties. This type of interactions is very useful when the producers are generating data in a periodic interval, and the consumers can subscribe to the event.
- A **query/response** is a direct interaction initiated by a consumer and followed by a single producer response containing one or more events. This event is useful when the consumer require a specific type of information, and similar to a database query.
- A **notification** can be sent by a producer to a consumer without any further interactions. These can be because of some alarms or triggers that need to be generated.

In addition to the three core components, the GMA also defines a *republisher* which is sometimes referred to as a compound component or intermediary and a *schema repository.*

- A **republisher** is any single component implementing both producer and consumer interfaces for reasons such as filtering, aggregating, summarizing, broadcasting, and caching.

- A **schema repository** holds the event schema, that is, the collection of defined event types. If a system is to support an extensible event schema, such a repository must have an interface for dynamic and controlled addition, modification and removal of any custom event types.

The GMA was devised as a framework for monitoring grid systems. However, it goes beyond that and provides a nice framework for combining monitoring and information systems. The model is not constrained by any protocol or data models. Therefore, implementers are free to choose their own data models for querying over the monitoring system.

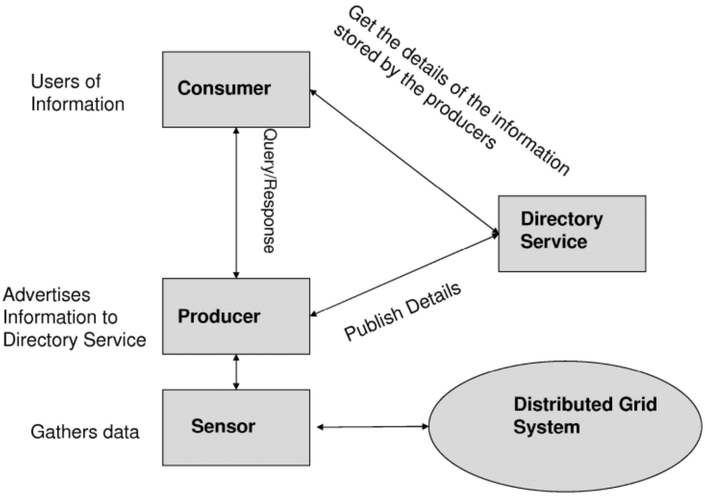


Fig. 11.1. Grid Monitoring Architecture (GMA) overview

11.3 Different Monitoring Tools/Frameworks

Let us now discuss the different monitoring tools available. Monitoring systems can be divided into three main types: *system level*, *cluster level,* and *grid level*. The system level monitors collect and communicate information about standalone systems or networks. The main difference between the cluster level and grid level monitors lies in the heterogeneity supported by the monitoring systems. The cluster level monitoring systems generally are homogeneous in nature and require deployment across cluster or a set of clusters for monitoring purposes. Grid level monitoring systems are much more flexible and can be deployed on top of different other monitoring systems. We will discuss the Simple Network Management Protocol (SNMP) and Mon as part as examples of system level monitors. It is to be noted that SNMP is a standard for communicating management and monitoring information and not a tool. However, it can be combined with other tools to provide network specific information. As part of our discussions on cluster level monitoring systems, we will discuss Ganglia and Hawkeye. On the other hand, we will discuss about R-GMA, MDS, MAGI, and GlueDomains as part of our discussions about grid level monitoring systems.

11.3.1 Simple Network Management Protocol (SNMP)

Perhaps the most popular protocol for managing and monitoring network devices is the Simple Network Management Protocol or SNMP [241]. With regard to the OSI stack, SNMP is an application-layer communication protocol that allows network devices to exchange management and monitoring information. By providing the monitoring and management of information, SNMP enables network administrators to manage network performance, find and solve network problems, and plan network growth. Traditionally, SNMP was designed and used to gather statistics for network management and capacity planning. For example, the number of packets sent and received on each network interface could be obtained. But because of its simplicity, SNMP's use has expanded into areas of interest to smaller networked devices. It is now used for many vendor-specific management functions, e.g., showing a thermostat temperature, machine tool RPM, or whether the front door was left open. An SNMP-managed network consists of three primary components: managed devices, agents, and management systems.

- **Managed Devices:** A managed device is a network node that contains an SNMP agent and resides on an SNMP-managed network. Managed devices collect and store management information and use SNMP to make this information available to management systems that use SNMP. Managed devices include routers, access servers, switches, bridges, hubs, computer hosts, and other network elements.

- **SNMP Agent:** A SNMP agent is a software module that resides in a managed device. A SNMP agent has local knowledge of management information and translates that information into a form compatible with SNMP. The SNMP agent gathers data from the Management Information Base (MIB), which is the repository for device parameter and network data. A MIB is a structured arrangement of managed objects consisting of a database that maps OIDs to actual variable instances. The MIB may be used as a stand-alone hierarchical database without SNMP if required. The SNMP agent can also send traps, or notification of certain events, to the manager.
- **Management System:** A management system executes applications that monitor and control managed devices. Management systems provide the bulk of the processing and memory resources required for network management. One or more management systems must exist on any managed network. A typical grid management system like MDS (refer to Sect. 11.3) can be extended to interact with SNMP. Other examples of management systems include HP® OpenView [242], CA® Unicenter [243], and so on.

SNMP is therefore essentially a request-reply type communication protocol operating between a management system and SNMP agents. Being standard-based, SNMP allows the integration with different management systems and information collecting devices (sensors) as long as they conform to SNMP standards.

11.3.2 Different System Monitoring Tools

In this subsection, let us provide an overview of some of the tools available for monitoring and managing the systems. Some of the tools mentioned are Orca, Mon, Aide, and Tripwire. There are a lot of tools available other than the tools mentioned here which can be used for monitoring purposes.

- **Orca Services:** Orcallator and Orca Services [244] collect system data and prepare it for Orca. Orcallator is a Linux version of Orca Services. While the Solaris version explores the SE toolkit kernel interface, the Linux version relies on information fetched from the /proc pseudo file system. Orca uses collected data to make very useful graphs through RRDTool. With rsync or ftp data can be collected from multiple hosts and visualized.

- **Mon:** *Mon* [245] is a general-purpose scheduler and alert management tool used for monitoring service availability and triggering alerts upon failure detection. *Mon* was designed to be open and extensible in the sense that it supports arbitrary monitoring facilities and alert methods via a common interface, all of which are easily implemented with programs in C, Perl, shell, etc., SNMP traps, and special *mon* traps.
- **Aide and Tripwire:** Aide [246] and Tripwire [247] are similar open source programs designed to monitor changes in a key subset of files identified by administrator, and report on any changes in any of those files. Files are scanned periodically (daily or more frequently) and the periodicity is defined through the cron facility. Any change, addition or deletion is reported by mail, so that proper action can be taken.

Most of the tools available for system monitoring purposes cannot be used as a standalone system for monitoring grid systems. However, information collected by the tools can be combined with the grid monitoring systems like MDS to provide an integrated system.

11.3.3 Ganglia

Ganglia [248] is an open source hierarchical monitoring system developed at the University of California, Berkeley, primarily designed for computer clusters but also used in grid installations. Ganglia has over 500 installations all over the world and is one of the most popular open source monitoring tools. Ganglia is designed in a hierarchical manner (see Fig. 11.2). It consists of three main components: intra-cluster monitoring, federation, and visualization. Intra-cluster monitoring is used to collect the information within a cluster, which is aggregated, and finally published through the visualizer.

Intra Cluster Monitoring

In the Ganglia system, the intra-domain Ganglia monitor or the gmon daemon (*gmond*), collects information based on different local metrics. Ganglia uses a multicast based listen/announce protocol to monitor the state within a cluster. The protocol works as follows: after collecting the information about the local system, each node sends the information to a well-known multicast address. Ganglia distinguishes between built-in metrics and application-specific metrics through a field in the multicast monitoring packets being sent. All nodes listen for both types of metrics on the multicast address and collect and maintain monitoring data for all other nodes. The information sent uses XDR format. Thus, all nodes always have an approximate view of the entire cluster's state and this state is easily reconstructed after a crash. However, this also introduces overheads in terms of having the multicast support, as well as message overhead. Within each cluster, Ganglia uses heartbeat messages on a well-known multicast address as the basis for a membership protocol. Membership is maintained by using the reception of a heartbeat as a sign that a node is available and the nonreception of a heartbeat over a small multiple of a periodic announcement interval as a sign that a node is unavailable. Therefore, Ganglia does not have any specific registration mechanism.

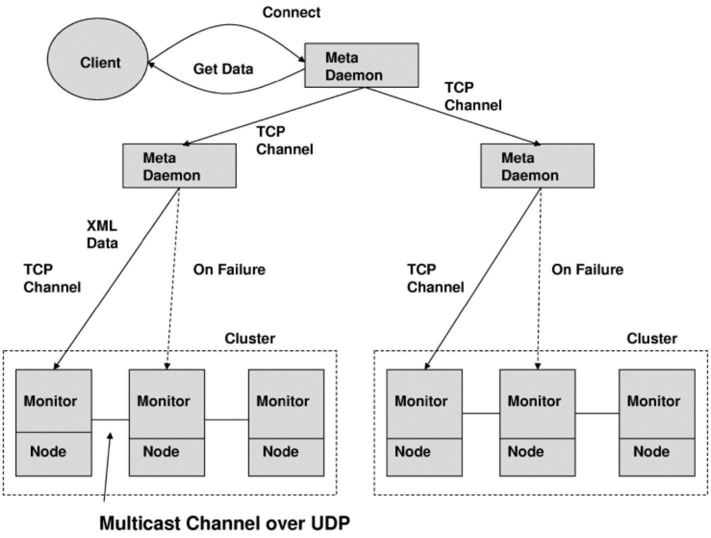


Fig. 11.2. Ganglia monitoring tool

Federation

Federation in Ganglia is carried out by Ganglia Meta Daemon (*gmetad)*. Ganglia constructs a tree of point-to-point connections to aggregate the states from multiple clusters. At each node, the Ganglia Meta Daemon collects the information from its children through XML over a TCP channel.

The data collection is carried out by polling multiple child nodes, which are mentioned in the configuration files. After collecting the information, the XML is parsed using a SAX parse, and sent to the parent node again over a TCP channel.

Visualization

Ganglia uses RRDTool (Round Robin Database) to store and visualize information for different types of systems like grid, clusters, etc. and for different metrics and time granularity. It also gives the flexibility for the developers to customize the information.

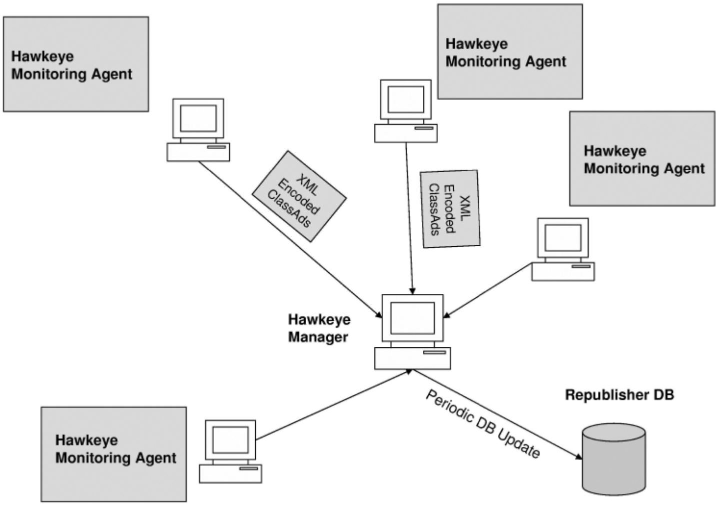


Fig. 11.3. Overview of the Hawkeye monitoring system

11.3.4 Hawkeye Monitoring System

Hawkeye [249] is another distributed monitoring system, coming from the University of Wisconsin Madison. It is used to monitor different aspects of the computing system which can be a cluster or a grid, namely monitoring the health of the nodes, system load, watching run-away processes, and so on. Mostly, Hawkeye system is used in combination with the cluster based scheduler called Condor [250, 251]. However, it is also available as a stand-alone version in Linux and Solaris.

As shown in Fig. 11.3, each node that is monitored hosts a Hawkeye monitoring agent which periodically calculates the different metrics that are measured. The information is periodically sent to the Hawkeye central manager using a XML based class advertisements as used in Condor. The manager stores the information periodically in a round robin database. The information can be queried through simple interfaces.

11.3.5 Relational GMA (RGMA)

The Relational Grid Monitoring Architecture (RGMA) [252] is built as part of the European Data Grid (EDG) project. It is a framework based on the GGF specification of GMA and combines the grid monitoring and information services based on the relational model.

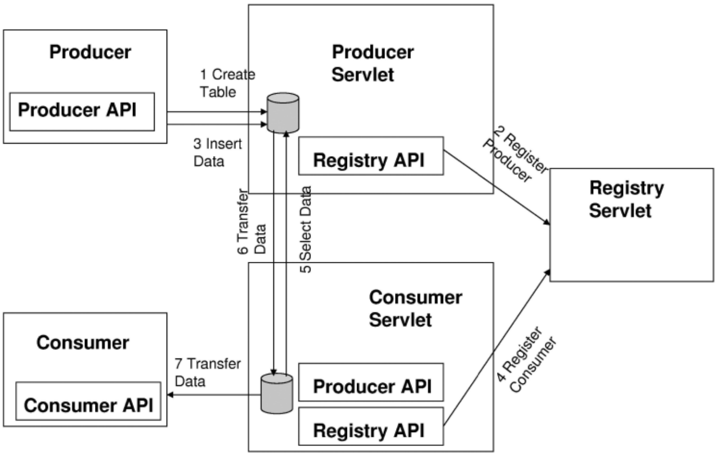


Fig. 11.4. RGMA architecture

Producers

Similar to GMA, RGMA also supports a producer-consumer architecture. In RGMA, the producers are categorized into four main classes namely, the *DatabaseProducer*, *StreamProducer*, *ResilientStreamProducer*, and the *LatestProducer*. The *DatabaseProducer* supports history queries and is employed for storing static data. This is slower than *StreamProducer*, where the data is stored in memory resident circular buffers. However, DatabaseProducers have more features like joins. The ResilientStreamProducer is similar to the StreamProducer but information is backed up to disk so that no information is lost in the event of a system crash. The LatestProducer supports the latest queries by holding only the latest records in an RDBMS. The architecture of the RGMA system is based on servlet technology. When a producer is created its registration details are sent via the producer servlet to the registry.

Consumers

Similar to GMA, the main purpose of a consumer is to get the information obtained by the producer. For this purpose, it takes the help of the registry. When a consumer is created its registration details are also sent to the registry via a consumer servlet. The registry records details about the type of data that the consumer has shown interest in. The registry then returns a list of producers back to the consumer servlet that matches the consumers selection criteria. The consumer servlet then contacts the relevant producer servlets to initiate transfer of data from the producer servlets to the consumer servlet and obtains the data.

Archiver

Another important RGMA component is the archiver which is a combined consumer-producer. An archiver works by taking over control of an existing producer and instantiating a consumer for each table it is asked to archive. This consumer then connects via the mediator to all suitable producers and data starts streaming from the producers, through the archiver and into the new producer. The inputs to an archiver are always streams from a StreamProducer or a ResilientStreamProducer.

Registry

Registry is perhaps the most important component, which connects the information gathering and information usage mechanisms. The registry stores information about all producers who are available. Extensions are being made to allow for multiple registries per Virtual Organization (VO), which would increase the scalability significantly. The registry stores information about the producer once the producer is created and registration details are being sent. The *global schema* includes a core set of relations, while new relations can be dynamically created and dropped by producers when required. The description of the data is actually stored as a reference to a table in the schema. In practice, the schema is collocated with the registry. Then when the producer publishes data, the data are transferred to a local producer servlet. Figure 11.4 shows the high level overview of the RGMA architecture.

11.3.6 Globus Monitoring and Discovery System (MDS)

The Monitoring and Discovery System (MDS) [253] of the Globus Toolkit is a suite of components for monitoring and discovering grid resources and services. The latest version of the MDS system (MDS4) is compliant with WSRF and WS-Notification specifications. The basic difference of MDS4 with cluster monitors like Ganglia is does not possess a detailed event handling mechanism like the latter. However, MDS4 can interface with different monitoring systems in multiple administrative domains. Example [253] also provides a simple case study about the deployment of MDS4 in an environment consisting of 30 sites. Out of the 30 sites, many of them run Ganglia as the monitoring system, some of them run Condor and Hawkeye, and many of them have proprietary queuing based scheduling systems like PBS, LSF Multicluster, etc.

The MDS4 has two main services which form the heart of the system which are responsible for gathering information from different sources and providing actionable trigger events.

- **Index Service:** It is perhaps one of the most important services in the MDS4 framework. It collects information about the grid resources and makes this information available. The index service collects information from the different data sources through standards WS-ResourceProperties and WS-BaseNotification services. Any service which can publish the information according to the WSRF specification can be indexed using the index service. Essentially, it is a republisher of data that was originally made available by some service. However, the index service does not guarantee the availability of the resource and the absolute freshness of the data available. It uses a soft consistency model and the data may be a little delayed to avoid unnecessary load in the system.

- **Trigger Service:** This service collects information and compares it to a set of conditions defined in the configuration file. Once a condition or a *trigger condition* is met a pre-defined action is taken. For example, an administrator needs to be emailed in case of node failure or when a server reaches its limits and so on.

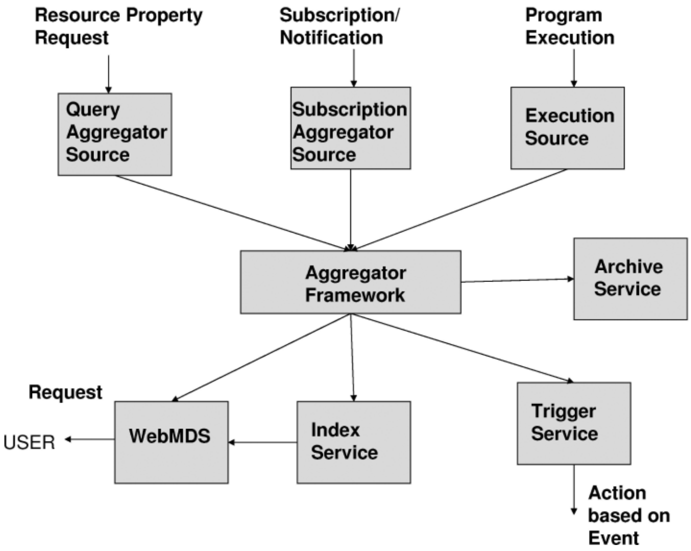


Fig. 11.5. Overview of the interactions in MDS4

The two services mentioned above collect information from the different types of sources called the *aggregator sources*. The aggregator sources may be of three types: The first type is a *Query* source which collects information using the WS-ResourceProperties interface. This source can work wherever the information is specified according to the specification. The second type is a *Subscription* source, which collects data from a service WS-Notification subscription/notification. The third and final type of source is an execution source which executes an administrator supplied program to collect data. This type of sources is very useful as most of the system and cluster administrators have custom defined scripts to collect data. The three different types of sources allow MDS4 to interact with data provided by other systems and custom defined scripts.

Figure 11.5 shows the interaction between the different components of the aggregator framework in MDS4. WebMDS is a web-based interface to WSRF resource property information that can be used as a user-friendly front-end to the index service.

11.3.7 Management of Adaptive Grid Infrastructure (MAGI)

Management of the Adaptive Grid Infrastructure (MAGI) [254] has been developed in the Grid Computing Focus Group of Infosys ® Technologies. MAGI includes the following features:

- **Multi-level QoS Repository:** The QoS repository of MAGI stores QoS-related data of different services or resources at different levels of a conceptual hierarchy. So there could be data about different parameters like: current queue length at a processor, average load value at a machine over the last 6 hours, and a composite reputation rating of different online book selling services. An important characteristic of this repository is that is makes the QoS related history of the service or resource persistent. Before making a selection of scheduling decision, therefore, a user can take into account all this historical data.
- **Scheduling:** MAGI allows the synching of scheduling with the QoS repository in the MAGI architecture. This allows scheduling decisions to be based on historical as well as transient data about the service. This also allows the user to specify constructs such as- "schedule the job at a resource with the minimum Q-length", or "select a service with a composite reputation rating greater than 80%."
- **Autonomic Capabilities:** In the context of multi-level QoS management, autonomic capabilities also allow the monitoring of contractual conditions based on QoS, and action to be taken in case these conditions are violated. These behave a lot similarly to the Trigger service in MDS4.

The components of MAGI include c-agents, i-agents, Meta Attribute Management Server (MAMS), Auditor and the Web based Interface. The details and the necessity of each of the components are described below.

- **c-Agents:** These agents are located at each execute node. It is the responsibility of a c-Agent (or contract agent) to see whether the node fulfills its contractual obligations to MAGI. For example a node can promise a physical memory of *M* over a certain period of time *t* and the c-agents monitor such resources on the node. It may so happen that a node fails to meet the contractual obligations. In such a case, there exists a reporting mechanism through which the auditor is informed.

- **i-Agents:** i-Agents (or infrastructure agents) are responsible for keeping an account of the resource availability at a given time. They differ from the c-agents in the sense that they are not aware of any contractual obligations of the node. They serve the purpose of reporting the resource availability on the nodes to the Job Submission and Scheduling System (JSSS), which can be a system like Condor, LSF Multicluster or PBSPro. Condor already provides this functionality through its ClassAd mechanism, but we have duplicated it for the sake of inter-operability with other systems.
- **Meta Attribute Management Server (MAMS):** MAMS is a database server containing three kinds of data. The transient data reflects all the contractual violations generally encountered in the last few hours, and is refreshed after a certain interval of time. While transient data reflects only the recent past, the entire history of the nodes' performance is stored in the historical service data of the MAMS database. The historical service data table is updated from the transient table periodically, averaging many of the parameters stored there. A third type of data in the database is the mapping data. This gives the user the option of not supplying the data which needs to be processed. MAGI, in such scenarios, takes the data to be processed from its own storage, or from a third party. The mapping data table contains the mapping of where this data is stored, and how to access it.
- **Auditor:** The *auditor* is a metering and monitoring system that also has exception handlers to handle any exceptions that may occur on account of the nodes on MAGI not meeting contractual obligations. In such cases, the auditor basically serves two purposes. Firstly, it updates the transient data of the MAMS database. The second function of auditor is taking control action, depending upon the type and severity of the violation. A typical example of control action can be checkpointing the job on a machine whose available memory is lower than a threshold in anticipation of node failure.
- **Web-based Interface:** MAGI provides a Web-based interface for observing the system performance. Users can query custom based as well as system specific information from the MAMS database.

11.3.8 GlueDomains

GlueDomains [255] is an interesting prototype which is included in our discussions mainly because of its novelty. GlueDomains supports the network monitoring activity of the prototype grid infrastructure of INFN, the Italian Institute for Nuclear Physics. GlueDomains follows a *domain-oriented* approach and the activity results are published using the Globus Monitoring and Discovery System (MDS). Following are the components of the GlueDomains monitoring system.

Domains

The domain oriented approach consists of an overlay network which divides the network into several partitions or domains. This concept is similar to the concept of Autonomous Systems (AS) in the Internet. The monitoring of the grid takes place in inter-domain rather than intra-domain which increases the overall scalability of the system. This type of monitoring scales in the order of O(D) rather than O(N) where D and N are the number of domains and the number of nodes in the grid infrastructure respectively.

Theodolite Service

A theodolite service monitors a number of grid network components which can be servers, storage, routers, and other components. In GlueDomains monitoring system, theodolites perform *active* network monitoring. In this type of monitoring systems, test traffic is induced into the grid infrastructure to see the effect and benchmarks are computed. Generally, each active monitoring service consists of two parts; one part (the *probe*) generates a specific traffic pattern, while the other (the *target*) cooperates by returning some sort of feedback.

Monitoring Database

The description of the overlay network is made available through a *topology database* which is generally static in nature. However, in a dynamic system this can be updated based on current network scenario. Observations collected by *active* monitoring tools are associated to a network service based on the location of the theodolites. Observations collected by *passive* traffic observers are associated to a specific network service using basic attributes (like source and destination IP address, service class, etc.) of the packets captured by such devices. The monitored information is updated periodically into an information system.

11.4 Discussions on the Different Monitoring Systems

Let us now try to summarize the different monitoring systems used for the Grid systems in this section.

11.4.1 Comparison

Let us first compare the monitoring systems based on the different qualitative parameters mentioned before. It is to be noted that grid and cluster based monitoring systems are complementary rather than competitive in nature. Grid monitoring systems can work at a higher level with the cluster monitoring systems providing information about the different clusters. Table 11.1 provides a comparison between the different monitoring systems.

Scalability

Monitoring systems which are hierarchical in nature are generally scalable in nature as they allow information to be aggregated and transferred. Most grid systems like R-GMA and MDS allow for hierarchical architecture and therefore scalable. MAGI and Hawkeye are architected as a one-level system and hence may suffer from scalability problems especially for a large number of nodes in the grid system. However, the issues are not as straight forward as this. As mentioned in the study carried out in [256, 257], the authors have concluded that R-GMA producer's and registry appear to be the least scalable compared to Hawkeye and MDS if the number of concurrent user's are very high. The study also showed that the number of consumers should be less that 400 for R-GMA and 500 for MDS (version 2).

Inter-Operability

This is one of the strong points of the grid monitoring systems. Most of the grid monitoring systems and frameworks have been designed to interoperate with other systems and interfaces, many times even vendor products. All the systems can interface with cluster and network monitoring systems for monitoring information. Cluster monitoring systems, on the other hand, are standalone in nature and require a significant amount of integration effort. However, it is to be noted that this is a complex issue as there exists plethora of communication methodologies (Java RMI, sockets etc.), information mechanisms (XML, flat files, RDBMS schema), and other issues. As mentioned in [256], producer-consumer based framework may not be sufficient in achieving inter-operability. However, more research is needed in this direction.

Security

Another important characteristic for the monitoring systems is security. Most of the grid monitoring systems have provisions for some type of security mechanisms. MDS and R-GMA use GSI credentials for authentication purposes. MAGI can use X.509 credentials as well as passwords for authentication purposes. Cluster-based monitoring systems do not explicitly provide any security mechanisms.

Table 11.1. Comparison between the different monitoring systems

Params	Ganglia	Hawkeye	R-GMA	MDS4	MAGI	GD
Hierarchical	Yes	No	Yes	Yes	No	Domain type
Integration with different systems	No	No	Yes	Yes	Yes	Through R-GMA
Registry	Not available	Using Condor manager	Through Registry system	Using Index Service	Using MAGI server	Not available
Security	Not explicitly	Not explicitly	GSI	GSI	X.509 based	SSL
Trigger Support	No	No	No	Using Trigger Service	Using Auditor	No
Visualization	Through RRDTool	No	Simple like Pulse	WebM DS	Web Interface	Possible
Scheduler Supported	Independent	Mostly Condor	Independent	Independent	Independent	Independent
Implementation Language	C	C++	Java Servlets	Java	C, Java, Perl	C, Perl
Comments	Requires Multicast support	Less features, stable performance	Not very stable under high load	Collects info. from different systems	Comparable to Hawkeye	Mostly collects network specific information

11.4.2 Applicability

Let us look at the current enterprise scenario. Enterprises are slowly looking at grid computing solutions. Monitoring is one of the immediate requirements for the enterprise grid systems. The reasons are threefold:

- **Pricing and Tracking:** Grid systems designed in most enterprises are composed of shared clusters and servers across the different departments of the enterprise. There is a need to track the usage of the different department and policies need to be developed based on the usage pattern. For example, an enterprise may assign 30% of its grid resources to Department A, 20% to Department B, and rest to Department C. The policies are generally not so straightforward and include time, users, priorities, and other parameters. Therefore, monitoring systems are needed and they should interact with the authorization system for proper policy implementation. The situation is complex as there are host level, cluster level, and network level monitors already implemented which need to be integrated with the grid monitoring systems.
- **Compliance:** The reason mentioned in the previous point is a more immediate concern for enterprises. However, compliance is one of the long-term concerns which enterprises are getting aware and grid systems need to conform to those. One of the major compliance issues is the Serbanes Oxley (Sox) compliance [258] which states that different entities in the enterprise supply chain need to comply to certain regulations. Regulations are also present in specific like life sciences where the processes need to comply to certain standards. These regulations require auditing of all the transactions taking place in the enterprise. The information available from the monitoring system can be used for auditing purposes.
- **Better Optimization:** In the next few years enterprises would also require scheduling of grid jobs based on business policies and metrics obtained from the monitoring systems. Therefore, better and more fine-grained monitoring would result in better optimization of the resources.

11.5 Chapter Summary

In this chapter, we have looked at the monitoring systems available in the grid. Data collection, data processing, data transmission, data storage, and data presentation are the different stages of a typical monitoring system. In designing a robust monitoring system, one should look at scalability, flexibility, portability, robustness, and security requirements. Monitoring tools/frameworks can be broadly divided into system level, cluster level, and grid level. The system level monitors collects and communicates information about standalone systems or networks. The main difference between the cluster level and grid level monitors lies in the heterogeneity supported by the monitoring systems. The cluster level monitoring systems generally are homogeneous in nature and require deployment across cluster or a set of clusters for monitoring purposes. Grid level monitoring systems are much more flexible and can be deployed on top of different other monitoring systems. In this chapter, we have discussed SNMP, Orca, mon, Aide, and Tripwire as examples of system level monitoring tools. Ganglia and Hawkeye have been discussed as part of the cluster level monitors, and R-GMA, MDS, MAGI, and GlueDomains have been discussed as grid level monitors. The different monitoring systems are compared and contrasted in the chapter. Researchers need to look at integrating the different host level, cluster level, and grid level monitoring systems for providing an end-to-end and holistic monitoring solution.

With this we come to the end of the discussions on management related security issues and solutions. In the next chapter, we will look at two case studies for integrating the different concepts discussed in the book.

12 Putting it All Together

12.1 Security in the European Data Grid (EDG)

Let us now concentrate on full fledged systems integrating all the components described in the previous chapters. The first example is from the European Data Grid (EDG) [259, 260] which has highly distributed resources spanning different management and security domains. EDG mainly deals with scientific applications which are typically long-running in nature. However users can submit jobs from different domains as well as geographical locations. Hence the security policies and mechanisms should be able to span multiple organizations and able to utilize services which are compatible with the grid at different levels. As in any other security system the different components include authentication, delegation, authorization, and credential management systems.

12.1.1 Authentication and Delegation

The authentication and delegation framework is based on the Grid Security Infrastructure (GSI). Please refer to Chap. 4 for more details. The default authentication mechanism in GSI is based on X.509 certificates, hence there is a need of a Certification Authority (CA). Instead of having a single CA which may result in a single point of failure in EDG it was decided that each participating country would have its own CA.

To prove its identity the user obtains a certificate from a CA and all the participants trust a user based on the issued certificate. As in GSI, the delegation in EDG is handled through proxy certificates. The user can issue the proxy certificate to some other delegated entity which can act on users behalf. For example there is a need of some agent to copy a file or run an application as the user. Since the resources are distributed, it may be unnecessary to authenticate the user every time. In this type of case delegation becomes important as the agent can do the activity as a user and submit the proxy certificate which the user had previously issued. The user credentials are delegated to the agent by copying the proxy certificate and transferred over a secured channel (SSL channel).

The secured channel prevents malicious adversaries from eavesdropping into the channel. It is to be noted that the proxy certificates are generally short lived, i.e. they expire after a limited amount of time. Therefore even if the certificate is stolen the adversary has limited time to orchestrate the attack using the compromised certificate. The authorization in EDG is provided by Virtual Organization Membership Service (VOMS), which is a central database of members constituting the Virtual Organization. More details about VOMS is provided in Chap. 5. VOMS uses a role-based mechanism for authorization. The VOMS database is managed by an authenticator and authorized administrator who is responsible for managing roles and responsibilities for different roles. An authenticated user can request group membership and capabilities which is provided by the VOMS service through a short lived Attribute Certificate (AC). This AC is presented by the user to the grid resources.

12.1.2 Credential Management

Another important component of the EDG security system is the MyProxy credential manager (for details see Chap. 9). The users store their long time credentials in the MyProxy system. Generally these credentials are valid for a few weeks to a month. The MyProxy server restricts the access of the proxy to a very select set of clients. One of the services that can access the proxy is the credential renewal service. The importance of this service lies in the fact that the jobs may actually take a longer time than what is specified in the proxy certificates. The credential renewal service periodically contacts the MyProxy server to ensure that a new proxy has been issued before the expiry of the old one. It is to be noted that the credential renewal service may need to contact the VOMS server in case the proxy certificate contains the VOMS specific information.

12.1.3 Job Execution

Now let us take a look at how a job is submitted to a grid resource (refer to Fig. 12.1).

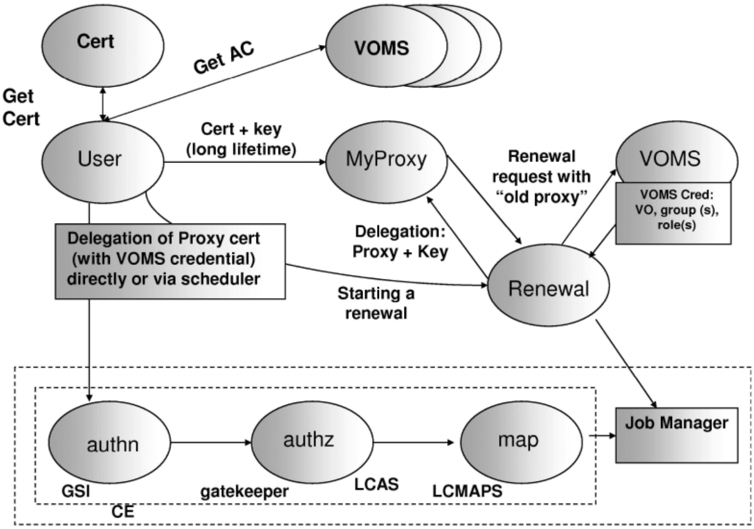

Cert
Get AC
VOMS
Get Cert
User
Cert + key (long lifetime)
MyProxy
Renewal request with "old proxy"
VOMS
VOMS Cred: VO, group (s), role(s)
Delegation of Proxy cert (with VOMS credential) directly or via scheduler
Delegation: Proxy + Key
Starting a renewal
Renewal
authn
authz
map
Job Manager
GSI
gatekeeper
LCAS
LCMAPS
CE

Fig. 12.1. Job execution overview in EDG

In the first step the user gets a certificate from a CA. The user then contacts the VOMS server to obtain the attribute certificate based on the user credential. It is to be noted that there may be multiple VOMS servers and the user needs to get the attribute certificates from each of them to be able to submit jobs to the resources which accept a specific VOMS credential. The user may then decide to store the long term credential in MyProxy credential manager. A proxy certificate can also be generated which can be delegated to a specific agent directly or with the help of a scheduler. In addition to the global authorization credentials supplied by the VOMS server a site may have local policies to be applied on top of the user specific authorizations. For example, a site may decide to ban or suspend a user or a group of users based on some policies. It may not be scalable to send the entire request to the VOMS server to handle all this site specific policies. The VOMS may have actually issued the certificate and may result in more complexities. To handle local authorization EDG has a Local Central Authorization Service (LCAS). The concept of LCAS is that different authorization modules may be incorporated to provide additional flexibility. LCAS is able to make authorization decision based on request resources the identity of the requestor the VOMS credential and the proxy certificate. Once the access control or local authorization decision has been made the grid credentials need to be mapped to the local fabric or unique operating system through a Local Credential MAPping Service (LCMAPS). In addition to these different components there is also a Job Repository (JR) which maintains a record of the credential information associated with all jobs running inside the operating system. Now as mentioned earlier the job submitted to the local site may be queued which may result in a large latency for the job. Here the credential renewal service plays its part. It contacts the MyProxy credential manager with the renewal request and the old proxy certificate. It may also contact the VOMS service with a certificate to obtain attribute credentials. It then supplies the credential and the new proxy certificate to the job manager in the local centre. This takes care of the expedition of the proxy certificate.

12.2 An Enterprise Case Study

In this section, we will look at a typical example of an enterprise grid computing system. The enterprise we are looking at here is a financial services company having offices in US, Europe, and India. Before the integration of grid, the IT infrastructure looked as follows:

- There were different clusters across the globe. The clusters were provisioned at their peak usage and hence grossly under utilized.
- The enterprise had spent considerable effort in consolidating the user identities through Windows Active Directory . However, users in different clusters were using multiple user accounts.
- The workload in the clusters was mainly composed of highly compute intensive jobs like the credit analysis type of job and the long running batch kind of applications. Since most of the batch jobs were running at night resource sharing across geography was considered a very effective mechanism of increasing resource utilization. Hence, grid was the obvious choice.
- In addition to in-house users, the enterprise allowed external users to run jobs on the statistical models. Though the usage and workload of external jobs were significantly less than in-house jobs, separate clusters were used to cater to those, mainly for security purposes.

The requirements that the enterprise had in moving towards a grid-based system are as follows:

- A centralized identity management system that tracks the users submitting jobs to different clusters. It should be combined with the centralized monitoring mechanism which would be able to track usage per department and groups. The above information would also be used to price external users.
- The enterprise would also like to use the huge pool of enterprise desktops mainly for the batch jobs. Since the jobs would be coming from different departments, the desktops need to have sandboxing mechanisms.
- Elaborate policy mechanisms are needed to bind the users, applications, resources, and different metrics.
- Since the enterprise had spent a significant effort towards integrating Service Oriented Architecture (SOA), all solutions should be compatible to SOA standards.
- There are multiple grid/cluster interfaces for submission of different applications. There should be a single sign on and authentication mechanism for all the different interfaces.

12.2.1 Overview of the Security Architecture

Figure 12.2 shows the high level architecture of the grid security solution for the enterprise. The main components of the architecture include: *authentication system, perimeter defense system*, *authentication and authorization system*, *monitoring system*, *intrusion detection system, SLA manager, local access controller*, and *host data protection system.*

Perimeter Defense

Perimeter defense is provided by controlling the incoming traffic through XML firewalls. The Web services compliant request necessitates XML firewall based requirements. The system is able to parse the XML and SOAP messages and makes sure that the XML is in order and no XML based attacks are possible. At this point, different policies are checked before finally allowing the job to enter the grid system. Policies can vary from the location and port of the acceptable request to the type of acceptable applications, the credentials attached and so on.

Authentication System

As mentioned earlier, one of the requirements of the system was the interoperability of the security mechanisms with all the existing interfaces of the enterprise. A design of the system is based on a centralized authentication system which validates the user credential and sends an authentication token back to the grid entry point. The grid entry point can be any interface which redirects the request to the centralized authentication system. The authentication token is signed by the authentication system and hence can be verified at any point. The authentication system currently supports password and X.509 tokens. It is to be noted that there are several other authentication systems which are still active for partners and departments who authenticate the user themselves. To cater to those types of job requests, SAML is used as an authentication token. The external interfaces submit the authentication token along with the jobs, which is revalidated by the grid authentication system.

Authorization System

The authentication token generated by the grid authentication system is passed on to the authorization system for generating the authorization tokens. The first point of contact with the authorization system is the Policy Decision Point (PDP). The PDP sends the authentication token to the centralized authorization system. The authorization decision is based on the policy information stored in the policy database. The authorization system verifies the authentication token and then consults the policy database and creates a SAML token based on the policies. The token contains information which binds the resources with the roles and the applications that run on these resources. For example, a policy statement can state that role R is entitled to run jobs on machines X, Y, and Z. Similarly, policy information can also state that: application A has x number of licenses, or application B is only installed on machines X1, Y1, and Z1, and so on. Another important component of the authorization system is the identity manager which contacts the centralized identity directory to get the identity information. The policy database is updated manually by an administrator as well as through an identity manager service.

Monitoring and Logging

Another important component of the security architecture is the monitoring and logging system. Each node in the grid infrastructure has a monitoring agent which reports the information about the grid node. The monitoring system is very similar to Ganglia (discussed in Chap. 11). Information sent by the agents are CPU utilization, memory, current status of the jobs, SLA status of the jobs, etc. The information obtained by the agents is mostly transient in nature. A hierarchical architecture is provided for a scalable retrieval and access of monitoring information. The information is used for three different purposes. Firstly, the information is placed in a Monitoring Database (MDB) which are used for auditing purposes. An extensive interface is provided for querying the information stored in the MDB. Secondly, the information is used for better scheduling as the run time information can be used to restrict the scope of the scheduler. Finally, the information is sent to the SLA manager for managing SLAs. The information is also used by the Intrusion Detection system for predicting malicious activities in the system.

Intrusion Detection System

Since a shared grid is used to cater to both internal and external job traffic a preliminary form of Intrusion Detection System (IDS) became necessary. The system is based on anomaly detection and detects the anomalous patterns in the system for raising alarms by looking at the information provided in MDB. If a job runs for significant amount of time, it is a cause of concern. Similarly, if the CPU utilization is extremely skewed ranging between 90%-100%, then an alarm is raised. The system behavior is learned by taking into account the different parameters over a period of time.

SLA and Trust Manager

Another interesting component in the security architecture is the SLA manager which creates a contract for all jobs through a WS-SLA. Based on the information obtained from the monitoring agents decision is made whether to transfer the job to some other node in case of SLA violation. Each node also has a trust rating associated with it. In the current implementation, the trust metric is a simple average based on the number of times the SLAs are met which is normalized to 1. Once a node fails to honor its SLA, its trust rating is downgraded. During scheduling of jobs, the trust rating of a node is taken into account. The dynamic trust information of a node is put in a trust table which is part of the MDB.

Local Access Controller and Host Data Protection

The centralized policy database stores long-term policy decisions. However, different clusters in different departments may have local policies. The local access controller denies suspended users the access to the grid resources. Some other access control policies can also be implemented in the local system. The data protection within a host is provided using the virtualization system which creates multiple Virtual Machines (VM) within a single physical machine. The local access controller also interacts with the Virtual Machine scheduler which submits the jobs to the virtual machines within a grid node. Xen (discussed in Chap. 7) is used to create the virtual machines within a grid node. Creating of VMs has three-pronged advantages. Firstly, the data protection can be provided through isolation. Secondly, the SLAs can be guaranteed in a more stringent manner as CPU and memory can be allocated precisely for a particular application. Finally, VMs provide a better utilization of resources.

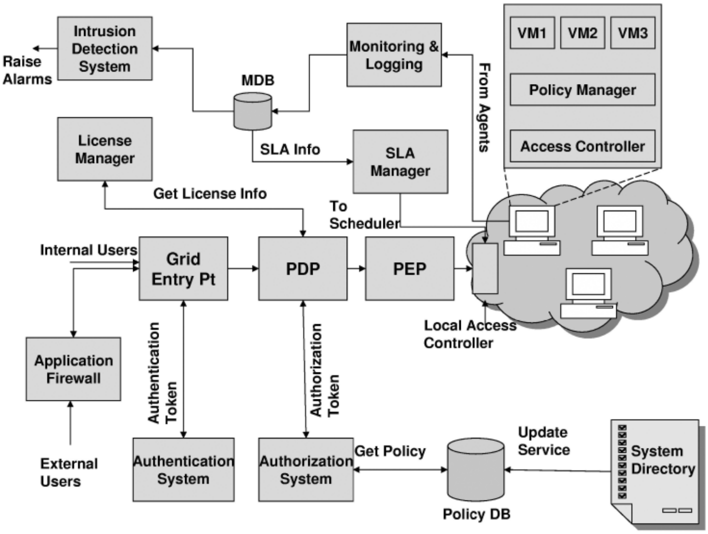


Fig. 12.2. Overview of the security architecture

12.3 Chapter Summary

In this chapter, we have looked at two interesting case studies which integrate the different concepts discussed as course of the book. The first case study is from the European Data Grid (EDG) where GSI is used as information security architecture and VOMS is used for authorization purposes. The main problem handled in this case study is authenticating and authorizing huge number of users of the EDG system. The second case study is that of a financial services firm, where the security problem lies in integrating security standards across different clusters spread across the geography, identity management, and SOA. The chapter explains the case study in detail.

In the next chapter, we will conclude the book by providing a few new technologies which could be useful in the long-term and mapping the different issues into immediate, medium-term, and long-term categories.

13 Conclusion

13.1 Looking at the Future

In this section we will look at some of the technologies that may have an impact on the grid security landscape in the future. One of the technologies that we will be looking at is Identity Based Encryption (IBE) which is a new cryptographic technique where encryption can be done using any known string associated with the receiver. We will also be looking at application oriented networking which is an exciting future technology.

13.1.1 Identity Based Encryption (IBE)

One of the most interesting advancements in the field of cryptography came from Dan Boneh and Matthew Franklin from Stanford [261], published in 2001 – it is called Identity Based Encryption (IBE). IBE is the solution to the problem floated by Shamir in 1984 [262]. The problem is as follows: Based on a set of global system parameters and a fixed master key, the problem is to generate a set of private keys corresponding to any set of public keys. The public keys would be used to encrypt messages which can only be decrypted by the corresponding private key. Let us take an example to understand the problem better. Alice wants to send a message to Bob using Bob's email address bob@nobody.com. Let ***G*** be the set of global parameters known to everybody including Alice and ***M*** be the secret master key. Let ***P*** be the private key corresponding to the string bob@nobody.com. It is to be noted that different private keys can be generated corresponding to different strings. Once the private key is generated the message can be encrypted by the string and decrypted using ***P***. This problem was solved by Boneh and Franklin based on the bilinear maps between groups. They showed that Weil pairing on elliptic curves is an example of such a map and implemented the system based on the pairing. The authors have proved that the system thus implemented has very strong security properties. The inventors had also floated a company called Voltage Systems $^{®}$[263] which implements the IBE based solution.

At this juncture, the readers may question the usefulness of such a scheme. One of the main advantages of the IBE scheme is that it frees the sender from having to obtain the public key of the receiver. In the case of IBE, the sender can encrypt with any string that can be associated with the receiver. For example, the email address or IP address can be a string that anyone can associate with the receiver and hence there is no need to obtain or store the public key of the receiver. This becomes important in a bandwidth constrained environment where getting public keys through certificates may result in a lot of bandwidth wastage. The authors have presented several situations where such a scheme can be very useful:

- **Public Key Revocation:** Public key certificates contain a preset expiration date. In an IBE system key expiration can be done by having Alice encrypt message sent to Bob using the public key: "bob@nobody.com||current-year." In doing so Bob can use his private key during the current year only. Once a year Bob needs to obtain a new private key from the trusted private key generator.
- **User Credential Management:** IBE system can also be used to generate user credentials. Alice encrypts the message with the following string: "bob@nobody.com||current-year||clearance." Bob would be only able to decrypt the message if he has the correct clearance i.e., private key for that string.

The IBE system described above can be very useful to a grid infrastructure in a constrained environment where security is required. An example of such a system would be a sensor grid. Using an IBE system, managing credentials and public keys would be possible in a much less expensive manner. However, it is to be noted that such a system is still in research and not being deployed yet. Another critical point about such an infrastructure would be storage of the master key as the security of the whole system hinges on that. In case the master key is compromised all the private keys need to be re-generated.

13.1.2 Application Oriented Networking (AON)

Another important technology that is gaining momentum is called Application Oriented Networking (AON). Traditional networking systems route packets to destinations solely by looking at the packet headers of incoming traffic. Application artifacts, integration applications or SOA, are wholly software based and have redundant usage of XML based operations and routing. Therefore, moving a lot of these operations over to the network layer may result in more *flexibility* by allowing disparate applications to communicate, better *consistency* in applying uniform security policies, better *visibility* and *monitoring* of information flow across the network, and better application *optimization* as redundancy of XML operations can be reduced with better load balancing capabilities. Application Oriented Networking (AON) is a step forward towards achieving that vision.

AON enabled devices have the capability to look into the packets and understand the contents before forwarding it to the next router or networking device. AON can not only route based on XML messages but it also has the capability to verify XML data integrity. It also offers message level security such as digital signatures, encryption, etc. to make the transfer over the network more secure and reliable. Most of the AON products and devices integrate over the existing routers and switches to provide the extra application level services. In terms of security, AON devices can interact with Web Services standards like WS-Security and other existing standards like Kerberos, OpenLDAP, etc.

One of the most important challenges in grid systems is the heterogeneity present in the infrastructure and ways to handle such heterogeneity. AON provides a mechanism to handle such heterogeneity in the networking infrastructure itself. Therefore, AON provides the grid with another option in integration that may reduce the redundancy at several levels and hence result in increased efficiency. AON products from Cisco® [264], DataPower® [265], etc. are already out in the market. Extensive studies are needed about how these products and solutions can be used in the overall secure grid infrastructure.

13.2 Summarizing the Security Issues in Grid

In the different chapters of this book, we have looked at security issues at different levels of grid computing stack. This section is our attempt to categorize issues in terms of *immediate*, *medium-term*, and *long-term*. As the name suggests, immediate issues require immediate attention and solution, probably within the next twelve months, medium-term issues need solutions in a 1-3 year timeframe, while long-term issues eventually need to be solved, probably in the next five years.

13.2.1 Immediate Issues

Let us now look at the current enterprise scenario to understand the security needs. Enterprises are looking at grids mostly from the resource sharing point of view. As illustrated in the case study in Chap. 12, most of the enterprises requiring compute intensive jobs have invested heavily on clusters. Most of these clusters are either managed by proprietary cluster managers like LSF Multicluster, PBSPro, Data Synapse® GridServer, or even open source systems like Condor. When the issue of resource sharing comes, the enterprise either moves completely to a proprietary solution like LSF Multicluster or Data Synapse or comes up with some adhoc way of integrating the different components. Rarely does one find the usage of the Globus toolkit for this purpose, mainly because of the enterprises' reluctance to go with open-source software which nobody is using. We do however find some pilots undertaken in this direction. With Univa® taking up the role of evangelizing, marketing, and developing Globus based solutions, in the near future we do expect to see more Globus based enterprise level grid solutions. However, these do not figure in the immediate concerns for enterprises. For enterprises, integration with disparate systems, identity management, authorization, and basic authentication are the immediate concern areas.

Integration

Perhaps the most important issue that the enterprises are grappling is that of integration. Grid system should work with the other components of the enterprise. There may be need to integrate with Kerberos based system in the enterprise or with the enterprise specific policies. Many of the already available components can be extended to provide the needed support. Service Oriented Architecture (SOA) based approach is also needed for enterprise wide adoption and standardization. We feel that for immediate need, ad-hoc solutions and customized integration may go a long way in solving the integration issues. In the long run, a Globus like standardized approach over SOA can be the ideal way to go forward.

Identity Management and Monitoring

Another issue closely related to the integration issue mentioned earlier is the issue of managing identities in the grid system. Many of the large enterprises have invested on hundred of clusters across the globe. Each of the clusters are well-managed, however there is very little management across clusters. Therefore, we find one user possessing multiple identities in the different clusters. This becomes a nightmare in monitoring and tracking the usage of users or departments, as redundancy of information is available resulting in complexity in collating the information. Added to this is the fact that most of the enterprises possess some kind of user directory like Active Directory or LDAP where the identities of the users of the enterprise are stored. Therefore, there is a need for integration of the enterprise user identities with the enterprise wide identity management systems. Many of the proprietary systems like LSF Multi-cluster already allow this type of integration. Regarding monitoring, most of the enterprises either have open source systems like Ganglia, Hawkeye, or other proprietary systems. Therefore, there is a need for integration of the identity management systems with the monitoring systems for department-wise and user-wise reporting. In the long-term, many organizations are also looking at Liberty framework for identity management and integration.

Authentication and Authorization

Some amount of authentication of users and policy based authorization is needed for an effective management of the grid system. Most of the grid-based implementation uses simple authentication like password based authentication or no authentication at all as most of the grid systems are considered to be safe as they are within the organization premises. However, a solution which integrates the enterprise authentication system like the Kerberos and the grid systems would be useful. A WS-Security based solution would be greatly appreciated by the enterprise community, though rarely undertaken due to the scale and cost involved.

Authorization is another important area where efforts need to be carried out to provide an enterprise wide solution. Most enterprises have very complex business policies and structure which need to be taken care before execution of jobs. At present, most of the enterprises are developing security and business policy as layers above the scheduler so that jobs which qualify after being subjected to verification and validation are only submitted to the scheduler for execution. This is necessitated by the fact that most of the policy managers are inadequate in addressing the needs of the enterprises.

13.2.2 Medium-term Issues

Let us now look at the medium-term issues in grid security which would impact the security landscape of the enterprises in the 1-3 year timeframe.

One of the steps that enterprises are going to take in this time-frame is to incorporate the different hosts in the grid infrastructure. Hence the issue of host-level security would become very important in that case. Probably, towards the end of this three year period, one may see the adoption of grid standards and Globus becoming more and more commonplace as an enterprise level grid solution. Furthermore, architecture level issues discussed in Chap. 4 would become more intense and important. Management of credentials would become more important as more and more applications would be using grid as the computing infrastructure. Some aspects of networking like firewalls, VPNs, etc. would also gain prominence as the users would be using other networks to use the grid infrastructure.

Host Level Security

Many enterprises are looking at including desktops into the grid infrastructure. The integration results in two types of concerns: Firstly, the data and applications in the hosts need to be protected resulting in the need for some type of sandboxing for the grid jobs on the host. The second concern is regarding the development of better security policy manager to handle the complexity. Most of the sandboxing solutions implemented by the enterprises are those provided by grid middleware systems like GridMP from United Devices®. However, several enterprises have started experimenting with proprietary virtualization solutions like VMWare®. However, the current trend suggests that many enterprises would be willing to experiment with open-source Xen as a virtualization solution which would also provide them with sandboxing options. Please refer to Chap. 7 for details of the different sandboxing solutions available.

Credential Management

In the medium term, the importance of credential management systems also would be growing. The main reason for that is the projected growth of grid based applications and usage in the enterprises. There would be a need for storage for different types of credentials, transporting them, regenerating them, and integrating them with the existing identity management systems.

Networks

With the growth of grid-based infrastructure in the enterprises, there would be need for incorporating the basic networking infrastructure like firewalls and VPNs with the grid infrastructure. As mentioned in Chap. 8, there are a plethora of issues that need to be solved before grid systems can easily inter-operate with these networking technologies. With the growth of interconnect technologies, enterprises would also be looking at secure transfer at high rate towards the end of the three year time period. This would find adoption in the enterprises given the research in this area could produce solutions that can be deployed and used. The other networking issues like multicasting, sensors, etc. can only be viewed as long-term and futuristic issues.

13.2.3 Long-term Issues

Let us now look at the long-term issues in grid computing security, the issues affecting the enterprises probably in the three-five year time period. During this time, we would probably see multi-enterprise level grid systems, where different enterprises come together to form a single grid. Moreover, during that time grid hosting services would also be in great demand. Given this vision, service level security would gain tremendous importance at that time. Denial-of-service and QoS attack prevention can take precedence over other types of solutions during that time. Multi-enterprise grids also would require a trust management system in place. Grid multicasting and sensor grid security would not be important until towards the end of the five year timeframe.

Service Security

The long term vision of grid security should capture the issues pertaining to managing the service level issues like preventing denial-of-service and prevention of QoS violation. Research is active in these areas and based on the result of the research, enterprises would be adopting them as part of the grid infrastructure.

Trust Management

In a controlled grid environment, trust can be managed using static configurations. However, with the growth of flexible grid systems, the need for managing trust would become more and more important. Please refer to the several trust systems described in Chap. 10. However, those systems would not be deployed in the enterprises until towards the end of the five year time period. More research also needs to be carried out for managing trust across disparate systems.

Multicast and Sensors

Multicast provide optimized use of bandwidths, and enterprises would be deploying them in a few years time as they are finding the transfer of data as one of the biggest bottlenecks facing the enterprise grid systems. In the case of selected broadcast, multicast technologies can provide a better bandwidth usage and hence higher throughput. However, multicast security may not be an immediate concern for the enterprises mainly because of the complexity involved. Sensor network based grid systems would also see some deployment in the next four to five years. However, security may not be the primary concerns there also because of the complexity involved in security the tiny sensor devices. Both these areas have a vibrant research community and positive outputs from research would surely see deployment towards the end of the five year time frame. The above discussion is summarized in Fig. 13.1.

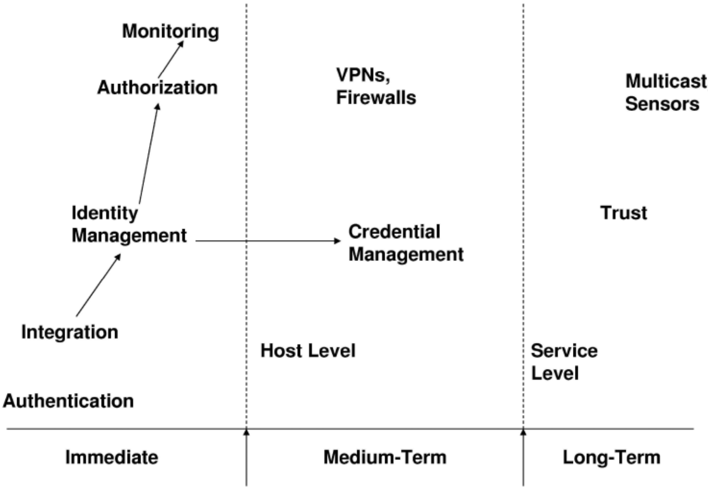


Fig. 13.1. Time frame of different security issues

13.3 Summarizing the Security Solutions in the Grid

In this section, we will try to analyze the solutions and map them to one of the three security issues mentioned above. We will also try to analyze the gaps existing which need to be addressed.

13.3.1 Solutions to Immediate Issues

As mentioned earlier, the immediate issues are the *integration issues*, *identity management issues*, *authentication*, *monitoring*, and *authorization* issues. Integration is not a single issue, it spreads itself to all the different aspects of the grid system. Mostly, the solution to integration problem is handled in a on-to-one basis and in an ad hoc manner. For example, if there is a need for integrating the identity management solution of the enterprise with the grid middleware it is done by building a layer on top of the middleware to handle the problem. Therefore there is a need for enterprise-wide standards based approach to integrate the grid specific solutions with the enterprise products and mechanisms. For identity management, authentication, monitoring, and authorization, stand-alone solutions exist. Therefore, integration with the grid specific requirements is the need of the hour. For example, identity management in enterprises are carried out using standard products which currently do not integrate with grid systems, therefore a gap exists. Similarly, GSI is an extensive solution, however cannot be employed directly in enterprises due to the inter-operability issues. Therefore a standards based approach is needed. Adoption of WS-Security is a step in the right direction. Similarly, several monitoring tools, techniques, and protocols exist which monitor different elements of the infrastructure. For example, SNMP is a standard protocol for network monitoring, Ganglia and Hawkeye are extremely powerful tools to measure cluster performance. Therefore, integration of the grid specific monitoring tools like MDS, R-GMA needs to be integrated with the system and cluster specific tools. It is to be noted that integrations have been achieved to solve specific problems. However, there is lack of a holistic integration solution available in system and cluster monitoring. A standardized approach is needed in this area. Similar observations can also be made about authorization solutions. Need for integration of resource level authorization solutions and Virtual Organization (VO) level authorization solution is needed. Some specific integration solutions like CAS and PERMIS have been carried out in a controlled manner. A more integrated and holistic solution is needed here. Table 13.1 summarizes the immediate issues.

Table 13.1. Immediate issues

Issues	Solutions	Comments
Integration	Integration solutions are ad hoc and specific in nature	More complete integration solution is the need of the hour, standardized efforts are needed
Identity Management	Identity management solutions are ad hoc and specific in nature	Need for integration of specific identity management systems with enterprise Identity Management systems
Authentication	Autentication using standard technologies like Public key Cryptography, GSI is an example	Most of the authentication solutions are available, GSI needs to be integrated with existing technologies and standards
Monitoring	System Based like Orca Cluster based like Ganglia Grid based like R-GMA, MDS	Grid based systems should be able to work with different system based and Cluster based monitoring systems
Authorization	VO Level Resource Level	Integration is again an important issue here. Need for integration of VO level and Resource level authorization solutions.

13.3.2 Solutions to Medium Term Issues

Next we will discuss the solutions to the medium term issues. Integration of grid solutions with VPNs and firewalls, credential management, and solutions to the host level issues are discussed here. Currently, all VPNs and firewalls need to be statically and manually configured. Extensive research is needed to integrate the existing VPNs and firewall techniques. Credential management is another issue that will become critically important in medium term. Several standalone solutions like MyProxy exist mainly as repositories for credentials. The Liberty Alliance type solution which integrates different credential types and systems can be used to integrate with the credential repository systems. More prototypes and research efforts are needed in this direction. Host level issues also are becoming critical for enterprise grids. Virtualization solutions provide good and powerful solutions to this problem. Research is needed to evaluate the performance of the different isolation solutions with respect to the application and infrastructure patterns. Table 13.2 summarizes the medium term issues.

Table 13.2. Medium term issues

Issues	Solutions	Comments
VPN Firewalls	Integration with Grid platforms is either manual or fixed.	Research is needed for providing automated and flexible management of VPNs and Firewalls
Credential Management	Credential Repositories like MyProxy, Smart Cards Credential Federation Systems like Liberty Alliance, Shibboleth	Integration is needed for Credential Repositories with Federation Systems.
Host Level – Data Protection	Application Level Sandboxing (Proof Carrying Code) Virtualization Flexible Kernels Sandboxing	Virtualization is a scalable and powerful solution. More research is needed for policy integration with different virtualization and sandboxing solutions.
Host Level – Job Starvation	Resource Reservation Priority Reduction	Resource reservation support is needed for most middlewares

13.3.3 Solutions to Long Term Issues

Long term issues include service security, trust management, and networking issues like multicasting and sensor grids. All these issues are research intensive and require extensive research and development efforts. The issues will only become important in the long term and hence there is some time for research efforts to materialize. The solutions to service level issues can be categorized into preventive, reactive, and QoS solutions. Most of these solutions can be inherited from the Networking and OS domain.

Currently, most trust management solutions are policy-based. Reputation-based solutions have huge potential. Therefore, active research is needed to develop more reputation-based solutions and integrate them with the policy-based ones. Multicasting and sensor networks are nascent areas and active research is currently underway. Table 13.3 summarizes the long term issues.

Table 13.3. Long term issues

Issues	Solutions	Comments
Service Security	Preventive DoS Solutions Reactive DoS Solutions QoS Solutions	More research is needed for developing the different service level solutions
Trust Management	Reputation based Policy based	Most of the implemented trust management solutions are policy based. Extensive research is needed for developing effective Reputation based solutions
Secure Multicasting & Sensor Grids	Multicast solutions like Tree based Key Management Sensor Solutions like SPINS and TinySec	Both the areas require extensive research before deployed as security solutions. In Multicasting, key distribution and management is a very important issue. In case of sensor networks, managing collusion and Sybil attacks is an important area of research.

Appendix

A.1 Web Services

Before going into the details about Web services one small story comes to my mind. At that time I was in United States doing my graduate studies and my passport had expired. I knew the process of passport renewal in India. However, being in a foreign country I had no idea about how complex the process was. Since I had no other choice, I went to the Indian consulate Web site, saw the instructions and was relieved to find that it was possible to renew my passport from the United States. I got a list of consulates from the Web site and selected Chicago as it was the nearest to Iowa State University where I was doing my graduate studies. From there I could understand the steps or protocols involved in getting my passport renewed. I followed the steps and got a new passport. At this point, the readers may wonder about the significance of this story. With the advent of the Internet, this is exactly what people do now-a-days. However, could anyone imagine what I would have done if the language of the Web site had been anything other than English, or I could not find the list of consulates, or I could not understand the steps mentioned in the web site? Therefore, to obtain any service, the service should be based on four basic principles: Understandable *language*, a *directory* which lists the service locations, a *description of the service*, and an understandable *protocol*. When we go back and analyze my actions during my passport renewal, it is quite clear that the consulate did follow these principles and resulted in my being satisfied with the service that I received. When we subsequently look at Web services, we find that it has these four principles embedded in it. These principles, in my opinion, result in making Web services simple and intuitive which helped in its popularity.

A.1.1 Components of Web Services

The last couple of decades has seen numerous implementations of distributed computing like CORBA [266], Java RMI [267], DCOM [268], etc. None of these systems were taken up in a big way by the industries mainly because of their tightly coupled nature. Current trends in the application space suggest that enterprises are moving away from monolithic tightly coupled systems towards loosely coupled dynamically bound components. With the growth of the Internet as a premier means of communication, a new paradigm called Web services [269] emerged. Web services can be thought of as reusable, loosely coupled software components which are deployed over the network or specifically the World Wide Web. There are some advantages which the experts claim as the major reasons for the adoption of Web services as a de facto standard for application integration. These are:

1. **Simplicity:** Implementation of Web services is very simple from the point of view of programmers and as a result easy and fast deployments are possible. All the underlying technologies and protocols are based on the Extended Markup Language (XML) [270] which is simple and intuitive.
2. **Loosely Coupled:** Since the very design of Web services is based on loose coupling of its different components, they can be deployed on demand.
3. **Platform Independent:** The Web services architecture is platform and language independent since they are based on XML technologies. Therefore, one can write a client in C++ running on Windows, while the Web service is written in Java running on Linux.
4. **Transparent:** Since most of the deployed Web services use HTTP [271] for transmitting messages, they are transparent to the firewalls which generally allow HTTP to pass through. This may not always be the case for CORBA, RMI etc.

According to many experts, CORBA and RMI provide a much better alternative to Web services because of the flexibility and features that CORBA provide. Moreover, performance wise the CORBA RMI combination may be better than protocol designed over HTTP. However, because of its simplicity and backing of the big commercial vendors Web services are steadily becoming a standard which none can ignore. There are many forums where debates are being pursued as we move on to the different components which constitute the Web services. There are three main components of Web services:

- **SOAP:** The Simple Object Access Protocol (SOAP) [272] is a lightweight protocol for exchange of information between diverse and distributed computing environments. It combines the extensibility and portability of XML with the ubiquitous Web technology of HTTP. It provides a framework for defining how an XML message is structured using rich semantics for indicating encoding style, array structure, and data types.
- **WSDL:** The Web Service Description Language (WSDL) [273] can be used to describe a Web service, providing a standard interface. A WSDL document is written in XML and describes a service as a set of endpoints, each consisting of a collection of operations. XML input and output messages are defined for each operation and their structure and data types are described using an XML schema in the WSDL document. The WDSL and XML schema provide a complete definition for the service interface, allowing programmatic access to the Web service, in the manner of an API. Tasks like data requests or code execution can be performed by sending or receiving XML messages using, for example, SOAP.
- **UDDI:** The Universal Description, Discovery and Integration (UDDI) [274] specification defines a way to publish and discover information about Web services. It is collaboration between Ariba®, IBM®, and Microsoft® to speed inter-operability and adoption of Web services. The project includes a business registry (an XML document) and a set of operations on it. The registry can be used by programs to find and get information about Web services and check compatibility with them, based on their descriptions. UDDI allows categorization of Web services so that they can be located and discovered and WSDL enables a programmatic interface to a service once it has been located.

Let us now look at how the different technologies and protocols which constitute the Web services through a simple example.

1. As mentioned before, a client may have no knowledge of what Web service it is going to invoke. So, the first step will be to *find* a Web service that meets the client's requirements. For example, the

client may be interested in finding the list of Indian consulates in the United States. The client will do this by contacting a UDDI registry.

2. The UDDI registry will reply, telling the client the location of servers which provide the information about the Indian consulates in United States.

3. The client now knows the location of a Web service, but it has no information about how to actually invoke it. It knows that the Web service inputs the name of the US state and returns the nearest Indian consulate. The client does not know the input types and other information about the service that the Web service is going to provide.

4. The Web service replies in a language called WSDL which allows the client to programmatically invoke the Web service.

5. The client now knows where the Web service is located and how to invoke it. The invocation itself is done in a language called SOAP and mentioned earlier, which is built on top of HTTP. Therefore, the client will first send a *SOAP request* asking for the nearest consulate to a certain US state.

6. The Web service will reply with a *SOAP response* which includes the name of the city where the consulate is located, or maybe an error message if the SOAP request was incorrect.

A.2 Web Services Security

Security is important in any distributed systems. However, in Web services security attains enormous proportions because of the following reasons:

1. The boundary of interaction between communicating partners is expected to expand from intranets to the Internet. For example, businesses increasingly expect to perform some transactions over the Internet with their trading partners using Web services. Obviously, from a security perspective, Internet communication is much less protected than intranet communication.

2. Communicating partners are more likely to interact with each other without establishing a business or human relationship first.

This means that all security requirements such as authentication, access control, nonrepudiation, data integrity, and privacy must be addressed by the underlying security technology.

3. More and more interactions are expected to occur from programs to programs rather than from humans to programs. Therefore, the interaction between communicating partners using Web services is anticipated to be more dynamic and instantaneous.

4. Finally, as more and more business functions are exposed as Web services, the sheer number of participants in a Web services environment will be larger than what we have seen in other environments.

Secure Socket Layer (SSL) or Transport Layer Security (TLS), described in Chap. 2, has become a standard in transport layer security. What is the necessity for developing a new set of security standards rather than use SSL as transport layer security? There are a few reasons which limit the development of Web services on SSL alone. They are:

1. SSL is designed to provide point-to-point security, which falls short for Web services because one needs end-to-end security, where multiple intermediary nodes could exist between the two endpoints. In a typical Web services environment XML-based business documents route through multiple intermediary nodes. In such a scenario, it proves difficult for those intermediary nodes to participate in security operations in an integrated fashion. Therefore, actions such as authorization, nonrepudiation, auditing, etc. are difficult in such a scenario.

2. SSL secures communication at transport level rather than at message level. As a result, messages are protected only while in transit on the wire and unprotected at the end points. For example, sensitive data on the hard disk drive of a HTTPs end point is not generally protected unless there exists specific mechanisms to protect the data

3. SSL or HTTPs in its current form does not support non repudiation well which is critical for many business transactions. Non repudiation means that a communicating partner can prove that the other party has performed a particular transaction. For example, if an online trading company *A* received a stock transaction order from one of its clients and performed the transaction on behalf of that

client, A would want to ensure that it can prove the completion of the transaction to an arbitration committee if a dispute arises. There is a need some level of nonrepudiation for Web services-based transactions.

4. Finally, SSL does not provide element-wise signing and encryption. For example, if one has a large purchase order XML document, one could want to only sign or encrypt a specific element say SSN or credit card number. However, signing or encrypting only that element with SSL proves rather difficult. This is due to the fact that SSL is a transport-level security scheme as opposed to a message-level scheme and the whole content is encrypted reducing the flexibility.

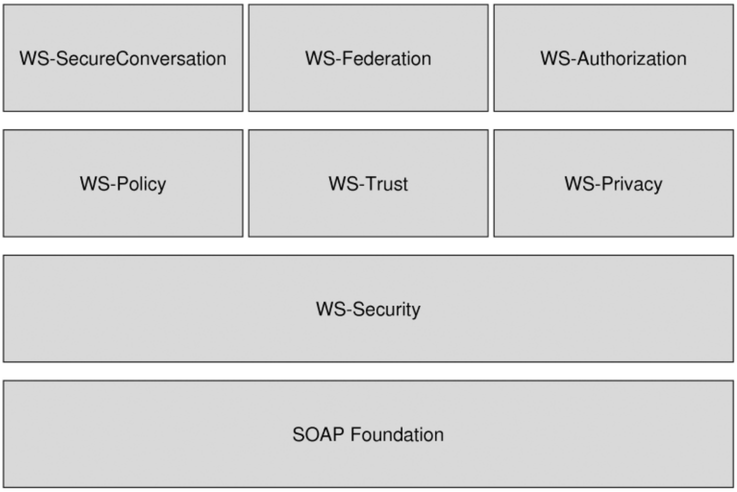


Fig. A.1. Web services security architecture

In Fig. A.1, different components of Web services security are shown. In the next section a brief primer will be provided for different security standards like WS-Security, WS-Policy, and WS-SecureConversation. In addition some basic information will be provided about policy languages like Security Assertion Markup Language (SAML) and Extensible Access Control Markup Language (XACML).

A.2.1 WS-Security

WS-Security [275] is a set of standards that describe the security mechanisms in a Web services scenario through the extensions of SOAP header to provide message integrity and confidentiality. WS-Security is flexible and is designed to be used as the basis for the construction of a wide variety of security models including Public Key Infrastructure (PKI), Kerberos, and SSL. Specifically WS-Security provides support for multiple security tokens, multiple trust domains, multiple signature formats, and multiple encryption technologies. It provides mechanisms for propagation of security tokens, message integrity and message confidentiality. These mechanisms by themselves do not provide a complete security solution. Instead, WS-Security is a building block that can be used in conjunction with other Web service extensions and higher-level application-specific protocols to accommodate a wide variety of security models and encryption technologies.

Following are the requirements which drove the adoption of WS-Security as a Web Services security standard:

- Multiple security tokens like username password based, X.509 certificates, Kerberos tickets, SAML assertions, and so on;
- Multiple trust domains;
- Multiple encryption technologies;
- End-to-end message-level security.

In the Fig. A.2, a WS-Security message structure is shown. It consists of a SOAP envelope which specifies how to route the message. Then it contains the <Security> header which contains the security information for an intended receiver which may be the ultimate receiver or an intermediary. The header consists of a security token which is associated with the message.

As mentioned in Fig. A.2, WS-Security allows two types of security tokens in the message, one is a simple username and password and the other is a binary security tokens which can be of the form of Kerberos tokens and X.509 certificates. Message integrity is provided by leveraging the XML signature in conjunction with security tokens to ensure that messages are transmitted without modifications. The integrity mechanisms are designed to support multiple signatures, potentially by multiple actors, and are extensible to support additional signature formats. Different algorithms as specified by [276] are allowed.

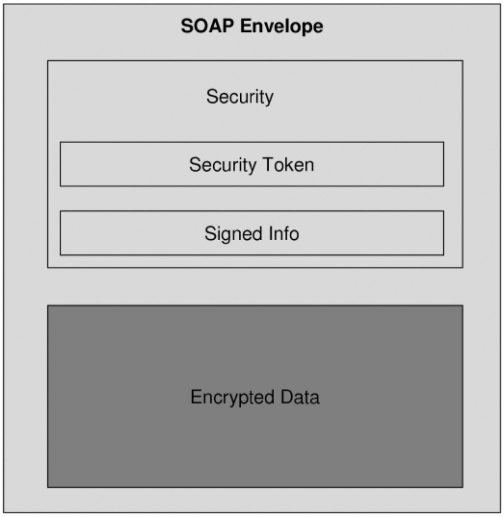


Fig. A.2. WS-Security message structure

The WS-Security specification allows encryption of any combination of body blocks, header blocks, any of these substructures, and attachments by either a common symmetric key shared by the sender and the receiver or a key carried in the message in an encrypted form. In order to allow this flexibility, WS-Security leverages the XML encryption [277] specifications. When a sender or an intermediary encrypts portion(s) of a SOAP message using XML encryption they will add a sub-element to the <Security> header block. Furthermore, the encrypting party then should prepend the subelement into the <Security> header block for the targeted receiver that is expected to decrypt these encrypted portions. The subelement should have enough information for the receiver to identify which portions of the message are to be decrypted by the receiver. This ability of WS-Security to provide partial encryption scores a big point over security provided at the transport level like SSL.

A.2.2 WS-Policy*

WS-Policy* define the policy framework for Web services. These include: *WS-Policy*, *WS-PolicyAttachment*, and Policy languages which include *WS-PolicyAssertions*, and *WS-SecurityPolicy*.

WS-Policy

WS-Policy [278] provides a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML Web services based system. WS-Policy provides a generic framework and a model for the expression of these characteristics and requirements as policies. It defines a policy as a collection of policy alternatives where each policy alternative is a collection of policy assertions. WS-Policy supports different requirements and capabilities like authentication systems, transport protocol, privacy policies, QoS characteristics, etc. WS-Policy does not specify how policies are discovered or attached to a Web service, which is generally done using WS-PolicyAttachment.

In the top level of the WS-Policy structure we have the policy expressions identified by <Policy> which describe the combination of assertions. Inside the policy expressions, operators are defined which describe the semantics of combination of assertions. These operators can be *All*, *OneorMore*, and *ExactlyOne*, which indicate the combination of assertion allowed by the Web services. In a WS-Policy structure, multiple assertions can also be included and the operators work on the combination of these assertions.

WS-PolicyAttachment

The WS-PolicyAttachment [279] specification defines mechanisms for associating policies with the subjects to which they apply. The policies may be defined as part of the existing metadata about the subject or the policies may be defined independently and associated through the external binding to the subject. The policies defined as part of the WS-Policy with one or more policy subjects. WS-PolicyAttachment allows associating a policy with a policy subject through UDDI, WSDL, or through end point references. It also allows mechanisms for signing the attachments to prevent tampering.

WS-SecurityPolicy

WS-SecurityPolicy [280] identifies the basic set of policy assertions for security. Different features supported by it are:

- **Message Integrity/Confidentiality:** Senders of a message can make use of the integrity/confidentiality mechanisms provided by WS-Security to verify and encrypt different aspects of the message. However, a service may require specific portions of the message to be signed and specific algorithms and keys to be used. For example, a service may require only the body to signed and only SHA for signing and RSA for encryption. This can be provided using the <Integrity> and <Confidentiality> assertions of WS-SecurityPolicy.
- **Message Replay Semantics:** This provides assertions for identifying the freshness of the messages within the system. This is achieved through <wsse:MessageAge> element in WS-SecurityPolicy, which is used to indicate the acceptable time period after which the message can be considered "stale" and discarded. If a policy specifies such an element, then service that is the target of the policy requires the <Timestamp> header of the WS-Security specification in the message, to evaluate and enforce the policy.
- **Security Tokens:** A Web service may require a specific type of security token to be attached to the message. Different tokens may be needed for different purposes. For example, a SAML authorization token may be required for authorization information and a Kerberos ticket for authentication purposes. The <wsse:SecurityToken> element is used to describe what security tokens are required and accepted by the Web service. Also it can specify the service's policy for associating security tokens when sending out messages.

A.2.3 WS-SecureConversation

As we have discussed earlier, WS-Security specifies mechanisms to provide confidentiality/integrity to the messages transmitted to and from a Web service. WS-Policy specifies the policy framework within which the Web Service operates. However, a normal transaction may involve multiple handshakes and multiple messages to be sent. Then it becomes necessary to store the context and send a message identifying the context and possibly encrypting the message based on the keys shared during the context. This is similar to what SSL does in the transport layer. WS-SecureConversation [281] specification has been developed primarily to tackle this problem. It introduces the concept of Security Context Token (SCT), which can be shared among the communicating parties for the lifetime of a communication session. SCT is identified by <SecurityContextToken> in the WS-SecureConversation specification. The elements and attributes of <SecurityContextToken> are:

- **Identifier:** This identifies the security context using an absolute URI.
- **Created/Expires:** This identifies the creation time/expiration time of the security context.
- **Shared Keys:** Holds the shared secrets of the security context.
- **SecurityTokenReference:** This references the shared secrets of the security context.

For a secret communication, a security context needs to be created and shared by the communicating parties before being used. The WS-SecureConversation defines three different ways of establishing a security context among the communicating parties.

- **By a Security Token Service:** The context initiator asks for a security token. For this purpose, the initiating party sends a *RequestSecurityToken* request to the token service and gets the *RequestSecurityTokenResponse* back containing the new security context token.
- **By One of the Communicating Parties:** Here, the initiator sends a security context token and sends it to the other parties. This model works if the sender is always trusted. In this case the sender generates a token and sends an unsolicited *RequestSecurityTokenResponse* to the other parties containing the signed security token.
- **Through Negotiation:** When there is a need to negotiate a security context among the communicating parties, then WS-SecureConversation specification allows the exchange of messages so that a new security context is established through the exchange of data.

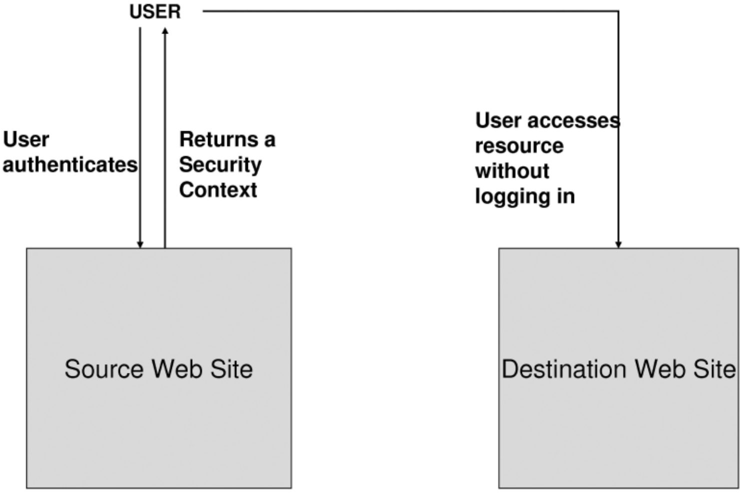

USER
User authenticates
Returns a Security Context
User accesses resource without logging in
Source Web Site
Destination Web Site

Fig. A.3. SAML example scenario

A.2.4 Security Assertions Markup Language (SAML)

In the context of Web services security, it sometimes becomes important to exchange authentication and authorization decisions. Take the example shown in Fig. A.3. There are two Web sites where the user needs to log in. Instead of authenticating twice, the user authenticates only to the source Web site and logs in to the destination Web site using the reference id that the host had provided. Therefore there is a need for a standard way of exchanging authentication information across different systems. Similar to the previous case, a user may be provided with authorization credentials or assertions which it uses to access certain resources. Secure Assertion Markup Language (SAML) [282] provides a standard way of exchanging authorization and authentication information across diverse platforms and systems.

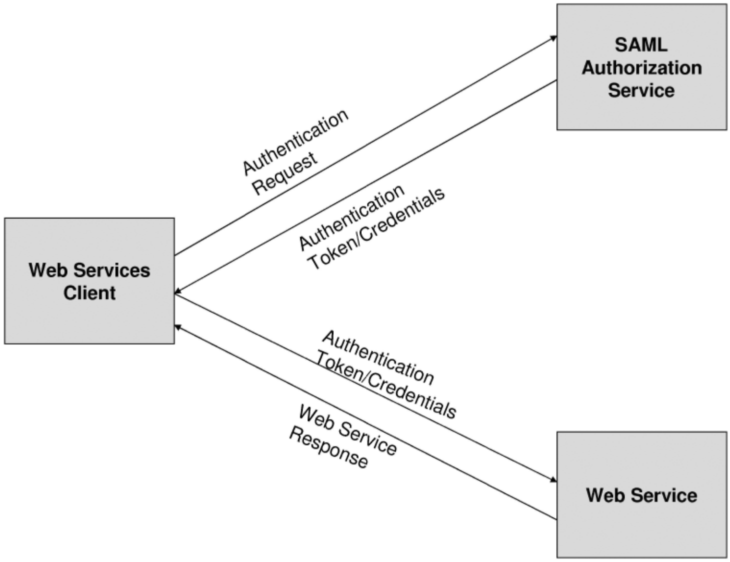

SAML Authorization Service
Authentication Request
Authentication Token/Credentials
Web Services Client
Authentication Token/Credentials
Web Service Response
Web Service

Fig. A.4. A typical Web service authorization dialog

At the heart of SAML lie the SAML assertions. SAML assertions make a statement about the subject which can be an individual or a service. SAML supports three types of assertions: authentication, attribute, and authorization assertions. The assertions can be digitally signed.

- **Authentication Assertions:** This type of assertion asserts that a subject S was authenticated by some means M at time T. The SAML generated by the issuer contains an identification or reference which can be used for single sign on purposes. It also contains some conditions like the assertion cannot be used before a certain time and after a certain time, the method by which the authentication has been carried out, and the information about the subject who is authenticated.
- **Attribute Assertions:** This type of assertion asserts that a subject S is associated with attributes A, B, C, etc**.** with values a, b, c, etc**.** For example, a subject Alice can be associated with attribute “University” and value “Iowa State University.”
- **Authorization Assertions:** This type of assertion asserts that a certain subject S is authorized to perform action A on a resource R

given evidence E. For example, a subject Bob can have action "Read" on resource "File Server X" if he is able to provide an authorization certificate which can be used as evidence.

In Fig. A.4, a typical Web service is shown using SAML. A client asks a SAML authorization service for credentials to access certain Web service or resource. The client returns credentials/tokens which are used by the client to access the Web service which requires the credentials/tokens.

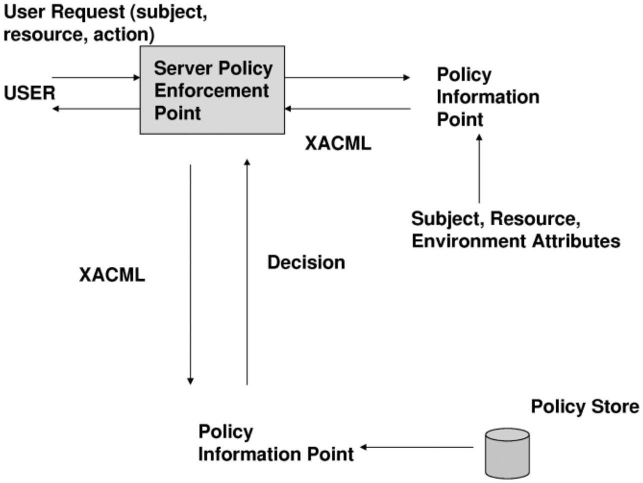

User Request (subject, resource, action)
USER
Server Policy Enforcement Point
Policy Information Point
XACML
Subject, Resource, Environment Attributes
XACML
Decision
Policy Store
Policy Information Point

Fig. A.5. XACML overview

A.2.5 eXtensible Access Control Markup Language

Extensible Access Control Markup Language (XACML) [283] provides a policy language which allows administrators to define the access control requirements for the enterprise resources. The language and schema support include data types, functions, and combining logic which allow complex (or simple) rules to be defined. XACML also includes an access decision language used to represent the runtime request for a resource. When a policy is located which protects a resource, functions compare attributes in the request against attributes contained in the policy rules ultimately yielding a permit or deny decision.

When a client makes a resource request upon a server, the entity charged with access control by enforcing authorization is called the policy enforcement point. In order to enforce policy, this entity will formalize attributes describing the requester at the policy information point and delegate the authorization decision to the policy decision point. Applicable policies are located in a policy store and evaluated at the policy decision point, which then returns the authorization decision. Using this information, the policy enforcement point can deliver the appropriate response to the client. The process is described in Fig. A.5.

A.3 Open Grid Services Architecture (OGSA)

The open standard as has been put forward by the GGF community is called Open Grid Standards Architecture (OGSA). OGSA defines mechanisms based on Web services for different systems to communicate and share the heterogeneous grid resources.

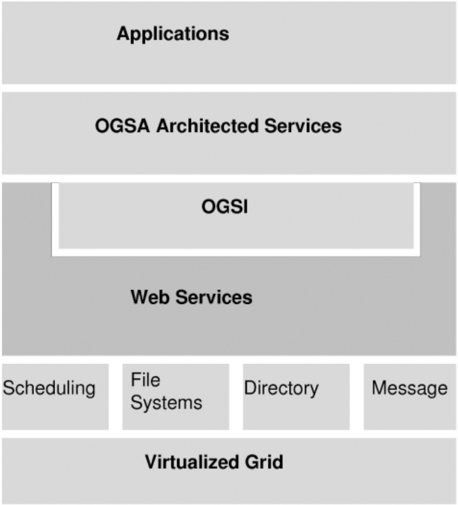


Fig. A.6. Open grid standards architecture

Figure 4.1 shows the high level view of the Open Grid Standards Architecture. The different layers are:

- **Resource Layer:** At the lowest layer we have the resources which comprise the grid infrastructure. The physical resources consist of servers, storage devices, and interconnection networks. Above the physical resources are the logical resources which provide additional functions by virtualizing and aggregating the resources in the physical layer.
- **Web Services:** The second layer in the OGSA architecture is Web services. All grid resources – both logical and physical – are modeled as services. The Open Grid Services Infrastructure (OGSI) specification defines grid services and builds on top of standard Web services technology. OGSI exploits the mechanisms of Web services like XML and WSDL to specify standard interfaces, behaviors, and interaction for all grid resources. OGSI extends the definition of Web services to provide capabilities for dynamic, stateful, and manageable Web services that are required to model the resources of the grid.
- **OGSA Architected Grid Services Layer:** This layer defines the services that are defined on top of the OGSI layer to manage and execute the underlying grid resources. Different services including the security services fall in this category.
- **Grid Applications Layer:** As the grid acquires mainstream stature, more and more grid applications are being written, becoming the highest layer in the OGSA stack.

A.3.1 Open Grid Services Infrastructure (OGSI)

Open Grid Services Infrastructure (OGSI) is primarily concerned with creating, addressing, inspecting, and managing the lifetime of stateful grid services. OGSI specification version 1 [61], defines a grid service as a Web service that conforms to a set of interfaces and behaviors that define how a client interacts with the grid service. These conventions, and other OGSI mechanisms associated with grid service creation and discovery, provide for the controlled, fault resilient, and secure management of the distributed and often long-lived state that is commonly required in advanced distributed applications. OGSI defines a component model by using extended WSDL and XML Schema definition. It introduces the concepts of stateful Web service instances, common metadata and inspection, asynchronous notification of state change, references to instances of services, collections of service instances, and service state data declaration. The last component augments the constraint capabilities of XML schema definition (refer to the appendix for the different components of Web services). More specifically, the OGSI specification defines:

- A set of WSDL extensions some of which have analogous support in WSDL 2.0
- WSDL constructs and standard operations for representing, querying, and updating *service data* (metadata and state data) associated with a service.
- The Grid Service Handle and Grid Service Reference constructs, used to address grid services.
- A definition of common fault information from operations that defines a base XML Schema and associated semantics for WSDL fault messages to support a common interpretation. The approach simply defines the base format for fault messages, without modifying the WSDL fault message model.
- A set of operations for creating and destroying grid services that provides for both explicit destruction of services and implicit garbage collection of expired services without the need for explicit destruction.
- A set of operations for creating and using heterogeneous by-reference collections of Web services.
- Mechanisms for requesting asynchronous notifications of changes in the value of service data elements.

A.3.2 Web Services Critique of OGSI

Once the OGSI framework was proposed, the Web services community started voicing their disapproval mainly because of the stateful nature of the grid services. Following are a few criticisms against the OGSI specification which resulted in the grid and Web services community working together and converging on the WS Resource Framework [62].

1. **Too Much Stuff in One Specification:** OGSI did not have a clean separation or factoring of functions to support incremental adoption. For example, event notification could have been made independent of service data as it was useful even without it. Metadata introspection is

a useful concept that does not require expression through service data. Therefore, the criticisms mandated more modular specification which was provided in WSRF.

2. **Does not Work Well with Existing Web Services and XML Tooling:** OGSI v1.0 uses XML schema aggressively, for example with substantial use of generic (xsd:any) attributes and "document-oriented" WSDL operations. These features cause problems with, for example, JAX-RPC. These considerations resulted in the use of standard XML schema mechanisms that are familiar to developers and are supported by existing tooling in the WS-Resource Framework.
3. **Too Object Oriented:** OGSI v1.0 models a stateful resource as a Web service that encapsulates the resource's state, with the identity and lifecycle of the service and resource state coupled. Since Web services supposedly do not have states or instances, this approach led to a lot of criticism from the Web services purists. In addition, some Web service implementations do not accommodate dynamic service creation and destruction. These considerations were used when the design of WS-Resource was drawn for the WS-Resource Framework.

A.3.2 Web Services Resource Framework (WSRF)

Based on the criticisms mentioned in the previous section, the Web services and the grid community had come together to define a set of specifications, collectively known as the *WS-Resource Framework.* The WS-Resource Framework is primarily concerned with the creation, addressing, inspection, and lifetime management of the stateful resources. There is also a WS-Notification family of specifications [63,64] which addresses notification, subscription, and delivery functionalities. Following are some of the high-level specifications which define WS-Resource Framework.

- **WS-ResourceProperties:** These specifications define how to associate the stateful resources and Web services to produce WS-Resources. The specifications also define mechanisms to retrieve, change, and delete the properties of a WS-Resource.
- **WS-ResourceLifetime:** These sets of specifications define mechanisms to manage the lifetime of WS-Resource by providing means to destroy immediately or in some scheduled time in the future.

- **WS-RenewableReferences:** Sometimes an endpoint reference may become invalid for many reasons. These specifications define mechanisms to retrieve a new endpoint when the old one becomes invalid.
- **WS-ServiceGroup:** These specifications allow the creation and use of heterogeneous by-reference collection of Web Services.
- **WS-BaseFault:** These are used for reporting errors by defining base faults.
- **WS-Notification:** These set of specifications do not fall under the WS-Resource Framework. They define notification mechanisms through publish and subscribe patters, and can be used by the WS-Resource Framework.

Bibliography

1. Spiderman movie. *http://spiderman.sonypictures.com,* accessed on 13th July, 2006
2. Shrek movie. *http://www.shrek.com,* accessed on 13th July, 2006
3. National Weather Service (NWS). *http://www.nws.noaa.gov/,* accessed on 13th July, 2006
4. European Organization for Nuclear Research (CERN). *www.cern.org*, accessed on 13th July 2006
5. SETI@home. *http://setiathome.ssl.berkeley.edu/,* accessed on 13th July 2006
6. A. Ghosh, A. Chakrabarti, R.A. Dheepak, S. Ali S, I. Gupta. Streamlining Drug Discovery Research by Leveraging Grid Workflow Manager. In *Life Sciences Grid (LSGrid),* Singapore, 2005.
7. L. Dimitriou. Financial services Grid virtualization for increased business performance and lower TCO. In *Grid in Finance,* NY, 2006.
8. R. Reuter R, R. Hoffmann, J. Kamarajan. Application of stochastic Simulation in Automotive Industry. In *Proc. AMERI-PAM*, Detroit, 2000.
9. G. Koch G. Discovering Multi-Core: Extending the Benefits of Moore's Law. In *Technology Intel® Magazine*, 2005.
10. C. Vilett C. Moore's Law vs. storage improvements vs. optical improvements. In *Scientific American*, 2001.
11. R. Buyya (Ed). *High Performance Cluster Computing: Architectures and Systems*. Kluwer Academic Publishers, 2003.
12. InfiniBand Trade Association. InfiniBand Architecture`Specification, Volume 1, Release 1.1, 2002.
13. I. Foster, C. Kasselman. *The Grid 2: Blueprint for a new Computing Infrastructure*. Morgan Kaufman Publishers, 2004.
14. I. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001.
15. Oracle® Corporation. Oracle® Grid Computing. *White Paper*, 2002, available at http://www.oracle.com/technologies/grid/OracleGridBWP0105.pdf, accessed on 13th July, 2006.
16. Sun® Microsystems. Employing Grid Computing for Competitive Advantage. *Executive Brief*, 2003.
17. G. Lee. HP and the Grid. In *Intl. Symp. on Grid Comp. (ISGC)*, Taiwan, 2003.
18. IBM® Grid Computing Group. IBM® and Grid. *White Paper,* 2003.
19. W. Gropp, E. Lusk, A. Skjellum. *Using MPI.* 2nd Edition, MIT Press, 1999.

20. A. Geist, A. Beguelin, A. Dongarra, W. Jiang, R. Manchek, V. Sunderam. *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing*. MIT Press, 1994.
21. Gnutella. *www.gnutella.com*, accessed on 13th July, 2006.
22. I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan H. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM*, San Diego, 2001.
23. S. Ratnasamy. A Scalable Content Addressable Network. *PhD Thesis*, University of California, Berkeley, 2002.
24. D. Heap. Scorpion: Simplifying the Corporate IT Infrastructure. *IBM® Research White Paper*, 2000.
25. S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol*, pp. 403-410, 1990.
26. Data Synapse®.˙ *www.datasynapse.com*, accessed on 13th July, 2006.
27. United Devices®.˙ *www.ud.com*, accessed on 13th July, 2006.
28. T.G. Lewis, H. El-Rewini. Parallax: A Tool for Parallel Program Scheduling. *IEEE J. Parallel and Distributed Technology*, Vol.1, No. 3, pp. 62–72, 1993.
29. T. Yang, A. Gerasoulis. PYRROS: Static Task Scheduling and Code Generation for Message-Passing Multiprocessors. In *Proc. 6th ACM Int'l Conf. Supercomputing*, ACM Press, New York, pp. 428–433, 1992.
30. P-Grade Team. Parallel Grid Runtime and Application Development Environment. *User's Manual version. 8.4.2*, 2006..
31. I. Ahmed, Y-K. Kwok, M-Y. Wu, W. Shu W. CASCH: A Tool for Computer-Aided Scheduling. *IEEE Concurrency*, vol. 8, no. 4, pp. 21-33, 2000.
32. Aspeed®.*www.aspeed.com*, accessed on 21st July, 2006.
33. Cornell Theory Center. *www.tc.cornell.edu*, accessed on 21st July, 2006.
34. K. Gor, R.A. Dheepak, S. Ali, L.D. Alves, N. Arurkar, I. Gupta, A. Chakrabarti, A. Sharma, S. Sengupta. Scalable Enterprise Level Workflow and Infrastructure Management in a Grid Computing Environment. In *Proc. CCGrid,* Cardiff (Wales), pp. 661-667, 2005.
35. S. Pfleeger, C.P. Pfleeger. *Security in Computing*. 3rd Edition, Prentice Hall Publishers, 2002.
36. US Dept. of Commerce/ NIST. Data Encryption Standard. *Federal Information Processing Standard (FIPS) Publication*, 46-2, 1993.
37. NIST. Advanced Encryption Standard. *Federal Information Processing Standard (FIPS) Publication 197,* 2001.
38. R. Rivest, A. Shamir, L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. *Communications of the ACM,* vol. 21, no. 2, 120-126, 1978.
39. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, v. IT-31, no. 4, pp. 469–472, 1985.

40. H.M. Heys. A Tutorial on Linear and Differential Cryptanalysis. *Technical Report CORR 2001-17*, Centre for Applied Cryptographic Research, Department of Combinatorics and Optimization, University of Waterloo, 2001.
41. W. Diffie, M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.
42. W.R. Cheswick, S.M. Bellovin, A.D. Ruben. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addision-Wesley Publishers, 2003.
43. R. Rivest. MD5 Message Digest Algorithm. *IETF RFC 1321*, 1992.
44. D. Eastlake, P. Jones. US Secure Hash Algorithm 1 (SHA1). *RFC 3174*, 2001.
45. M. Bellare, R. Canetti, H. Crawczyk. Keying Hash Functions for Message Authentication. In *Proc. Advances in Cryptography (Crypto),* Springer-Verlag, 1996.
46. R. Housley, W. Ford, W. Polk, D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *IETF RFC 2459*, 1999.
47. C. Kaufman, R. Perlman, M. Speciner. *Network Security: Private Communication in a Public World*. Prentice Hall Publishers, 1995.
48. T. Dierks, C. Allen. The TLS Protocol Version 1.0. *IETF RFC 2246*, 1999.
49. Netscape® Corporation. *www.netscape.com*, accessed on 13th July, 2006.
50. Internet Engineering Task Force (IETF). *www.ietf.org*, accessed on 13th July 2006.
51. J.T. Kohl, B.C. Neuman, T.Y. T'so. The Evolution of the Kerberos Authentication System. *Distributed Open Systems*, IEEE Computer Society Press, pp. 78-94, 1994.
52. R. Needham, M. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM,* vol. 21, pp. 393-399, 1978.
53. S. Kent, R. Atkinson. Security Architecture for Internet Protocol. *IETF RFC. 2401*, 1998.
54. W. Stallings. IP Security. *Internet Protocol Journal*, vol. 7, no. 1, 2000.
55. S. Kent, R. Atkinson. IP Authentication Header. *IETF RFC 2402*, 1998.
56. S. Kent, R. Atkinson. IP Encapsulated Security Payload (ESP). *IETF RFC 2406,* 1998.
57. G. McDaniel. *IBM® Dictionary of Computing*, McGraw-Hill, 1994.
58. Global Grid Forum (GGF). *www.ggf.org*, accessed on 13th July 2006.
59. I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Tech Report,* Argonne National Lab, 2002.
60. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman. Grid Service Specification. *Tech Report*, Argonne National Lab, 2002.
61. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt. Open Grid Services Infrastructure: v. 1.0. *GGF Draft*, 2003.
62. I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, T. Storey, W. Vambenepe, S. Weerawarana. Modeling Stateful Resources with Web Services. *IBM®*

Developer Works, 2004, available at http://www.128.ibm.com/developer works/library/ws-resource/ws-modelingresources.pdf, accessed on 13th July, 2006.

63. S. Graham, D. Hull, B. Murray. Web Services Base Notification 1.3 (WS-BaseNotification,). *OASIS Draft,* 2006, available at http://www.oasis-open.org/committees/download.php/13488/wsn-ws-base_notification, accessed on July 13th, 2006.
64. D. Chappell, L. Liu. Web Services Brokered Notification 1.3 (WS-BrokeredNotification). *OASIS Draft,* 2006, available at http://www.oasis-open.org/committees/download.php/13485/wsn-ws-brokered_notification.pdf, Accessed on 13th July, 2006.
65. N. Nagarathnam, P. Janson, J. Dayka, A. Nadalin, F. Siebenlist, V. Welch, I. Foster, S. Tuecke. The Security architecture for Open Grid Servies. *GGF Draft,* 2002, available at http://www.cs.virginia.edu/~humphrey/ogsa-sec-wg/OGSA-SecArch-v1-07192002.pdf, accessed on 13th July 2006.
66. www.globus.org, accessed on 13th July 2006.
67. B. Sundaram. Introducing GT4 Security. *IBM® Developer Works,* 2005, available at http://www.ibm.com/developerworks/grid/library/gr-gsi4intro, accessed on 13th July, 2006.
68. Documentation Team. *GT4 toolkit,* 2006, available at, http://www-unix.globus.org/toolkit/docs/4.0/admin/docbook, accessed on 13th July 2006.
69. M. Abdalla, O. Chevassut, D. Pointcheval. One-time Verifier-based Encrypted Key Exchange. In *Proc. International Workshop on Practice and Theory in Public Key Cryptography (PKC)*, Switzerland, pp 47-64, 2005.
70. O. Kornievskaia, P. Honeyman, B. Doster, K. Coffman. Kerberized Credential Translation: A Solution to Web Access Control. In *Proc. USENIX Security Symposium*, Washington, pp. 235-249, 2001.
71. S. Shirasuna, H. Nakada, S. Matsuoka, S. Sekiguchi. Evaluating Web Services based Implementation of GridRPC. In *Proc. IEEE High Perf. Dist. Computing (HPDC),* pp. 237-245, 2002.
72. D. Bell. Secure computer systems: A network interpretation. In *Proceedings on 3rd Annual Computer Security Application Conference*, pp. 32–39, 1987.
73. T. Lee. Using mandatory integrity to enforce “commercial” security. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland (CA), pp. 140–146, 1988.
74. B. Lampson. Protection. In *Proceedings of the 5th Symposium on Information Sciences and Systems,* Princeton (NJ), pp. 437–443, 1974.
75. R. Sandhu, E. Coyne, H. Feinstein, C. Youman. Role-Based Access Control Models. *IEEE Computer*, vol. 29, no. 2, pp. 38-47, 1996.
76. D.E. Denning. A Lattice Model of Secure Information Flow. *Communications of the ACM,* 19 (2), pp. 236–243, 1976.
77. R. Sandhu. Role Hierarchies and Constraints for Lattice-based Access Controls. In *Proceedings of the Conference on Computer Security,* New York, pp. 65–79, 1996.

78. S. Osborn, R. Sandhu, Q. Munawer. Configuring Role -Based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Trans. on Information and System Security*, vol. 3, no. 2, pp. 85-106, 2000.
79. L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke. A Community Authorization Service for Group Collaboration. In *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, Monterey (CA), pp 50-59, 2002.
80. R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, A. Frohner, A. Gianoli, K. Lorentey, F. Spataro. VOMS: an Authorization System for Virtual Organizations. In *1st European Across Grids Conference*, Santiago e Compostella (Spain), 2003.
81. A. Chakrabarti, A. Damodaran. Enterprise Authorization and Licensing Service. *Infosys Tech. Report,* 2006.
82. M. Thompson, A. Essiari, S. Mudumbai. *ACM Transactions on Information and System Security, (TISSEC),* vol. 6, issue 4, pp: 566-588, 2003.
83. D. Chadwick, O. Otenko. The PERMIS X.509 Role Based Privilege Management Infrastructure. In *ACM SACMAT*, Lake Tahoe (CA), pp. 135-140, 2002.
84. www.earthsystemgrid.org, accessed on 13th July, 2006.
85. M.B. Juric, B. Mathew, P. Sarang. *Business Process Execution Language for Web Services: BPEL and BPEL4WS*, PACKT Publishing, ISBN 1904811817, 2006.
86. D. Chadwick, O. Otenko, V. Welch. Using SAML to Link the Globus Toolkit to the PERMIS Authorization Infrastructure. In *Proc. IFIP TC-6 TC-11 Conf. on Comm. and Multimedia Security*, Windermere (UK), 2004.
87. Altair® Engineering. www.altair.com, accessed on 13th July 2006.
88. Platform® Computing. www.platform.com, accessed on 13th July, 2006.
89. K.J. Houle, G.M. Weaver. Trends in Denial of Service Attack Technology. *CERT Advisory,* v1.0, 2001.
90. CSI/FBI. *Computer Crime and Security Survey,* 2001, available at http://www.crime-research.org/news/11.06.2004/423, accessed on July 13th, 2006.
91. D. Moore, G.M. Voelker, S. Savage. Inferring Internet denial-of-service activity. In *Proceedings of the 2001 USENIX Security Symposium*, Washington DC, 2001.
92. P.J. Criscuolo. Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht *CIAC-2319*. Department of Energy Computer Incident Advisory Capability (CIAC), UCRL-ID-136939, Rev. 1., Lawrence Livermore National Laboratory, 2000.
93. Web Report. Yahoo on Trail of Site Hackers. *Wired.com,* 2000, available on http://www.wired.com/news/business/0,1367,34221,00.html, accessed on 13th July, 2006.
94. Web Report. Powerful Attack Cripples Internet. *Associated Press for Fox News,* 2002, available at http://www.linuxsecurity.com/content/view/112716/2/, accessed on July 13th 2006.

95. Cisco® White Papers. *Strategies to Protect against Distributed Denial of Service Attacks (DDoS),* 2001.
96. K. Park, H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internet. In *Proc. ACM SIGCOMM,* San Diego (CA), pp. 15-26, 2001.
97. A.D. Keromytis, V. Misra, D. Rubenstien. SOS: Secure Overlay Services. In *Proc. ACM SIGCOMM*, Pittsburgh (PA), pp. 61-72, 2002.
98. D.K. Yau, J.C.S. Lui, F. Liang. Defending Against Distributed Denial of Service Attacks with Max-min Fair Server-centric Router Throttles Quality of Service. *Tenth IEEE International Workshop on QoS (IWQoS),* Miami (FL), pp. 35-44, 2002.
99. J. Allen. State of The Practice: Intrusion Detection Technologies. *Carnegie Mellon, SEI, Tech,* Report CMU/SEI-99-TR-028, ESC-99-028, 2000.
100. S. Axelsson. Intrusion Detection Systems: A Survey and Taxonomy. *Technical report 99-15,* Dept. of Computer Engineering, Chalmers University of Technology, Goteborg (Sweden), 2000.
101. Snort. *http://www.snort.org*, accessed on 13th July, 2006.
102. S. Kenny, B. Coghlan. Towards a Grid wide Intrusion Detection System. In *Proc. European Grid Conference (EGC)*, Prague, 2005.
103. M.F. Tolba, M.S. Abdel-Wahab, I.A. Taha, A.M. Al-Shishtawy. GIDA: Toward Enabling Grid Intrusion Detection System. In *Proc. Conference on Cluster Computing and Grid (CCGrid)*, Cardiff (Wales), 2005.
104. www.oracle.com/technology/products/bi/odm/pdf/odm_based_intrusion_detection_paper_1205.pdf, accessed on 13th July, 2006.
105. T. Ryutov, C. Neumann, L. Zhou. Integrated Access Control and Intrusion Detection (IACID) Framework for Secure Grid Computing. *Tech. Report.*, University of Southern California, 2005.
106. H. Burch, B. Cheswick. Tracing anonymous packets to their approximate sources. In *Proc. USENIX LISA Conf.*, New Orleans (LA), pp. 319-327, 2000.
107. G. Sager. Security Fun with OCxmon and eflow. *Internet2 Working Group Meeting,* 1998.
108. A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, T. Kent, T. Strayer. Hash-Based IP Traceback. In *Proc. ACM SIGCOMM,* San Diego (CA), pp. 3-14, 2001.
109. S.M. Bellovin. ICMP Traceback Messages. *Internet Draft,* 2000. draft-bellovin-itrace-00.txt.
110. S. Savage, D. Wetherall, A. Karlin, T. Anderson. Network Support for IP Traceback. *IEEE Trans. on Networking*, vol. 1, no. 3, pp. 226-237, 2001.
111. D.X. Song, A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *Proc. INFOCOM,* Alaska, pp. 878-886, 2001.
112. A. Habib, S. Fahmy, S.R. Avasarala, V. Prabhakar, B. Bhargava. On Detecting Service Violations and Bandwidth Theft in QoS Network Domains. *Computer Communications*, Elsevier, vol. 26, issue 8, pp. 861-871, 2003.
113. Y. Breitbart et al. Efficiently monitoring bandwidth and latency in IP networks. In *Proc. IEEE INFOCOM*, Alaska, pp. 933-942, 2001.

114. M.C. Chan, Y-J. Lin, X. Wang. A scalable monitoring approach for service level agreements validation. In *Proc. of the International Conference on Network Protocols (ICNP)*, Osaka (Japan), pp. 37–48, 2000.
115. M. Dilman, D. Raz. Efficient reactive monitoring. In *Proc. IEEE INFOCOM,* Alaska, 2001.
116. N.G. Duffield, F.L. Presti, V. Paxson, D. Towsley. Inferring link loss using striped unicast probes. In *Proc. IEEE INFOCOM*, Alaska, pp. 915-923, 2001.
117. K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, R.A. Olsson. Detecting Disruptive Routers: A Distributed Network Monitoring Approach. In *Symp. on Security and Privacy,* Oakland (CA), pp. 115-124, 1998.
118. A. Barmouta A, R. Buyya. GridBank: A Grid Accounting Services Architecture (GASA) for Distributed Systems Sharing and Integration. In *International Parallel and Distributed Processing Symposium (IPDPS'03),* Nice (France), 2003.
119. T. Sandholm. Service Level Agreement Requirements of an Accounting-Driven Computational Grid. *Royal Institute of Technology, Technical Report TRITA-NA-05332005*, 2005.
120. B. Bershad, S. Savage, P. Pardyak, E.G. Sirer, D. Becker, M. Fiuczynski, C. Chambers, S. Eggers. Extensibility, Safety, and Performance, in the SPIN Operating System. In *ACM Symp. On Operating Systems Principles (SOSP),* Copper Mountain (CO), pp. 267-283, 1995.
121. G. Necula. Proof Carrying Code. *Principles of Programming Languages*, Paris (France), 1997.
122. VMWare®. *www.vmware.com*, accessed on 13th July, 2006.
123. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield. Xen and the Art of Virtualization. In *ACM Proc. Symp. On Operating Systems Principles (SOSP)*, NY, pp. 164-177, 2003.
124. Virtuozzo Team. A Complete Server Virtualization and Automation Solution. *Virtuozzo White Paper and Data Sheet*, 2005.
125. VServer. *http://linux-vserver.org/Documentation*, accessed on 13th July, 2006.
126. R. Uhlig, G. Neiger, D. Rodgers, A.L. Santoni, F.C.M. Martins, A.V. Anderson, S.M. Bennett, A. Kagi, SF.H. Leung, L. Smith. Intel® Virtualization Technology. *IEEE Computer*, vol. 38, no. 5, pp. 48-56, 2005.
127. AMD® Corporation. www.amd.com, accessed on 13th July 2006.
128. J.S. Robin, C.E. Ervine. Analysis of the Intel® Pentium's Ability to Support a Secure Virtual Machine Monitor. In *Proc. 9th USENIX Security Symposium*, Denver (CO), 2000.
129. S. Sundarrajan, H. Nellitheertha, S. Bhattacharya, N. Arurkar. Nova: An Approach to On-Demand Virtual Execution Environments for Grids. In *Proc. CCGrid,* Singapore, pp. 544-547, 2006.
130. S. Soltesz, H. Potzl, M.E. Fiuczynski, A. Bavier, L. Peterson. Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. *Princeton White Paper,* 2005.

131. B.W. Lampson. On reliable and extendable operating systems. *State of the Art Report,* Infotech, 1, 1971.
132. B.W. Lampson, R.F. Sproull. An Open Operating System for a Single-User Machine. In *Proc. Seventh ACM Symposium on Operating Systems Principles*, Pacific Grove (CA), pp. 98–105, 1979.
133. W. Wulf, E. Cohen, W. Corwin, A. Jones, R. Levin, C. Pierson, F. Pollack. HYDRA: The Kernel of a Multiprocessor Operating System. *Communications of the ACM,* vol. 17, no. 6, , pp. 337-344, 1974.
134. D.R. Engler, M.F. Kaashoek, J. O'Toole Jr. Exokernel: An operating system architecture for application-level resource management**.** In *ACM Symp. On Operating Systems Principles (SOSP),* Copper Mountain (CO), pp. 251-266, 1995.
135. W. Wulf, C.G. Bell. C.mmp – a multi-mini-processor. In *Proc. AFIPS, FJCJ,* Vol. 41, AFIPS Press, pp. 765-777, 1972.
136. I. Goldberg, D. Wagner, R. Thomas, E.A. Brewer. A Secure Environment for Untrusted Helper Applications: Confining the Wiley Hacker. In *USENIX Security Symposium*, San Jose (CA), 1996.
137. M. Bernaschi, E. Gabrielli, L.V. Mancini. REMUS: a Security-Enhanced Operating System. *ACM Transactions on Information and System Security*, vol. 5, no. 1, pp. 36-6, 2002.
138. B. Calder, A. Chien, J. Wang, D. Yang. The Entropia Virtual Machine for Desktop Grids. In *Intl. Conf. on Virtual Execution Env*, Chicago (IL), 2005.
139. P. Brucker. *Scheduling Algorithms*, 4th Edition, Springer Publishers, ISBN 3-540-20524-1, 2004.
140. C.S.R. Murthy, G. Manimaran. *Resource Management in Real Time Systems and Networks,* MIT Press, ISBN 0262133768, 2001.
141. GGF Document. Advanced Reservation: State of the Art. *GGF Draft*, ggf-draft-sched-graap-2.0, 2003.
142. Sun® One Grid Engine, *http://wwws.sun.com/software/gridware*, accessed on 13th July, 2006.
143. D. Abramson, R. Buyya, J. Giddy J. A Computational Economy for Grid Computing and its Implementation in the Nimrod-G Resource Broker. *Future Generation Computer Systems,* vol. 18, issue 8, pp. 1061-1074, 2002.
144. A. Chakrabarti, G. Manimaran. Internet Infrastructure Security: A Taxonomy. *IEEE Networks*, vol. 16, no. 6, pp. 13-21, 2002.
145. K. Ingham, S. Forrest. History and Survey of Network Firewalls. *Tech. Report. of University of New Mexico,* TR-2002-37, 2002.
146. Cisco® Systems. Evolution of Firewall Industry. *White Paper*, Cisco® Systems, 2002.
147. T.D. Yao. Adaptive Firewalls for the Grid. *Master's Thesis*, Technical University of Denmark, 2005.
148. D. Davis. Firewalls: Web Services Achilles Heel? In *IBM® Developer Works*, 2005, available at http://www.128.ibm.com/developerworks/library/ws-pollingpaper.html, accessed on 13th July, 2006.

149. Reactivity XML Firewall. *http://www.react ivity.com/products/solution.html*, accessed on 13th July 2006.

150. Quadrasis EASI Soap Content Inspector. http: / /www.quadrasis.com/ solutions/ products/ easi_product_packages/ easi_soap.htm, accessed on 13th July, 2006.

151. M. Cremonini, E. Damiani, S. De Capitani di Vimercati, P. Samarati P. An XML-based Approach to Combine Firewalls and Web Services Security Specifications. *ACM Workshop on XML Security,* Fairfax (Virginia), pp. 69-78, 2003.

152. D. Caromel, A. Costanzo, D. Gannon, A. Slominski. Asynchronous Peer-to-Peer Web Services and Firewalls. In *Proc. IPDPS*, Washington DC, 2005.

153. W. Augustyn, Y. Serbest. Service requirements for layer 2 provider provisioned virtual private networks. *Internet Draft,* 2003, draft-augustyn-ppvpn-l2vpn-requirements-02.txt.

154. E. Rosen, Y. Rekhter. BGP/MPLS VPNs. *IETF RFC 2547,* 1999.

155. H. OuldBrahim, G. Wright, B. Gleeson, T. Sloane, R.B.C. Sargor, I. Nehusse, J. Yu, R. Bach, A. Young, L. Fang, C. Weber. Network based IP VPN architecture using virtual routers. *Internet Draft,* 2003, draft-ietf-ppvpn-vpn-vr-03.txt.

156. T. Bates, Y. Rehkter, R. Chandra, D. Katz. Multiprotocol extensions for BGP-4. *IETF RFC 2858,* 2000.

157. N.G. Duffield, P. Goyal. Greenberg, P. Mishra, K.K. Ramakrishnan, J.E. van der Merive. A flexible model for resource management in virtual private networks. In *Proc. of the Conference on Applications, Technologies, Architectures, and Protocols. Computer Communication*, ACM Press, pp. 95–108, 1999.

158. S. Andreozzi, T. Ferrari, E. Ronchieri. On-Demand VPN Support for Grid Applications. In *Conf. on High Energy Physics (CHEP)*, Interlaken (Switzerland), 2004.

159. G. Malkin. RIP Version 2. *IETF RFC 2453,* 1998.

160. J. Moy. OSPF Version 2. *IETF RFC 1583,* 1995.

161. Y. Rekhter, T. Li. A Border Gateway Protocol 4. *IETF RFC 1771*, 1995.

162. K. Zhang. Efficient Protocols for Signing Routing Messages. In *Proc. Symp. on Network and Distributed System Security (NDSS),* San Diego (CA), 1998.

163. S. Kent, C. Lynn, K. Seo K. Secure Border Gateway Protocol (S-BGP). *IEEE J of Selected Areas of Comm. (JSAC),* vol. 18, no. 4, pp. 582-592, 2000.

164. F. Wang, F. Gong, F.S. Wu, R. Narayan. Intrusion Detection for Link State Routing Protocol Through Integrated Network Management. In *Proc. ICCCN*, Boston (MA), pp. 694-699, 1999.

165. A. Chakrabarti, G. Manimaran. Secure Link State Routing Protocol. *Technical Report,* Dept. ECpE, Iowa State University, 2002.

166. B.R. Smith, S. Murthy, J.J. Garcia-Luna-Aceves. Securing Distance-Vector Routing Protocols. In *Proc. NDSS,* San Diego (CA), pp. 85-92, 1997.

167. A. Chakrabarti, G. Manimaran. An Efficient Algorithm for Routing Update Detection & Recovery in Distance Vector Protocols. In *Proc. ICC,* Alaska, 2003.
168. C. Diot, W. Dabbous, J. Crowcroft. Multipoint communications: A survey of protocols, functions and mechanisms. *IEEE J. Select. Areas Communications (JSAC)*, vol. 15, no. 3, pp. 277-290, 1997.
169. J. Hou, B. Wang. Multicast routing and its QoS extension: Problems, algorithms and Protocols. *IEEE Networks,* pp. 22-36, 2000.
170. L. Sahasrabuddhe, B. Mukherjee. Multicast routing algorithms and protocols: A tutorial. *IEEE Network,* pp. 90-102, 2000.
171. J.C. Pasquale, G.C. Polyzos, G. Xylomenos. The multimedia multicasting problem. *Multimedia Systems*, vol.6, no.1, pp.43-59, 1998.
172. M. Ramalho. Intra- and Inter- domain multicast routing protocols: A survey and taxonomy. *IEEE Communications Surveys and Tutorials*, vol. 3, no. 1, pp. 2-25, 2000.
173. AccessGrid. *www.accessgrid.org*, accessed on July 13th, 2006.
174. S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *Proc. ACM SIGCOMM,* Cannes (France), pp. 277-288, 1997.
175. T. Hardjono, B. Cain. Key establishment for IGMP authentication in IP multicast. In *Proc. IEEE ECUMN*, Colmar (France), 2000.
176. A. Ballardie. Scalable Multicast Key Distribution. *RFC 1949,* 1996.
177. C. Shields, J.J. Garcia-Luna-Aceves. KHIP - A Scalable Protocol for Secure Multicast Routing. In *Proc. ACM SIGCOMM*, Cambridge (MA), pp. 53-64, 1999.
178. D.M. Wallner, E.J. Harder, R.C. Agee. Key Management for Multicast: Issues and Architectures. *IETF RFC 2627*, 1999.
179. C. Wong, M. Gouda, S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM Trans. On Networking (ToN)*, vol.8, issue 1, pp. 16-30, 1998.
180. Y.R. Yand, X.S. Li, X.B. Zhang, S.S. Lam. Reliable Group Rekey: A Performance Analysis. In *Proc. ACM SIGCOMM*, San Diego (CA), pp. 27-38, 2001.
181. R. Gennaro, P. Rahtogi. How to sign digital streams. In *Proc. CRYPTO*, Santa Barbara (CA), Springer-Verlag, pp. 180-197, 1997.
182. C.K. Wong. Digital Signatures for Flows and Multicasts. *IEEE/ACM Trans. on Networking*, vol. 7, no. 4, pp. 502-513, 1999.
183. M. Gaynor, S. Moulton, M. Welsh, E. LaCombe, A. Rowan, J. Wynne. Integrating Wireless Sensor Networks into the Grid. *IEEE Internet Computing*, vol. 8, no. 4, pp.32-39, 2004.
184. P. Pietzuch, J. Shneidman, J. Ledlie, M. Welsh, M. Seltzer, M. Roussopoulos. Hourglass: A Stream-Based Overlay Network for Sensor Applications. *Harvard Industrial Partnership (HIP'04)*, 2004.
185. Y.C. Hu, A. Perrig, D.B. Johnson. Wormhole detection in wireless ad hoc networks. *Tech. Rep. TR01-384*, Department of Computer Science, Rice University, 2002.

186. A. Perrig, P. Szewczyk, J.D. Tygar, V. Wen, D. Culler D. SPINS: Security Protocols for Sensor Networks. *Wireless Networks*, vol. 8, no. 5, pp. 521-534, 2002.
187. C. Karlof, N. Sastry, D. Wagner. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proc. ACM SenSys,* Baltimore (MD), 2004.
188. P. Balaji, W. Feng, Q. Gao, R. Noronha, W. Yu, D.K. Panda. Head-to-Toe Evaluation of High Performance Sockets over Protocol Offload Engines. In *Proc. Cluster Computing and Grid (CCGrid)*, Cardiff (Wales), 2005.
189. R. Dimitrov, M. Gleeson. Challenges and New Technologies for Addressing Security in High Performance Distributed Environments. In *Proc. of the 21st National Information Systems Security Conference*, Arlington (Virginia), pp. 457-468, 1998.
190. M. Lee, E.J. Kim, M. Yousif. Security Enhancements in Infiniband Architecture. In *Proc. IPDPS*, Denver (CO), 2005.
191. H.J. Chao, R. Karri, W.C. Lau. CYSEP – A Cyber Security Processor for 10 Gbps Networks and Beyond. In *IEEE MILCOM*, Monterey (CA), pp. 1114-1122, 2004.
192. J. Basney, W. Yurcik, R. Bonilla, A. Slagell. Credential Wallets: A Classification of Credential Repositories, Highlighting MyProxy. In *31st Annual TPRC, Research Conference on Communication, Information, and Internet Policy*, Arlington (Virginia), 2006.
193. A. Arsenault, A. Diversinet, S. Farrel. Securely Available Credentials – Requirements. *IETF RFC 3157*, 2001.
194. D. Gustafson, M. Just, M. Nystrom. Securely Available Credentials (SACRED) - Credential Server Framework. *IETF RFC 3760*, 2004.
195. S. Petri. An Introduction to Smart Cards. *Messaging Magazine*, Sep. Issue, 1999.
196. B. Schneier, A. Shostack. Breaking Up is Hard to Do: Modeling Security Threats for Smart Cards. In *USENIX Workshop on Smart Card Technology*, Chicago (IL), pp. 175-185, 1999.
197. R. Sandhu, M. Bellare, R. Ganesan. Password-Enabled PKI: Virtual Smartcards versus Virtual Soft Tokens. In *1st Annual PKI Workshop*, pp. 89-96, 2002.
198. J. Basney, M. Humphrey, V. Welch. The MyProxy Online Credential Repository. *IEEE Software Practice and Experience*, vol. 35, issue 9, pp. 801-816, 2005.
199. M. Lorch, J. Basney, D. Kafura D. A Hardware-secured Credential Repository for Grid PKIs. In *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, Chicago (IL), pp. 640-647, 2004.
200. K. Mosebach, L.D. Alves, A. Chakrabarti. Virtualized Credential Management in Inter-domain Grid System. *Trusted Internet Workshop (TIW)*, 2004.
201. Liberty Alliance. Introduction to Liberty Alliance Identity Architecture. *Liberty Alliance White Paper and Documentation*, available at Liberty Web site http://www.projectliberty.org/resources/whitepapers/LAP%20Identity%20Architecture%20Whitepaper%20Final.pdf, accessed on 13th July, 2006.

202. Shibboleth Internet2 project. *http://shibboleth.internet2.edu*, accessed on 13th July, 2006.

203. V. Welch, T. Barton, K. Keahey, F. Siebenlist. Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration. *4th Annual PKI R&D Workshop*, 2004.

204. M. Deutsch. Cooperation and Trust: Some Theoretical Notes. *Nebraska Symposium on Motivation*, M. R. Jones, Nebraska University Press, pp. 279-319, 1962.

205. M. Deutsch. The Resolution of Conflict: Constructive and Destructive Processes. *New Haven*,Yale University Press, 1973.

206. D. Gambetta. *Trust: Making or Breaking of Cooperative Relations*. Oxford, Blackwell, 1990.

207. T. Grandison, M. Sloman. A Survey Of Trust in Internet Applications. *IEEE Communications Surveys*, vol. 3, no. 4, pp. 2-16, 2000.

208. G. Suryanarayan, R.N. Taylor. A Survey of Trust Management and Resource Discovery Technologies in Peer-to-Peer Applications. *ISR Technical Report*, UCI-ISR-04-6, 2004.

209. A. Abdul-Rahman, S. Hailes. Supporting trust in virtual communities. *Hawaii International Conference on System Sciences*, Maui (Hawaii), 2000.

210. Q. Zhang, T. Yu, K. Irwin. A Classification Scheme for Trust Functions in Reputation-Based Trust Management. *Workshop on Trust, Security, and Reputation on the Semantic Web*, Hiroshima (Japan), 2004.

211. L. Mui, M. Mohtashemi, A. Halberstadt. A Computational Model of Trust and Reputation. In *35th Hawaii International Conference on System Science (HICSS)*, Maui (Hawaii), 2002.

212. L. Xiong, L. Liu. Building Trust in Decentralized Peer-to-Peer Electronic Communities. In *The 5th International Conference on Electronic Commerce Research. (ICECR)*, Montreal (Canada), 2002.

213. C-N. Ziegler, G. Lausen. Spreading Activation Models for Trust Propagation. In *IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04)*, Taipei (Taiwan), 2004.

214. M. Blaze, J. Feigenbaum, J. Lacy. Decentralized Trust Management. In *Proc. IEEE Symposium on Security and Privacy*, Oakland (CA), pp. 164-173, 1996.

215. W. Nejdl, D. Olmedilla, M. Winslett. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web. In *Proc. of the Workshop on Secure Data Management in a Connected World (SDM'04)*, Springer, Toronto (Canada), 2004.

216. M. Freedman, R. Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *ACM Conference on Computer and Communications Security*, Washington DC, 2002.

217. C. Shields, B. Levine. A Protocol for Anonymous Communication Over the Internet. In *ACM Conference on Computer and Communications Security*, Athens (Greece), pp. 33-42, 2000.

218. V. Scarlata, B. Levine. Responder anonymity and anonymous peer-to-peer file sharing. In *IEEE International Conference on Network Protocols*, Riverside (CA), 2001.

219. L. Xiong, L. Liu. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. *IEEE Transactions of Knowledge and Data Engineering*, vol. 16, no. 7, pp. 179-194, 2004.

220. K. Aberer. P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In *Proc. Ninth Int'l Conf. Cooperative Information Systems*, Springer-Verlag, pp. 179-194, 2001.

221. A. Rowstron, P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. IFIP/ACM Intl. Conf. on Dist. Systems and Platforms*, Heidelberg (Germany), Springer-Verlag, pp. 329-350, 2001.

222. B. Dragovic, E. Kotsovinos. XenoTrust: Event-based distributed trust management. *Second International Workshop on Trust and Privacy in Digital Business*, Prague (Czech Republic), 2003.

223. B. Dragovic, S. Hand. Managing trust and reputation in the XenoServer Open Platform. *First International Conference on Trust Management*, Crete (Greece), 2003.

224. S. Lee, R. Sherwood, B. Bhattacharjee. Cooperative peer groups in NICE. In *IEEE INFOCOM,* San Francisco (CA), 2003.

225. S. Banerjee, B. Bhattacharjee, C. Kommareddy. Scalable application layer multicast. In *Proc. ACM SIGCOMM*, Pittsburgh, pp. 205-217, 2002.

226. S. Song, K. Hwang, M. Macwan. Fuzzy Trust Integration for Security Enforcement in Grid Computing. In *Proc. NPC*, Wuhan (China), pp. 9-21, 2004.

227. S. Song, K. Hwang, Y.K. Kwok. Trusted Grid Computing with Security Binding and Trust Integration. *Journal of Grid Computing*, vol. 3, no. 1, pp. 24-34, 2005.

228. J.W. Lloyd. *Foundations of Logic Programming*, 2nd Edition, Springer-Verlag, 1987.

229. B. Grosof. Representing e-business rules for the semantic web: Situated courteous logic programs in RuleML. In *Proceedings of the Workshop on Information Technologies and Systems (WITS)*, New Orleans (LA), 2001.

230. B. Grosof B, T. Poon. SweetDeal: Representing agent contracts with exceptions using XML rules, ontologies, and process descriptions. In *WWW12*, Budapest (Hungary), 340-349, 2003.

231. I. Horrocks, P. Patel-Schneider. A proposal for an OWL rules language,. http://www.cs.man.ac.uk/ horrocks/DAML/Rules, accessed on 13th July, 2006.

232. J. Trevor, D. Suciu. Dynamically distributed query evaluation. In *PODS*, Santa Barbara (CA), pp. 28-39, 2001.

233. M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing*, vol. 7, no. 6, pp. 45-52, 2002.

234. A. Hess, J. Jacobson, H. Mills, R. Wamsley, K.E. Seamons, B. Smith. Advanced client/server authentication in TLS. In *NDSS*, 2002.

235. J. Basney, W. Nejdl, D. Olmedilla, V. Welch, M. Winslett. Negotiation Trust on the Grid. In *Workshop on Semantics in P2P and Grid Computing*, NY, 2004.

236. N. Li, J. Mitchell. RT: A Role-based Trust Management Framework. In *DARPA Information Survivality Conference and Exposition (DISCEX)*, Washington DC, pp. 201-212, 2003.

237. M. Mansouri-Samani. Monitoring Distributed Systems (A Survey). *Tech. Rep.*, DOC92/23, Imperial College, London, 1992.

238. M. Mansouri-Samani, M. Sloman. Monitoring distributed systems. *IEEE Network*, vol. 7, no. 6, pp. 20–30, 1993.

239. B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, R. Wolski. A Grid Monitoring Architecture, GWDPerf. *16–3, Global Grid Forum*, August 2002.

240. Z. Balaton, P. Kacsuk, N. Podhorszki, F. Vajda. Use Cases and the Proposed Grid Monitoring Architecture. *Tech. Rep. LDS-1/2001*, Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2001, available at http://www.lpds.sztaki.hu/publications/reports/lpds-1-2001.pdf, accessed on 13th July, 2006.

241. J. Case, M. McCloghrie, M. Rose, S. Waldrusser. Protocol Operations for Version2 of the Simple Network Management Protocol (SNMP). *IETF RFC 1905*, 1996.

242. http://h20229.www2.hp.com/, accessed on 13th July, 2006.

243. http://www3.ca.com/Solutions/Solution.asp?ID=1629, accessed on 13th July, 2006.

244. Orca. *http://www.orcaware.com/orca*, accessed on July 13th, 2006.

245. Mon. *http://www.kernel.org/software/mon*, accessed on July 13th, 2006.

246. Aide. *http://sourceforge.net/projects/aide*, accessed on July 13th, 2006.

247. Tripwire. *http://sourceforge.net/projects/tripwire*, accessed on July 13th, 2006.

248. M.L. Massie, B.N. Chun, B.E. Culler. Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, vol. 30, pp. 817–840, 2004.

249. Hawkeye Team. Hawkeye, Monitoring and Management Tool for Distributed Systems, 2004. *http://www.cs.wisc.edu/condor/hawkeye*, accessed on 16th July, 2006.

250. M. Litzkow, M. Livny, M. Mutka. Condor - A Hunter of Idle Workstations. In *Proc. of the 8th International Conference of Distributed Computing Systems*, pp. 104-111, 1988.

251. D. Thain, T. Tannenbaum, M. Livny. Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323-356, 2005.

252. B. Coghlan, A. Djaoui, S. Fisher, Magowan, M. Oevers. Time, information services and the grid. In *Advances in Database Systems (BNCOD): Supplement*

to the Proceedings of the 18 th British National Conference on Databases at RAL, pp. 9–11, 2001.

253. J.M. Schopf, M.D. Arcy, N. Miller, L. Pearlman, I. Foster, C. Kasselman. Monitoring and Discovery in Web Services Framework: Functionalities and Performance of Globus Toolkit's MDS4. *Tech. Report*, #ANL/MCS-P1248-0405, 2005.
254. K. Gor, R.A. Dheepak, S. Ali, L.D. Alves, N. Arurkar, I. Gupta, A. Chakrabarti, A. Sharma, S. Sengupta. Scalable Enterprise Level Workflow and Infrastructure Management in a Grid Computing Environment. In *Proc. Cluster Computing and Grid (CCGrid)*, Cardiff (Wales), pp. 661-667, 2005.
255. S. Andreozzi, D. Antoniades, A. Ciuffoletti, A. Ghiselli, E.P. Markatos, M. Polychronakis, P. Trimintzios. Issues about the Integration of Passive and Active Monitoring for Grid Networks. In *Proc. CoreGrid Integration Workshop*, Pisa (Italy), 2005.
256. X. Zhang, J. Freschl, J. Schopf. A performance study of monitoring and information services for distributed systems. In *Proc. of the 12th IEEE High Performance Distributed Computing (HPDC)*, IEEE Computer Society Press, Seattle (WA), pp. 270–282, 2003.
257. S. Zalikonas, R. Sakellariou. A Taxonomy of Grid Monitoring Systems. In *FGCS*, vol. 21, issue 1, pp. 163-188, 2004.
258. TNT Systems. The Role of IT Monitoring, Alerting, and Reporting in Satisfying Serbanes-Oxley Requirements. *TNT White Paper*, 2005.
259. European Data Grid. *www.cern.org/eu-datagrid*, accessed on 13th July, 2006.
260. L.A. Cornwall, J. Jensen, D.P. Kelsey, A. Frohner, D. Kouril, F. Bonnassieux, S. Nicoud, J. Hahkala, M. Silander, R. Cecchini, V. Ciaschini, L. dell'Agnello F. Spataro, D. O'Callaghan, O. Mulmo, G.L. Volpato, D. Groep, M. Steenbakkers, A. McNab. Security in Multi –domain Grid Environments. In *Journal of Grid Computing*, Kluwer Academic Press, 2004.
261. D. Boneh, and M. Franklin. Identity based Encryption from the Weil Pairing. *SIAM J. of Computing*, vol. 32, no. 3. pp. 586-615, 2003.
262. A. Shamir. Identity-based Cryptosystems and Signature Schemes. In *Proc. CRYPTO*, pp. 47-53, 1984.
263. Voltage® systems. *www.voltage.com*, accessed on 13th July, 2006.
264. http://www.cisco.com/en/US/products/ps6692/Products_Sub_Category_Home.html, accessed on 13th July, 2006.
265. DataPower. *www.datapower.com*, accessed on 13th July, 2006.
266. S. Vinoski. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, vol. 14, no. 2, pp. 46-55, 1997.
267. W. Grosso. *Java RMI*, 1st Ed., O'Reilly Publishers, 2001.
268. W. Ruben, M. Brain. *Understanding DCOM*, Prentice Hall, ISBN 0-13-095966-9, 1999.
269. E. Cerami. *Web Services Essentials*, O'Reilly, ISBN: 0596002246, 2002.
270. E.T. Ray. *Learning XML*, O'Reilley, ISBN: 0596004206, 2001.

271. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masintere, P. Leach, T. Berners-Lee. Hypertext Transfer Protocol – HTTP 1.1. *IETF RFC 2616*, 1999.
272. N. Mitra(Ed). SOAP Version 1.2, Part 0: Primer. *W3C Recommendations*, 2003.
273. R. Chinnici, J.J. Moreau, A. Ryman, S. Weerawarana. Web Services Description Language (WSDL) 1.1. *W3C Note*, 2001, available at http://www.w3.org/TR/wsdl20, accessed on 13th July 2006.
274. Computer Associates ®, IBM®, Microsoft®, Oracle®, SAP®, SeeBeyond Technologies®, Systinet®, and Others. UDDI v.3.0. *OASIS Standard*, 2005.
275. A. Nadalin, C. Kaler, R. Monzino, P. Hallam-Baker (Ed). Web Services Security: SOAP Message Security 1.1. *OASIS Standard Specification.*
276. D. Eastlake, J.R.D. Solo, M. Bartel,J. Boyer, B. Fox , E. Simon XML Signature Syntax and Processing. *W3C Recommendation*, 2002, available at http://www.w3.org/TR/xmldsig-core/, accessed on 13th July, 2006.
277. T. Imamura, B. Dillaway, E. Simon. XML Encryption Syntax and Processing. *W3C Recommendations*, 2002, available at http://www.w3.org/ TR/xmlenc-core/, accessed on 13th July, 2006.
278. S. Bajaj, D. Box, D. Chappell. Web Services policy Framework (WS-Policy). *IBM® Developer Works*, 2006.
279. S. Bajaj, D. Box, D. Chappell, et al. Web Services Policy Attachment (WS-PolicyAttachment). *W3C Member Submission*, 2006, available at http:// specs.xmlsoap.org/ws/2004/09/policy/ws-policyattachment.pdf, accessed on 13th July, 2006.
280. G. Della-Libera, M. Gudgin, P. Hallam-Baker, et. al. Web Services Security Policy Language (WS-SecurityPolicy), 2005, available at http://specs. xmlsoap.org/ws/2005/07/securitypolicy/ws-securitypolicy.pdf, accessed on July 13th, 2006.
281. S. Anderson, J. Bohren, T. Boubez, et. al. Web Services Secure Conversation Language (WS-SecureConversation). *OASIS Specification*, 2005, available at http://specs.xmlsoap.org/ws/2005/02/sc/WS-SecureConversation.pdf, accessed on July 13th, 2006.
282. E. Maler, P. Mishra, R. Philpott. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) v 1.1. *OASIS Standard*, 2003, available at http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf, accessed on 13th July 2006..
283. T. Moses (Ed). eXtensible Access Control Markup Language (XACML) Version 2.0. *OASIS Standard*, 2005, available at http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, accessed on 13th July, 2006.

Index

Advanced Reservation, 155
AES, 18
AGF, 166
Aide, 256
Akenti, 90
AON, 282
Application Filtering, 120
Application Level Firewalls, 163
ASPEED, 10
Athorization Credentials, 195
Attribute Certificates, 92
Authentication, 15
Authentication Credentials, 194
Authentication Header, 29

BGP, 175

Caesar Cipher, 17
CAN, 4
CAS, 80
CASCH, 9
Cerrtification Authority, 25
Certification Authority, 22
Chaining, 182
CHORD, 4
Circuit level Firewalls, 163
Confidentiality, 15
Confusion, 18
CORBA, 294
Core-based Key, 180
Cornell Theory Center. *See* CTC
CRL, 77
Cryptanalysis, 19
CTC, 10
CYSEP, 190

DAC, 70
DCOM, 294
DDoS, 111
Delegation Service, 64
Denial-of-Service, 108
DES, 18
DHT, 4
Diffie-Hellman, 20
Diffusion, 18
Distance Vector, 175
Distributed Hash Table. *See* DHT
DNS Hacking, 116
Dynamic Packet Filtering, 164

EALS, 88
EDG, 87, 271
Egress Filtering, 119
El Gamal, 18
Entropia VM, 152
ESP, 30
Exokernels, 146

Fabrication, 16
Firewalls, 161

Ganglia, 256
GGF, 49
Globus, 54
GlueDomains, 265
GMA, 251
Grid Definition, 3
Grid Evolution, 4
Grid Security Infrastructure. *See* GSI
Grid Taxonomy, 35
GridBank, 131
GridMap, 100
GridShib, 212
GSI, 50
GT4, 61

Hawkeye, 258
HMAC, 24
Hose, 171
Hosted Virtualization, 140
Hydra, 145

IBE, 281
Identity Credentials, 194
IDS, 122
Infiniband, 188
Information Security, 49
Integrity, 15
Interception, 16
Interruption, 16
IPSec, 29

Janus, 149
Java RMI, 294

KDC, 28
Kerberos, 27
Key Graphs, 180
KX.509, 208

L2VPN, 169
L3VPN, 169
LBAC, 69
LCMAPS, 274
Liberty Alliance, 209
Link State, 175
Link Testing, 123
LSF, 103

MAC, 22, 69
MAGI, 263
MD5, 23
MDS, 261
Message Level Security, 62
Message Passing Interface. *See* MPI
Mon, 256
Moore's Law, 2
MPI, 4
Multicasting, 178
MyProxy, 202

NICE, 233
Non-Repudiation, 16
Nova, 143

OGF, *See* GGF
OGSA, 49, 307
OGSI, 308
Orca, 255
OSPF, 175
OWL, 239

Packet Marking, 125
Parallax, 9
Parallel Virtual Machines. *See* PVM
Para-Virtualization, 141
Path Vector, 175
PBS, 103
PeerTrust, 228
PeerTrust Negotiation, 238
PERMIS, 94
P-Grade, 9
PKI, 24
Policy Certificates, 92
Priority Reduction, 156
Proof-Carrying-Code, 136
Proxy Certificate, 59
Proxy Certificates, 203
PUSH+ACK, 113
PVM, 4
PYRROS, 9

RBAC, 70
Rekeying, 181
Remus, 151
Replication, 16
Reputation, 217
RGMA, 259
RIP, 175
RSA, 18
RuleML, 239

SACRED, 195
SAML, 304
Schema Poisoning, 118
Secure Overlay Service, 120

SeGO, 236
Sensor Grids, 182
Service, 106
SETI@Home, 1
SHA-1, 23
Shibboleth, 210
Sinkhole Attacks, 185
SLIP, 177
Smart Cards, 200
Smurf, 114
SNMP, 254
SNORT, 122
SOAP, 295
SPIE, 125
SPINS, 186
SQL Injection, 118
SSL, 27
Static Packet Firewalls, 163
Stream Signing, 182
SweGrid, 131
Sybil Attacks, 184
SYN Flood, 112

TESLA, 186
TGS, 28
Throttling, 121
TinySec, 187
TLS. *See* SSL
Tripwire, 256
Trust, 216
TrustBuilder, 240

UDDI, 295
Use Condition Certificates, 92

VCMan, 206
Virtual Organization, 50
Virtual Smart Cards, 201
Virtualization, 138
VMWare GSX, 140
VOMS, 87
VPN, 168
VServer, 144

WATCHERS, 130
Wormhole Attacks, 185
WSDL, 295
WS-Policy, 301
WS-PolicyAttachment, 301
WS-Resource Framework, 310
WS-SecureConversation, 302
WS-Security, 299
WS-SecurityPolicy, 302

X.509, 24
XACML, 306
XDoS, 117
Xen, 142
XenoTrust, 231