Scientific Computing Department
Faculty of Computer and Information Sciences
Ain Shams University

SECURITY OF GRID COMPUTING ENVIRONMENTS

A Thesis Submitted to the Department of Scientific Computing, Faculty of Computer and Information Sciences, Ain Shams University, in the Partial Fulfillment of the Requirements for the Master Degree of Computer and Information Sciences

By:

Ahmad Mohammad Al-Shishtawy

B.Sc. Degree in Computer and Information Sciences
Demonstrator at Scientific Computing Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

Supervised By:

Prof. Dr. Mohammed Fahmy Tolba

Professor at Scientific Computing Department,
Faculty of Computer and Information Sciences.
Vice president for education and student affairs,
Ain Shams University.

Prof. Dr. Mohammed Said Abdel-Wahab

Professor at Scientific Computing Department.
Ex-Dean of the Faculty of Computer and Information Sciences.
Head of Information Systems Department.
Ain Shams University.

Dr. Eng. Ismail Abd Elhamid Taha

Associate Professor at Military Technical College,
Cairo, Egypt.

Cairo 2006

Table of Contents

List of Abbreviations

Abbreviation	*Meaning*
AJO	Abstract Job Object
API	Application Programming Interface
ASU	Ain Shams University
DAG	Directed Acyclic Graph
DAM	Data Analysis Module
DGM	Data Gathering Module
DNS	Domain Name Server or Domain Name System
GASS	Globus Access to Secondary Storage
GGF	Global Grid Forum
GIDA	Grid Intrusion Detection Architecture
GIIS	Grid Information Index Service
GIS	Grid Information Service
GMD	Grid Market Directory
GRAM	Globus Resource Allocation Manager
GRIS	Grid Resource Information Service
GSB	Grid Service Broker
GSC	Grid Service Consumer
GSI	Grid Security Infrastructure
GSP	Grid Service Provider
GUI	Graphical User Interface
GWU	George Washington University
IDA	Intrusion Detection Agent
IDB	Incarnation Data Base
IDE	Integrated Development Environment
IDL	Interface Description Language

Abbreviation	*Meaning*
IDS	Intrusion Detection Server
IP	Information Providers
IS	Information Science
IT	Information Technology
JMC	Job Monitor Controller
JPA	Job Preparation Agent
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LOA	Legion object address
LOID	Legion Object IDentifier
LVQ	Learning Vector Quantization
MDS	Monitoring and Discovery Service
MIR	Magnetic Resonance Imaging
MPI	Message Passing Interface
NIDS	Network Intrusion Detection System
NJS	Network Job Supervisor
OGSA	Open Grid Services Architecture
OID	Object IDentifier
OPR	Object Persistent Representation
P2P	Peer-to-Peer
PDA	Personal Digital Assistant
PKI	Public Key Infrastructure
QoS	Quality of Service
RP	Resource Proxy
RSA	Rivest, Shamir, and Adleman, the inventors of the RSA cryptosystem
RSL	Resource Specification Language

Abbreviation	*Meaning*
SDK	Software Development Kit
SHTTP	Secure Hyper Text Transfer Protocol
SOAP	Simple Object Access Protocol
SPMD	Single Program Multiple Dataset
SSL	Secure Socket Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TSI	Target System Interface
UNICORE	UNiform Interface to COmputer REsource
UP	User Proxy
Usite	UNICORE Grid site(s)
Uspace	UNICORE space
UUDB	UNICORE User Data Base
Vsites	Virtual sites
WSDL	Web Service Definition Language
XML	eXtensible Markup Language

List of Figures

List of Tables

Acknowledgments

I'd like to start by expressing my appreciation to Prof. Dr. Essam Khalifa the dean of our faculty for providing us with the appropriate climate for research by encouraging researchers and supporting us with the needed resources.

I'm also grateful to my supervisor Prof. Dr. Mohamed Fahmy Tolba for his guidance throughout my work by his observations and comments that kept me focused on achieving my goals.

I'd like to thank my supervisor Prof. Dr. Mohamed Said Abdel-Wahab for continuous support, helpful ideas, time, and encouragement. I also thank him for his efforts in fine-tuning my work till its final state.

Also I thank Dr. Ismail Abd Elhamid Taha for being my supervisor throughout my research and helping me with correct directions and advices.

I'd like to thank my parents very much for raising me up and helping me to be where I am now. I want also to tell them that they are the best parents in the world.

Finally I greatly appreciate my wife Marwa for standing by me and encouraging me to complete this work. Also big thanks to my colleague and best friend Ahmad Anbar.

Abstract

With the rapid advance in science, engineering, and business; people seek more computational power and resources to solve their problems more efficiently in terms of accuracy, time, and money. The field of Grid computing was born to fill the gap between available technology and increasing demand for computational power. The Grid provides a powerful computational environment by coupling distributed resources to enable seamless aggregation and sharing to create more powerful resource. The term distributed here does not refer only to geographical locations but also to administration that may span multiple organizations.

Security issues were addressed from the beginning of the Grid computing because of their importance to the success of such field. Intrusion detection is an important component of any modern security system because it is considered as a second line of defense against bugs and security holes as well as providing protection against insiders.

This thesis studies the problem of intrusion detection in Grid environments since it is considered as an important security issue. It introduces flexible cooperative distributed intrusion detection architecture for computational Grids. This work is based on the study of latest Grid projects and intrusion detection systems to deliver an architecture that suits and benefits form the underlying computational Grid environment.

A prototype implementation of the proposed architecture for the purposes of validation and verification is also introduces. The presented prototype uses homogeneous distributed intrusion detection servers that use the Learning Vector Quantization (LVQ) neural network for classification to detect intrusion cases if occurred.

The introduced prototype was tested against various Grid environments with different organizations and architectures through a Grid environment simulator that was developed to suit the study of security and intrusion detection. The test results showed the applicability of the proposed system in Grid environments and also showed distinct advances versus centralized systems. The thesis also presents the different parameters that may affect the proposed intrusion detection system showing and explaining their effects on the overall system performance.

Chapter 1

Introduction

Chapter 1: Introduction

This chapter introduces the concept of the Grid and presents how this field emerged. It describes the main components and the special characteristics of a working Grid environment that distinguishes it from other distributed or peer-to-peer systems. The chapter continues by introducing the motivation, objectives, and scope of work presented in this thesis.

1.1 General Knowledge about the Field of Grid

The Grid concept began to appear in the mid 1990s [34], it started as a project to link supercomputers at different sites [54] to solve state-of-the-art science, engineering, and business problems that did not fit on a single supercomputer either because the problem size was large or because it required a combination of different hardware and software that could not be combined in a single supercomputer. Because of the rapid advances in computers, high speed networks, and the Internet, this project grew far beyond its initial plans and goals to become what is now known as the Grid.

The Grid was inspired from the analogous advances in the electrical grids [56]. It is believed that the real revolution, which leads to the current advances in this area, was not because the invention of electricity, but because of the

construction of electrical power grids that grow in size rapidly to become international. After building the required infrastructure – generators, wiring, wall plugs, etc... – the consumers of electrical energy could satisfy their needs, because the deployed electrical grid will provide a consistent, dependable, reliable, pervasive, and relatively cheap source of electricity. Otherwise consumers of electricity will have to build their own generators (which the case nowadays for computational power) and this is in most cases infeasible because of the high costs, decrease of reliability and in some cases not possible because for example some generators have to be placed in special locations at waterfalls or wind.

Research in different fields in science and engineering is faster than the advances in the computer technology, this lead to sophisticated computational problems that can not be solved with the current available computational power. Solutions to this gap can be classified into three categories [23]:

- **Work Harder:** researchers in Information Technology (IT) and Information Science (IS) areas try to improve computer architecture and design to allow them to solve larger and more sophisticated computational problems. Today workstations are more powerful than early supercomputers.

- **Work Smarter:** researchers in IT & IS areas try to

improve the algorithms used to solve computational problems so they will work efficiently on the available computer technology. They also should seek to find new approaches and models to solve computational problems.

- **Get Help:** researchers in IT & IS areas try to solve large and sophisticated problems by allowing more than one computer to work together to solve one problem by dividing and distributing the task among available computers.

Distributed computing, cluster computing, and Grid computing are among research areas that fit into the last category of narrowing down the gap between current computational requirements and demands and available computing power. However, Grid computing is distinguished from conventional distributed computing by its focus on large-scale resource sharing, innovative applications, and in some cases, high-performance orientation [36]. The Grid is also distinguished from cluster computing by its lack of centralized control and single policy, its heterogeneous resources, and the fact that the state of the Grid system is not well known at any point of time.

Grid environments will pool the available resources to create a virtual supercomputer or Virtual Organizations [36] by coupling of these resources. These Virtual Organizations are similar to the temporary alliance between enterprises or organizations that share their resources and experiences to

improve business [53]. A resource in the Grid is anything that can be allocated including processor time, memory, secondary storage, databases, network bandwidth, special devices and sensors. The resources may be heterogeneous, geographically distributed, and owned by different enterprises or organizations. These pooled resources coupled by the Grid will enable solving problems that were not possible otherwise. It will also narrow the gap between the demands of science, engineering and business for computational power and the

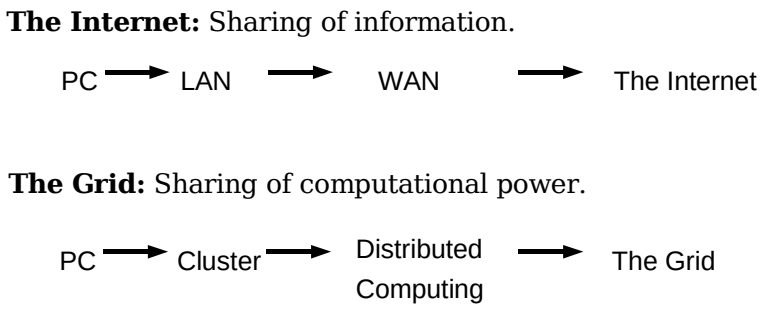

Figure 1.1: The evolution of the Grid compared to the Internet.

available technology. The Grid, as shown in *Figure 1.1*, has evolved in a way similar to the Internet, the difference is that the Internet is sharing of information while the Grid is sharing of computational power.

The Grid is needed for many reasons including for

example:

- Some resources that are needed to solve problems can not be efficiently duplicated such as an expensive supercomputer or an electrical telescope that must be placed at a specific geographical position. The Grid will enable remote access to such resources.

- Some problems need heterogeneous resources that are not available in the same machine, such as a combination of parallel and vector machines, to be solved efficiently. The Grid will enable such scenarios.

- Resources needed to solve a problem may be geographically distributed in different countries. The Grid will couple such resources in a seamless manner.

Grid Research aims to develop the necessary infrastructures that will make access to computational power as easy and reliable as access to electrical power, and create a seamless, integrated computational and collaborative environment to solve innovative applications [53]. This new field was known by several names such as metacomputing, scalable computing, global computing and more recently as the Grid or Grid Computing. The Grid was defined as "coordinated resource sharing and problem solving in a dynamic, multi-institutional virtual organizations" [36]. It has many special characteristics, requirements, and components

that distinguished it from other similar fields. It also enabled new application models that are more suitable to their nature.

The Grid includes special characteristics such as:

- **Multiple administrative domains and autonomy:** Resources in the Grid are controlled by different administrative domains and owned by different organizations. The autonomy of each domain must be protected, and the Grid infrastructure must cooperate with rather than replace the local policies at each domain [53].

- **Heterogeneity:** The resource pool in the Grid contains a collection of different resources that may range from electronic sensors and Personal Digital Assistants (PDAs) to supercomputers and large databases. These resources have different technologies and are controlled by different operating systems and software. The Grid should couple these resources seamlessly [53].

- **Scalability:** A Grid environment can grow from few computers to span the entire earth, the Grid infrastructure must be able to handle all resources and be scalable and flexible. The applications must be designed to handle possible degradation in performance when using large number of resources by designing the applications to be latency tolerant [53].

- **Dynamicity or Adaptability:** In the Grid the failure is the rule. Among all the resources in a grid environment, the probability of one component failure is high. The Grid must deal with such failures and allow resources to join or leave a grid environment as they want [53].

With these characteristics in mind, a working grid environment infrastructure must provide some basic services as shown in *Figure 1.2*. On top of this infrastructure, developers should build their grid enabled applications.

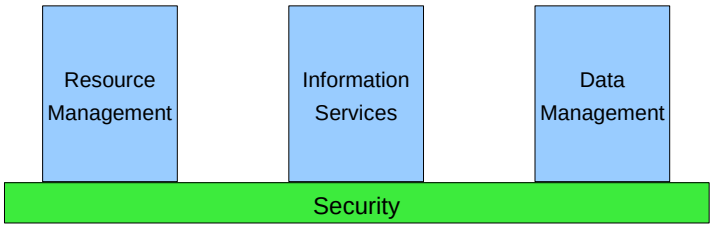

Figure 1.2: The basic Grid services.

The basic services of such Grid environments are:

- **Resource Management:** These services are required to manage the resources available in a Grid environment. The services span both basic services, such as resource allocation, up to more advanced services such as scheduling, co-allocation, advanced reservation, and payment management. The resource management should provide solutions to problems such as site autonomy, resource

heterogeneity, policy extensibility, and on-line control [50].

- **Information Service:** These are responsible of providing all required information about the Grid environment such as a directory service for easy search of available resources, naming service to provide uniform name space, monitoring and discovery services for monitoring resources and jobs running and discovering new resources. Thus allowing careful selection and configuration not only of computers, networks, and other resources but also of the protocols and algorithms used by applications [81].

- **Data Management:** Innovative applications usually need to deal with large amounts of data in most cases at remote sites. Thus large data collections are emerging as important community resources. The volume of interesting data is already measured in terabytes and will soon total petabytes. The Grid infrastructure must handle this data efficiently such as providing parallel transfer, replica management, processing subsets of huge datasets, and manage distributed data [98].

- **Security:** Considered the heart of the Grid, it spans all the services in the Grid. It must provide services such as secure communication and single sign on. Security in the Grid is complicated by the need to establish secure

relationships between a large number of dynamically created parties and across a range of administrative domains, each with its own local security policy [37].

To provide these services, the Grid infrastructure must develop the necessary Software Development Kits (SDKs), Application Programming Interfaces (APIs), and protocols [36]. The infrastructure must be open and build on top of available standards such as the Transmission Control Protocol/Internet Protocol (TCP/IP) and the Lightweight Directory Access Protocol (LDAP), and it should not require the replacement of existing site policies, operating systems, or network protocols. It must not enforce programming paradigm, language, or tools. It should also protect site autonomy, not compromise existing security, and be fault tolerant with no single point of failure [53]. To fulfill these requirements, the hour glass model represents a good example for both the Grid and the Internet [13]. Its narrow neck maps to a small set of core protocols that are used to build a larger set of protocols and applications at the top, and this small set can be mapped on different underlying technologies at the bottom. The Grid infrastructure consists of four major layers as shown in *Figure 1.3* [36][54]. These major layers are:

- **The Fabric Layer:** This layer consists of all the resources that will be shared by the Grid. This includes physical resources such as computers and scientific

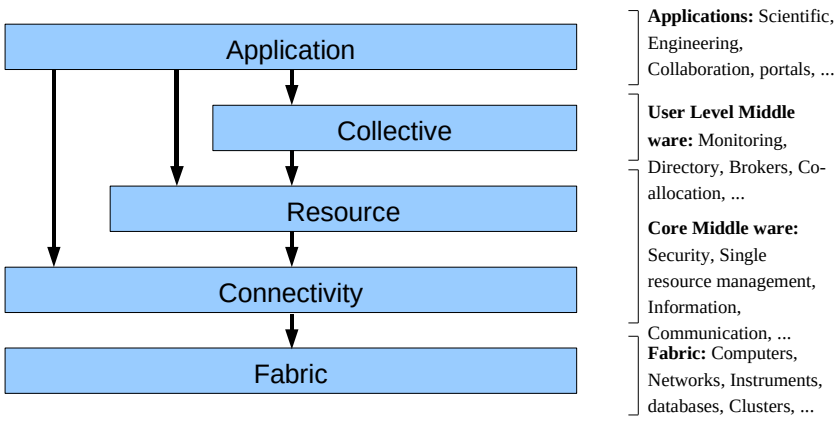

Figure 1.3: The layered Grid architecture.

instruments, and virtual resources such as clusters and distributed file systems. Each resource may include local protocols and management systems that are independent on the Grid. These Resources, according to their nature, must provide appropriate capabilities upon which upper services will be built such as program execution and monitoring, access to files and sub files, and reservation services.

- **The Connectivity Layer:** Protocols in this layer are responsible for providing communication and authentication services. This will enable the exchange of data between fabric layer resources and secure mechanism to verify the identity of communicating parties. Communication services include transport, routing, and naming. Security services include single sign on, delegation, and integration with local

security services.

- **The Resource Layer:** This layer is concerned with accessing to single resource in the fabric layer. This includes management protocols such as resource allocation, monitoring, controlling, and payment mechanisms. And information protocols to get information about resource structure, and status. Connectivity and resource layers are considered the core protocols – the tight neck of the hour glass – so they should be few and precisely specified. These core protocols are mapped on a large number of resources in the fabric layer and the upper level protocols will use them to build a wide variety of Grid enabled applications.

- **The Collective Layer:** This is build on top of the resource layer. It is used to provide coupling and coordinated access to a collection of resources and not associated with any single resource. This includes Directories for information about available resources, co-allocation, monitoring, diagnosing, workload management, data replication, and so on. Compared to Resource layer which is generic, this layer spans a wide spectrum from general purpose to highly application specific services that are build on top of the few generic resource layer services.

- **The Application Layer:** This layer contains the Grid enabled science, engineering and business applications.

These applications can be built on top of either the connectivity, resource, or collection layers or any combination of them.

The wide spread of Grid new concepts and methodologies in the past few years made a confusion about which system is a Grid and which is not. There was a strong need for a way to identify Grid systems. A three point checklist was introduced to classify Grid systems [31]:

- A Grid system should coordinate resources that are not subject to centralized control. So a scheduler deployed on a Local Area Network (LAN) is not a Grid system because of the scheduler central control nature. Grid systems integrate users and resources at different administrating domains and address communication, security, payment and all services presented before.

- A Grid system should use standard, open, general-purpose protocols and interfaces. Otherwise it is an application specific system. The Grid core middle ware reviewed before must use standard, open, general-purpose protocols.

- A Grid system should deliver nontrivial Qualities of Service (QoS). A Grid coordinates its resources to deliver various qualities of service, such as response time, throughput, availability, and/or co-allocation to adapt

complex user demands, so that the utility of the combined system is significantly greater than the sum of its parts.

Below are some of the major application classes that are enabled by the grid, and for which the grid will be used for [34]:

- **Distributed Supercomputing:** Uses the grid to couple supercomputers to solve problems that can not be solved on a single supercomputer such as distributed interactive simulations.

- **High-Throughput Computing:** Uses the grid to schedule a large number of loosely coupled or independent tasks on idle workstations. The nature of tasks led to different types of problems and problem solving methodologies such as parametric studies and hard cryptographic problems.

- **On-Demand Computing:** Uses the grid to satisfy the short term requirements of applications for a specific resource that can not be available locally in a cost effective manner. This differs from distributed supercomputing in that it is driven by cost-performance constraints not absolute performance. For example computer-enhanced Magnetic Resonance Imaging (MIR) uses supercomputers to achieve real-time image processing.

- **Data-intensive Computing:** Uses the grid to synthesize new information form data that is maintained in geographically distributed repositories, digital libraries, and databases. These applications are both computationally and communication intensive. For example future high-energy physical experiments will generate terabytes per day.

- **Collaborative Computing:** Uses the grid to enable and enhance human-to-human interactions. Applications are concerned with enabling the shared use of computational resources such as data archives and simulations.

1.2 Peer to Peer Computing

Peer-to-Peer (P2P) [5] is a new computing paradigm that employ distributed resources – computing power (cycles) , data (storage and content), network bandwidth, and presence (computers, human, and other resources) – available at the edge of the Internet to perform a critical function – distributed computing, data/content sharing, communication collaboration, and so on – in a decentralized manner[11][19]. The Grid and P2P technologies are two different approaches for distributed computing that are close to each other and have the same ultimate goal of the pooling and coordinated use of large sets of distributed resources. The difference is that each of these technologies started – aiming to achieve the ultimate goal – from different points of interests, this is because of the

different communities, design requirements, and applications targeted by both technologies. It is expected for these technologies to converge in the future [33]. The remainder of this section briefly reviews P2P and contrasts the differences between it and the Grid technologies based on real systems not theoretical assumptions and goals.

P2P started to take enormous interest with the appearance of Napster music sharing [67]. After that, many P2P applications and infrastructures appeared such as Gnutilla [20], SETI@home [14], and JXTA [48] among others. The main design goals of P2P include [19]:

- **Cost sharing/reduction:** In client/server model the server that served many clients was responsible for the majority of the system's cost that may grow very large with the system. P2P systems share the cost of ownership among the peers. For example file sharing systems divide the storage space needed by the shared files among peers.

- **Improved scalability/reliability:** The lack of central server helps improving system scalability and reliability. But this requires new innovative algorithms for resource discovery and search that is a hot topic with many research projects.

- **Resource aggregation and interoperability:** Each node in a P2P network provide a small amount of needed

resource – such as storage space or processor cycles – that must be integrated into a large system to enable applications that benefits from huge amounts of these resource to solve the larger problems.

- **Increased autonomy:** This means that all data and work of the user of a P2P system be performed locally without relaying on any centralized server.

- **Anonymity/privacy:** A peer in the network may not want anyone or any service provider to know about his environment in the system. This is related to autonomy. It is difficult to keep anonymity with the existence of central server because a server may identify its clients with at least their IP address. In P2P systems users may avoid having to provide any information about themselves to anyone else.

- **Dynamism:** The computing environment of any P2P system is highly dynamic. With resources joining and leaving dynamically and unpredictably. P2P applications must support this highly dynamic nature.

- **Enabling ad-hoc communication and collaboration:** This is related to dynamism. Ad-hoc refers to environments were members come and go based perhaps on their current physical location or current interests. P2P applications must take into account changes in the group of participants.

In addition to these goals P2P system must address issues such as self organization, performance, security, transparency, usability, fault resilience, and interoperability.

P2P targets resources on the edge of the Internet that considered in the past to be dump useless clients but now – with the advance in technologies and lower cost – grew to be interesting. Resources on the edge of the Internet have unstable connectivity and unpredictable IP addresses. Because resources may connect or disconnect unpredictably and obtain IP address dynamically from ISPs. This is completely different from Internet servers having stable always on connection with fixed IP Address. This means that P2P must operate outside the Domain Name Server (DNS) system and have significant or total autonomy from central servers. P2P should also be able to handle a large number (millions) of resources around the Internet owned by anonymous users. On the other hand the Grid started as a project to link super computers and expanded to scientific collaborations. This environment is more stable with well known moderate size (thousands) resources with high availability and more professional administration than P2P and stronger trust relationship between resource owners.

The Grid couples more powerful and diverse resources with better connectivity than P2P resources. Grid resources – such as databases, scientific instruments, clusters, and supercomputers – have in general more value and are

administrated in a more organized way according to some well known policies. In contrast P2P deals with nonuniform and highly variable behavior. The majority of P2P resources are normal home computers.

Grid technologies are used to solve a wide variety of scientific problems according to the requirements of the scientific collaborations sharing their resources. On the other hand, P2P systems are more specialized to solve a particular specific resource sharing problem such as file sharing. More work has been spent in the Grid on infrastructure that provides the required basic services presented in the previous sections and this infrastructure assumes at least limited level of trust and did not address the situation of the absence of trust which is the case in P2P computing dealing with anonymous resources. The Grid infrastructure is based on open, general purpose and standard protocols which helps the integration of diverse grid infrastructure. In contrast each P2P application defines its own protocol that solves a specific problem of that application.

According to I. Foster, and A. Iamnitchi it is concluded that [33]:

- Both P2P and Grid computing are concerned with the same general problem, namely, the organization of resource sharing within virtual communities.

- Both take the same general approach to solving this problem. Namely the creation of overlay structures that coexist with, but need not correspond in structure to, the underlying organizational structures.

- Each has made genuine technical advances, but each also has crucial limitations, which summarized as Grid computing addresses infrastructure but not yet failure, whereas P2P addresses failure but not yet infrastructure.

- The complementary nature of the strengths and weakness of the two approaches suggests that the interests of the two communities are likely to grow closer over time.

1.3 Problem Motivation

Grid computing is a new and fast growing field that came to fill the gap between the increasing need for processing power and available technologies [34]. This new field, Grid computing, is considered as one of the top ten technologies that will change the future [102]. The name came from an analogy to the electrical power grid, to show the ultimate goal of the Grid computing field which is making access to computational power as easy as the access to electricity. Security was the heart of Grid computing from its early beginnings because designers of its infrastructure knew that it will not succeed without efficient security and that securing the Grid will be hard to add later [37]. The importance of

security came from the power of this new Grid environment that will enable seamless access to powerful resources. This turns the Grid to an attractive target for many attackers who may want to misuse the Grid powerful resources.

The Grid computing field is still under research, and so are the security mechanisms that are used to protect Grid environments. Both research domains, Grid computing and Grid security mechanisms, are growing and evolving through time which motivates the research. Security mechanisms had started with simple password based systems and evolved to Public Key Infrastructure (PKI) based systems. With many requirements and services needed by both networks and Grid users, other security mechanisms had also evolved such as authentication, authorization, single sign on, and encryption [37].

Intrusion detection is considered as a second line of defense and it is needed to detect attackers when first line security mechanisms fail to prevent these attacks. In addition intrusion detection provides protection against legitimate users of the system that is not possible with security mechanisms. Unfortunately intrusion detection mechanisms have not been addresses yet by researchers in the field of Grid security. This motivates the research work presented in this thesis and encouraged the researcher to also study and get an insight into the problem of intrusion detection in Grid environments to

build a Grid enabled intrusion detection system for the purpose of improving the security of future Grids.

1.4 Objectives and Scope of Work

The main objective of this thesis is to design and develop a Grid enabled intrusion detection system as a second line security mechanism that can prevent Grid resources from different attack types.

Based on this main objective the scope of this research work can be stated in some directions as follows:

- Study and analyze the current Grid architectures and projects to understand and synthesis the requirements and constraints of this field in order to underlying and identifying the capabilities of current security mechanisms.

- Study and comprehend the current intrusion detection systems while keeping the Grid constraints in mind to identify how these intrusion detection systems are compatible with Grid environments and to specify their limitations, capabilities, and suitableness.

- Propose a Grid Intrusion Detection Architecture based on the findings in the Grid and Intrusion detection fields. The proposed architecture should be compatible with various Grid environments and should comprehensively address their needs.

- Design and develop a prototype implementation of the proposed Grid Intrusion detection architecture to prove its applicability in grid environment and to test its performance in various Grid environments through computer simulation.

- Analyze the obtained experimental results from the various experiments to check the applicability of the proposed system and to determine the effect of different parameters on the overall system performance.

1.5 Thesis Organization

This thesis has four main parts. The first part is a literature review of the Grid field. This part includes chapter 1 and chapter 2. Chapter one presents the literature review of the Grid computing field and introduces different Grid computing infrastructures showing their components, services, layers, requirements, and constraints. Chapter two picks four of the most famous current Grid computing projects and illustrates them by depicting their architectures, basic components, vision, and how they meet the basic Grid requirements.

The second part presents security issues related to the Grid and intrusion detection. Chapter Three focuses on the Grid security issues and it depicts different Grid Security Infrastructures and identifies their capabilities and limitations. It ends by presenting a Grid research project between Ain

Shams University in Egypt and George Washington University in USA to build a Grid enabled application. The aim of the application is to help us, through hands on practice, to better understand the new Grid environments specially its security mechanism. The latest research work in the field of intrusion detection is introduced in chapter four. It presents different approaches and classifications of intrusion detection systems and identifies the advantages and disadvantages of each presented approach.

The third part, chapter five, presents the proposed Grid Intrusion Detection Architecture and its different modules that are based on the survey of the previous chapters. The proposed architecture was designed with almost all the Grid characteristics and requirements in mind to make it compatible with Grid environments. The components of the proposed system and their functionalities and features are illustrated in this chapter.

The forth part includes chapter six and seven. Chapter six presents an implementation of the proposed architecture. The presented implementation is based on homogeneous intrusion detection servers that employ neural networks techniques using the Learning Vector Quantization (LVQ) neural network. The Grid environment was simulated to facilitate testing in different environments with different characteristics. A computer simulator was developed to enable

the testing of different security related issues. Chapter seven examines the experimental results obtained from various simulated experiments of different Grid environments. It presents the effect of different parameters on the conducted experiments against the proposed GIDA performance. These results provided a deep insight on better understanding the problem of intrusion detection in Grid environments by underlying the conditions affecting their performance. Moreover, the obtained results may be exploited in designing more enhanced and fine tuned future intrusion detection systems.

The final conclusion of this research is presented in Chapter eight along with possible future directions and expansions of the presented research.

Chapter 2

Analysis of some Grid Architectures

Chapter 2: Analysis of some Grid Architectures

This chapter introduces four of the Grid projects, namely, Globus, GridBus, Legion, and UNICORE. Showing their architectures, basic components, and how they meet the basic Grid requirements presented in Chapter 1.

2.1 Globus

The Globus project [93] is a US R&D project aiming to put standards for the Grid infrastructure and to develop the open source Globus ToolKit that facilitates the construction of Grids and Grid applications.

The Globus ToolKit is used by many Grid Communities as a technology base [35]. It is a community based, open architecture, open source set of services and software libraries that support Grid and Grid applications [32]. It is packaged as a set of components that can be used either independently or together to develop Grid enabled applications. These components provide the basic services needed by Grid applications and can be grouped into four basic categories – resource management, information services, data management, and security – as shown in *Figure 1.2*. The ToolKit adopts a layered architecture, as shown in *Figure 1.3*, and for each component it defines the needed protocols and application

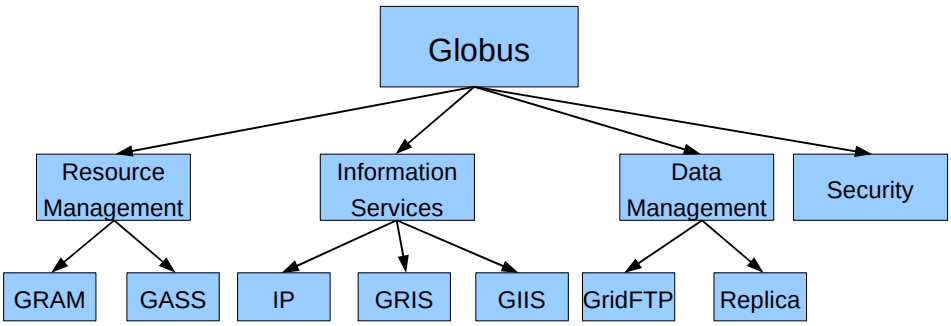

Figure 2.1: The Globus ToolKit components.

programming interfaces (APIs). It allows sharing of resources – computing power, scientific instruments, databases, and so on – across corporate, institutional and geographic boundaries seamlessly while keeping local autonomy. The Globus ToolKit components, as shown in *Figure 2.1*, are:

- **Resource Management:** This main component is responsible for providing services for resource allocation, job submission and monitoring, and result gathering. It consists of two components:

  - **Globus Resource Allocation Manager (GRAM):** GRAM [50] is responsible for remote execution and status reporting. A user can execute a job on a remote host by contacting the gatekeeper – a GRAM component – on that host. The gatekeeper will mutually authenticate with the user – the UP acting on his behalf – and then checks the Grid map file – a file containing mapping

between global user name and local account or ID – to see if the user is authorized to use the resource. If the user is authenticated and authorized the gatekeeper receives the job details through the Globus Resource Specification Language (RSL) and then starts a job monitor that initiates and monitor the job execution.

- **Globus Access to Secondary Storage (GASS):** GASS [42] is used for accessing remote files. It is used for staging-in executables and input files before starting the job and retrieving outputs after finishing the job. GASS can also be used for redirecting standard output and standard error streams of a job. GASS is GSI enabled and uses Secure Hyper Text Transfer Protocol (SHTTP) based streams.

● **Information Services:** This main component is responsible for delivering static and dynamic information about the available resources. In the Globus ToolKit they are called the Monitoring and Discovery Service (MDS) [51]. The information is represented as a hierarchy of entries with zero or more attribute-value pairs based on the Lightweight Directory Access Protocol (LDAP) [84]. MDS has three-tier structure:

- **The Information Providers (IP):** They are at the bottom of the structure. Each resource should have one or

more information provider that is responsible for gathering data about a specific system attribute or status. There are standard information providers with the Globus ToolKit that provide generic information such as the CPU type, memory size, free disk space, and operating system. The resource administrators could implement their own information provider to gather any needed properties of the resource. After gathering the needed data it should be converted into standard format (LDAP) and then published into the GRIS described below to make it available to every one.

- **The Grid Resource Information Service (GRIS):** There could be multiple information providers on each resource but only one GRIS. The GRID receives the data published by the information providers and responds to queries about the resource attributes. For dynamic data, the GRIS updates its database based on a time-to-live value by querying the relevant information provider.

- **The Grid Information Index Service (GIIS):** They are at the top of the structure. A GIIS receives resource information from other GRISs and GIISs registered with them. The GIIS indexes the received aggregated information about the resources and facilitates efficient searches for multiple resources by querying one GIIS.

- **Data Management:** Innovative scientific applications demand efficient access to large data sets that are growing in size and sometimes in distribution. The data management components are responsible for providing utilities and libraries for transmitting, storing and managing massive data sets efficiently and securely. In the Globus ToolKit there are two main components for data management [7]:

  - **GridFTP:** This component is a GSI-enabled, efficient, and reliable data movement extension of the standard FTP protocol. GridFTP supports third-party control of transfer, parallel transfer, striped transfer, partial file transfer, automatic negotiation of TCP buffer/window sizes, and support for reliable and resumable transfer.

  - **Replica Location and Management:** This component is responsible for managing complete and partial copies of data sets. This is done by registering files and their copies in a Replica Catalog. A file is identified by its Logical File Name (LFN) that is mapped to real names at different locations of the file. Replica Location Service (RLS) is used to registering files and then creating and deleting its replicas.

- **Security:** The Globus ToolKit implements the Grid

Security Infrastructure (GSI) presented in the next chapter. This component provides services for authenticating users and resources, single sign on, and delegation among others. It is based on standards such as the Secure Socket Layer (SSL), Public Key Infrastructure (PKI), and the X.509 standard for encoding certificates.

The Globus Project also tries to define new Grid standards through the Open Grid Services Architecture (OGSA) [38] framework. The OGSA look at the Grid from a service point of view. A service is defined as a network-enabled entity providing some capability needed by the Grid such as computational resource, storage resource, security and so on. OGSA is based on Web Services (WS) and eXtensible Markup Language (XML) standards such as Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), and WS-Inspection. The OGSA was presented at the Global Grid Forum (GGF) on 2002 [24] which has set up an Open Grid Services working group to review, refine, and document the Grid service architecture.

2.2 Legion

Legion [95] [1] is an object-based meta-systems software project started in late 1993 at the University of Virginia. Legion provide its users, working from their workstations, the illusion of a single virtual computer by

combining different resources such as digital libraries, physical simulations, cameras, linear accelerators, and video streams. It also supports user groups and collaborations through the construction of shared work spaces. All complexities are hidden from the users by Legion's transparent scheduling, data management, fault tolerance, site autonomy, and a wide range of security options.

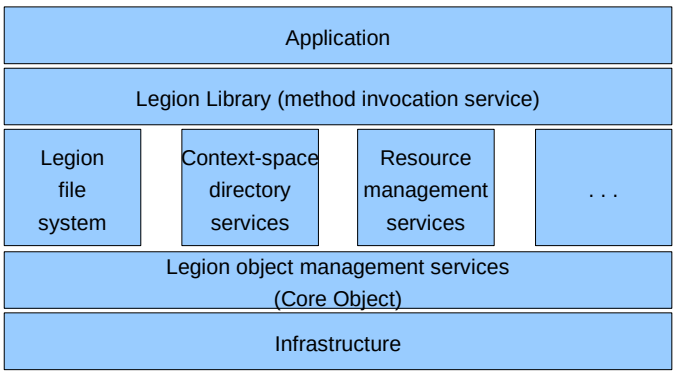

Figure 2.2: The Legion architecture

Legion is an open and flexible system that can be adapted to new and changing users needs. It is designed to encourage third party development of new or updated applications, run-time library implementations, and core components. As shown in *Figure 2.2* Legion is a middle ware that sets on top of the user's resources and operating systems. It can protect its own resources against other Legion users, so that administrators can choose appropriate policies for who uses which resources under what circumstances. To allow

users to take advantage of a wide range of possible resources, Legion offers a user-controlled naming system called *context space*, so that users can easily create and use objects in far flung systems. Users can also run applications written in multiple languages, since Legion supports interoperability between objects written in multiple languages. The Legion philosophy can be summarized in the following points:

- **Everything is an object:** Each hardware or software resource accessible through the Legion Grid will be represented by a Legion object. A Legion objects is an active process that has member functions invocable by other Legion objects. Legion defines the message format and high-level protocol for object interaction, but not the programming language or the communications protocol.

- **Classes manage their instances:** A class object is an active Legion object that define and manage other Legion objects. Class objects are managers and policy makers that are given system-level responsibilities such as creating new instances, schedule their execution, activate and deactivate them, and provide information about their current location to client objects that wish to communicate with them. Classes whose instances are themselves classes are called *metaclasses*.

- **Users can provide their own classes:** Users are

allowed to define and build their own class objects and even change the system-level mechanisms that support their objects. Legion 1.4 (and future Legion systems) contains default implementations of several useful types of classes and metaclasses. Users are not forced to use these implementations if they do not meet their performance, security, or functionality requirements.

- **Core objects implement common services:** Legion defines the interface and basic functionality of a set of core object types that support basic system services, such as naming and binding, object creation, activation, deactivation, and deletion. Core Legion objects provide the mechanisms that classes use to implement policies appropriate for their instances. Examples of core objects include hosts, vaults, contexts, binding agents, and implementations.

Legion objects are independent, logically address-space-disjoint active objects that communicate with one another via non-blocking method calls that may be accepted in any order by the called object. Each method has a signature that describes the parameters and return value, if any, of the method. The complete set of method signatures for an object fully describes that object's interface, which is determined by its class. Legion class interfaces can be described in an Interface Description Language (IDL), several of which will

be supported by Legion.

Legion implements a three-level naming system. At the highest level, users refer to objects using human-readable strings, called context names. Context objects map context names to Legion Object IDentifiers (LOIDs), which are location-independent identifiers that include an RSA public key. Since they are location independent, LOIDs by themselves are insufficient for communication; therefore, an Object Identifier (OID) is mapped to a Legion Object Address (LOA) for communication. A LOA is a physical address (or set of addresses in the case of a replicated object) that contains sufficient information to allow other objects to communicate with the object (e.g., an <IP address, port number> pair).

Legion will contain too many objects to simultaneously represent all of them as active processes. Therefore, Legion requires a strategy for maintaining and managing the representations of these objects on persistent storage. A Legion object can be in one of two different states, active or inert. An inert object is represented by an Object Persistent Representation (OPR), which is a set of associated bytes that exists in stable storage somewhere in the Legion system. The OPR contains state information that enables the object to move to an active state. An active object runs as a process that is ready to accept member function invocations; an active object's state is typically maintained in the address space of the

process (although this is not strictly necessary).

Several core object types, as shown in *Figure 2.3*, implement the basic system-level mechanisms required by all Legion objects. Like classes and metaclasses, core objects are replaceable system components; users (and in some cases resource controllers) can select or implement appropriate core objects.

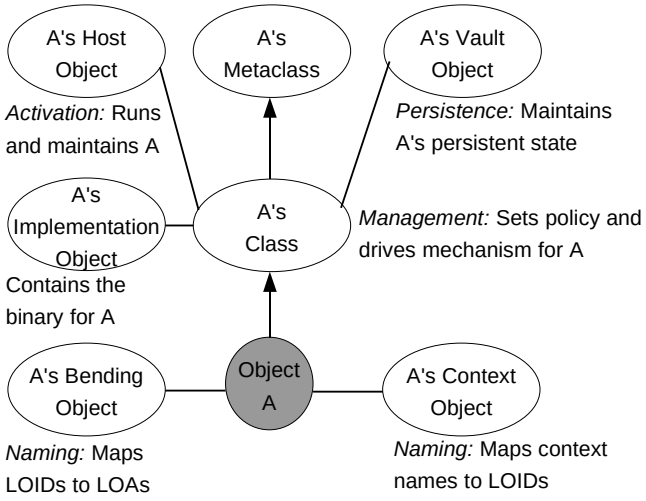

Figure 2.3: Legion object model and relationships

- **Host objects:** Host objects represent processors in Legion. One or more host objects run on each computing resource that is included in Legion. Host objects create and manage processes for active Legion objects. Classes invoke the member functions on host objects in order to activate instances on the computing resources that the hosts represent. Representing computing resources with Legion

objects abstracts the heterogeneity that results from different operating systems having different mechanisms for creating processes. Further, it provides resource owners with the ability to manage and control their resources as they see fit.

- **Vault objects:** Just as a host object represents computing resources and maintains active Legion objects, a vault object represents persistent storage, but only for the purpose of maintaining the state, in OPRs, of the inert Legion objects that the vault object supports.

- **Context objects:** Context objects map context names to LOIDs, allowing users to name objects with arbitrary high-level string names, and enabling multiple disjoint name spaces to exist within Legion. All objects have a current context and a root context, which define parts of the name space in which context names are evaluated.

- **Binding agents:** Binding agents are Legion objects that map LOIDs to LOAs. A <LOID, LOA> pair is called a binding. Binding agents can cache bindings and organize themselves in hierarchies and software combining trees, in order to implement the binding mechanism in a scalable and efficient manner.

- **Implementation objects:** Implementation objects allow other Legion objects to run as processes in the system. An implementation object typically contains machine code

that is executed when a request to create or activate an object is made; more specifically, an implementation object is generally maintained as an executable file that a host object can execute when it receives a request to activate or create an object. An implementation object (or the name of an implementation object) is transferred from a class object to a host object to enable the host to create processes with the appropriate characteristics.

2.3 UNICORE

The UNICORE [96][40] project – UNiform Interface to COmputer REsource – is funded by the German Ministry of Education and Research. Its goal is to provide seamless, secure and intuitive access to distributed computing resources, applications and data. It is a vertically integrated Java based Grid computing environment. It provides the users with easy to use graphical user interface to create, submit, monitor, and control jobs. To achieve its goal UNICORE has to [18]:

- Hide the heterogeneity resulting from different hardware architectures, vendor specific operating systems, incompatible batch systems, different application environments, historically grown computer center practices, naming conventions, file system structures, and security policies.

- Build security into the UNICORE design from the

start. Security is based on the emerging X.509 standard for certificates authenticating servers, software, and users and encrypting the communication over the Internet.

- Be usable by scientists and engineers without having to study vendor or site specific documentation. A Graphical User Interface (GUI) was developed to assist the user in creating and managing complex jobs and to integrate important applications. UNICORE was designed to be adapted to existing proven practices at the participating centers.

- The administrative autonomy of participating sites had to be retained, including the decision of who may use the resources.

UNICORE has developed a rich set of core functions. These functions allow users to create and manage complex batch jobs that can be executed on different systems at different sites. All complexities are hidden by UNICORE. These core functions are [18]:

- **Job creation and submission:** The user can create complex and interdependent jobs using a graphical interface. These jobs can be executed on any UNICORE site without changes to the job definitions. A UNICORE job consists of a group of jobs and is represented by an Abstract Job Object (AJO) which is submitted to a user selected site

for execution. UNICORE ensures that a successor is executed only if all predecessors have completed successfully and all necessary data sets are available at the target system.

- **Job Management:** The job management system gives the user full control over jobs and data through the graphical user interface. Also detailed log information is available to analyze error conditions. The job output that is written to standard output stream (stdout) and standard error stream (stderr) by the execution systems can be reviewed or transferred to the client workstation. Jobs may be terminated and removed from the UNICORE grid by the user.

- **Data management:** Each job group has a temporary UNICORE space (Uspace) for storing needed files. During job creation the user specifies which data sets are to be imported into or exported from the Uspace or transferred to a different Uspace. UNICORE performs the necessary data movement at run time without user intervention.

- **Application support:** UNICORE supports the creation of custom plugins to provide GUI to scientific and engineering application packages without a graphical user interface.

- **Flow control:** The job model can be described as a set of one or more Directed Acyclic Graphs (DAGs).

● **Meta-computing:** UNICORE is extended to the simultaneous use of two or more systems by one application through its support for Message Passing Interface (MPI) libraries. But UNICORE do not attempt to co-schedule systems because most of available resources do not support advanced reservation which is a prerequisite for co-scheduling.

● **Single sign-on:** Single sign-on is provided through the X.509V3 certificates that are mapped to local account at each UNICORE side. The site has full control on weather or not to grant access to the user based on his unique identifier - global name - or information in his certificate.

● **Support for legacy jobs:** Traditional batch processing is supported by allowing users to include their old job scripts as part of a UNICORE job. This will simplify migration but on the other hand does not guarantee seamlessness.

● **Resource management:** At the time of the job submission, UNICORE users have information about the currently valid resources. So the users can select the target system and specify the required resources. The UNICORE client is in a position to verify the formal correctness of jobs with respect to resources and alert users to correct errors immediately.

The UNICORE architecture is shown in *Figure 2.4*. It is a three-tier model consisting of the UNICORE client running on the users PC, The UNICORE Grid site(s) (Usite) defined by each participating computer center at the top level that the clients connect to, and finally each Usite is organized as one or more Virtual sites (or Vsites) which can represent the execution and/or storage systems at the computer centers. The architecture shown in *Figure 2.4* is an example of a UNICORE system consisting of two Usites with a total of three Vsites.

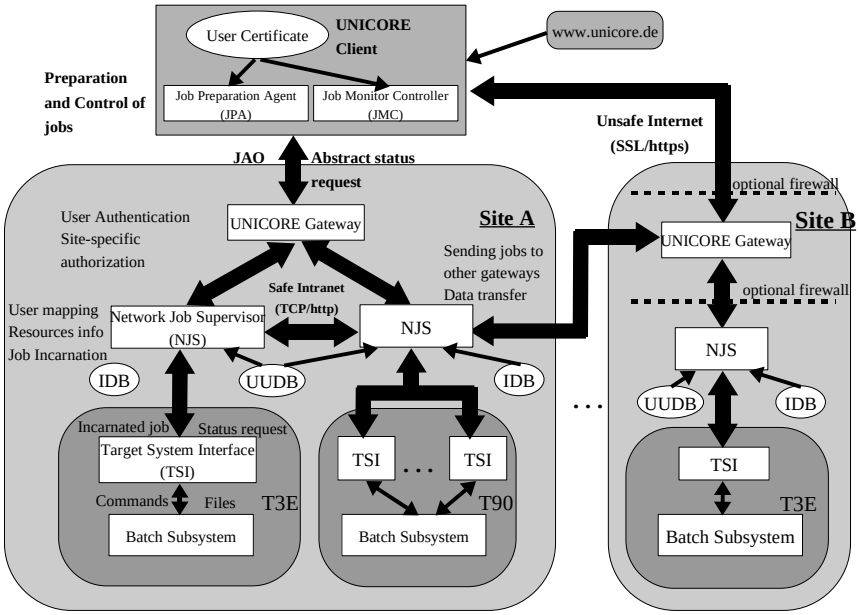

Figure 2.4: The UNICORE architecture

The UNICORE client is a Java application that the user uses to connect to a Usite gateway. A list of available UNICORE Gateways can be found as an XML document at

www.unicore.de or configured by the user. The user certificate is needed to authenticate with the gateway and to sign the submitted jobs. The jobs are created at the Job Preparation Agent (JPA) part of the Client. The Job Monitor Controller (JMC) part of the Client is used to monitor the status and results of the running jobs. The jobs, status requests, or the results are formulated in an abstract form using the Abstract Job Object (AJO) Java classes.

The UNICORE Gateway provides an Internet address and a port accessible from the outside for SSL connections. It is a single entry point for all UNICORE connections into a Usite. A Gateway can be installed inside or outside of a firewall depending on the site's security requirements.

The UNICORE Vsite consists of the Network Job Supervisor (NJS) and a Target System Interface (TSI). The NJS Server manages all submitted UNICORE jobs. It uses the UNICORE User Data Base (UUDB) to authenticate the user by looking for a mapping of the user certificate to a valid login. The NJS incarnates jobs from the abstract AJO definition into the appropriate concrete command sequence for a given target execution system, and hands the incarnated tasks and jobs over to the TSI. The incarnation is based on the specifications in the Incarnation Data Base (IDB). The NJS also checks the dependencies between job components, schedule the components accordingly, stores all job status and result

information, and replies to status and result requests from the client. In case of sub-jobs which are specified to run on a Vsite at a different Usite, the NJS takes the role of a Client and submits the sub-job to the remote Gateway.

The UNICORE Target System Interface (TSI) accepts incarnated job components from the NJS, and passes them to the local batch systems for execution. In addition, file import and export tasks are handled by the TSI. Moreover it implements low level status reporting and control of batch jobs.

2.4 GridBus

The GridBus project [94][74] is an open source software toolkit that extensively leverages related software technologies and provides an abstraction layer to hide idiosyncrasies of heterogeneous resources and low-level middleware technologies from application developers. It focuses on realization of utility computing and market-oriented computing models scaling from clusters to grids and to peer-to-peer computing systems. The research and innovation project is led by the University of Melbourne GRIDS Lab with support from the Australian Research Council.

The idea of a computational economy helps in creating a service-oriented computing architecture where service providers offer paid services associated with a particular

application and users, based on their requirements. These requirements will be optimized by selecting the services they require and can afford within their budget. Gridbus emphasizes the end-to-end quality of services driven by computational economy at various levels – clusters, peer-to-peer (P2P) networks, and the Grid – for the management of distributed computational, data, and application services.

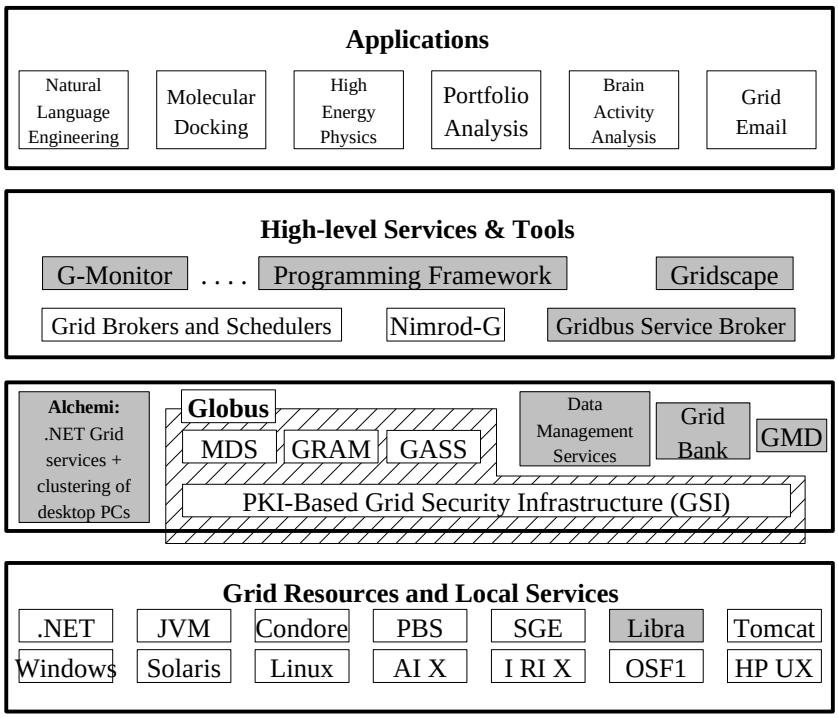

Figure 2.5: The GridBus architecture

The layered architecture depicting the GridBus components in conjunction with other middleware technologies – Globus and Unicore that have been reviewed

before and Alchemi – is shown in *Figure 2.5.* Items in shaded boxes are pursued by the Gridbus project. Gridbus provides software technologies that spread across the following categories [75]:

- **Visual Parameter Sweep Application Composer:** This is a Java based Integrated Development Environment (IDE) for rapid creation of parameter sweep (data parallel/SPMD) applications. It also allows the rapid creation and manipulation of the parameters. While being flexible, it is also simple enough for a non-expert to create a parameter script, known as a plan file, within minutes. The composed parameter sweep applications can be deployed on global Grids using the Gridbus resource broker.

- **G-Monitor:** This is a web portal for initiating, monitoring and steering application execution on global grids. It uses services provided by Grid Service Brokers (GSBs) such as Nimrod-G and Gridbus Broker to deploy applications. It allows users to manage their Grid credentials and provides secure access to remote hosts running brokers. The users can either upload applications and data at runtime or select from those already available on the broker host. G-Monitor provides an easy to use interface for the end-user to monitor and control jobs running within the Grid environment.

- **Gridbus Grid Service Broker:** This component makes scheduling decisions on where to place the jobs on the Grid depending on the computational resources characteristics (such as availability, capability, and cost), the users QoS requirements such as the deadline and budget, and the proximity of the required data or its replicas to the computational resources. It also has the capability to locate and retrieve the required data from multiple data sources and to redirect the output to storage where it can be retrieved by processes downstream. It has the ability to select the best data repositories from multiple sites based on availability of files and quality of data transfer.

- **Grid Market Directory (GMD):** serves as a registry for high-level service publication and discovery in virtual organizations. It enables service providers to publish the services which they provide along with the costs associated with those services. Consumers browse GMD to find services that meet their requirements. GMD is built over standard Web service technologies such as SOAP and XML. Therefore, it can be queried by other programs. To provide with an additional layer of transparency, a client API has been provided that could be used by programs to query the GMD without the developers having to concern themselves with SOAP details. The Gridbus scheduler interacts with the GMD to discover the testbed resources and their high-level attributes such as access price.

- **GridBank:** This is a secure Grid-wide accounting and (micro) payment handling system. It maintains the users (consumers and providers) accounts and resource usage records in a database. GridBank supports protocols that enable its interaction with the resource brokers of Grid Service Consumers (GSCs) and the resource traders of Grid Service Providers (GSPs). It has been primarily designed to provide services for enabling a Grid computing economy; however, it can be used in e-commerce applications as well. The GridBank services can be used in both co-operative and competitive distributed computing environments.

- **Gridscape:** It is a tool that enables the rapid creation of interactive and dynamic testbed portals without any programming effort. Gridscape primarily aims to provide a solution for those users who need to be able to create a grid testbed portal but do not necessarily have the time or resources to build a system of their own from scratch.

- **Alchemi:** It is a .NET-based grid computing framework that provides the runtime machinery and programming environment required to construct desktop grids and develop grid applications. It allows flexible application composition by supporting an object-oriented grid application programming model in addition to a grid job model. Cross-platform support is provided via a web services interface and a flexible execution model that

supports dedicated and non-dedicated (voluntary) execution by grid nodes. Because grid computing software has been primarily written for Unix-class operating systems, Alchemi will enable the effective utilization of the computing resources of the vast majority of desktop computers i.e. those running variants of the Microsoft Windows operating system.

- **Libra:** It is a cluster scheduling system that guarantees a certain share of the system resources to a user job such that the job is completed by the deadline specified by the user provided he has the requisite budget for it. Jobs whose output is required immediately require a higher budget than those with a more relaxed deadline. Thus, Libra delivers utility value to the cluster users and increases their satisfaction by creating realistic expectations for the job turnaround times.

- **GridSim:** This toolkit provides facilities for the modeling and simulation of resources and network connectivity with different capabilities, configurations and domains. It supports primitives for application composition, information services for resource discovery and interfaces for assigning application tasks to resources and managing their execution. It also provides a visual modeler interface for creating users and resources. These features can be used to simulate parallel and distributed scheduling systems such

as resource brokers or Grid schedulers for evaluating performance of scheduling algorithms or heuristics.

2.5 Conclusion

As described in this chapter, the need for Grid computing infrastructures is becoming one of the main research areas in the field of parallel and distributed computing. Different Grid projects adopted different approaches to achieve their ultimate goal of creating a Grid infrastructure. Their implementations varied to span a wide variety including toolkits providing a bag of services that the user can select from, object oriented approaches to Grid design, Java based Grids for portability, and computational economy Grids.

In spite of the variance between the implementations of these projects, they share the same characteristics and requirements of any Grid environment as presented in the previous chapter. Also all the analyzed projects included security as a core component of its architecture. These projects concentrated, at this early stage of Grid evolution, on basic security requirements and ignored the importance of intrusion detection as a second layer of security. This fact motivated us to design and implement an intrusion detection architecture for Grid security.

Chapter 3

The Grid Security Infrastructure

Chapter 3: The Grid Security Infrastructure

This chapter presents and analyzes the Grid security infrastructure that is implemented in various Grid environments. Then it presents a research project implemented using one of the Grid environments that uses the presented security infrastructure. This research project helped in better understanding of the Grid environment and security related issues.

3.1 Security Infrastructure

Security is important to all computer systems to enable administration and policy enforcement. These policies control the users of the system by specifying which user can access the system, what operation is allowed by each user, and protects the system of being compromised or misused. This is mapped to the basic security requirements of any system which are authentication, authorization (access control), integrity, privacy, and non repudiation [103]. Implementing security mechanisms in the Grid is important to protect the large number of resources and users. The problem in implementing such security mechanisms in the Grid is that grid application may require access to multiple computational and data sources that may be geographically distributed and administrated locally and independently. Grid applications may involve hundreds of processes running on different resources and need

to authenticate and communicate with each other securely. These processes and resources may also join or leave dynamically which makes it impossible to initialize the security relationship between them at the application startup.

All these problems complicate the implementation of a Grid security system and add new security problems not addresses by existing distributed security mechanisms such as Kerberos and the secure shell [73]. Also the special Grid characteristics and the different application scenarios have added – in addition to the normal security requirements – special requirements including [37]:

- **Single sign-on:** Grid applications may take days and require accessing and authentication with multiple resources. The user must only be required to authenticate once at the beginning of the application and then the application should continue acquiring and releasing resources seamlessly without further user intervention.

- **Protection of credentials:** A user credential is a piece of information – such as passwords or private keys – that is used to prove his identity during the authentication process. It is required that the user credential be protected to ensure security.

- **Interoperability with local security solutions:** Grid resources are administrated locally and independently, so it

is impossible to require the replacement of existing security mechanisms at each site. The Grid security provides only interdomain security and must cooperate rather than replace existing security mechanisms. The access to a resource will be determined by its local security policy and mechanisms.

- **Exportability:** Laws that govern the export of encryption technologies are complex, dynamic, and varies from country to country. Because the Grid may span multiple countries, security mechanisms are required to be exportable by avoiding the need of using bulk encryption.

- **Uniform credentials/certification infrastructure:** A common way for expressing identity is required for interdomain access. So it is recommended to use standards to encode the credentials of users and resources.

- **Support for secure group communication:** A Grid application may consist of many processes working together to solve a problem. These processes need to communicate and coordinate with each other as a group in a secure way not simply as in client server applications.

- **Support for multiple implementations:** The Grid security policy should not require special implementation technology. It should be possible to implement it with various security technologies.

A Grid Security Infrastructure (GSI) was first introduced by foster, et.al. in 2001 [37] it addressed the special grid requirements. This architecture is shown in *Figure 3.1* and briefly described below showing how it satisfies the requirements of the Grid.

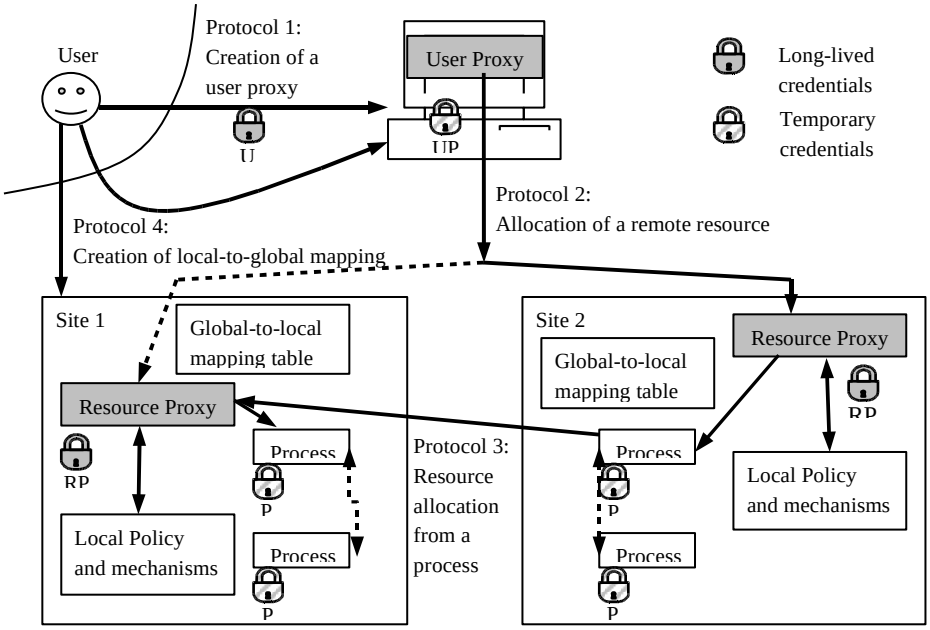

Figure 3.1: A typical Grid security architecture [37].

The GSI concentrates on two main issues. The first one is to provide an authentication mechanism between users, user processes, and resources. This authentication will enable different local security policies to be integrated into a single global framework. The second issue is to enable the application of local access control mechanisms without

changing them. The infrastructure consists of four protocols, the User Proxy (UP), and the Resource Proxy (RP).

The UP is a process that acts on behalf of the user for a limited time to manage a computation session. The purpose of the UP is to enable single sign-on. The UP is created first through protocol 1 as shown in *Figure 3.1*. The user first gains access to the machine on which the UP will be created using local access control mechanisms. Then the user uses his credential ($\mathbf{C_U}$) to create temporary credential ($\mathbf{C_{UP}}$). Finally the UP process is created and given its temporary credentials ($\mathbf{C_{UP}}$) that will be used for further authentications and acting on behalf of the user.

The temporary credentials consist of a tuple – containing various information such as the valid interval of this credential, authorized actions, user ID, and so on – signed by the user credentials. It was possible to give the user credentials to the UP to enable single sign-on which is very simple solution. Alternatively, temporary credentials were used for two reasons:

- To protect the user credentials. Because giving the user credentials to the UP and then to processes acting on behalf of the user increases the probability of user credentials being hacked and misused.

- To control the UP and the user processes by

delegating a subset of the user rights – the signed tuple – to these processes through the temporary credentials and so reducing security risks.

After the UP is created, the user can leave the computation – indicated by the curved line in *Figure 3.1* separating the user from the rest of the GSI – leaving the UP handle the required operation on the user behalf.

Interoperability with local security solutions is achieved through the RP that acts as an agent translating between interdomain security operations and local intradomain mechanisms. The UP contacts the RP through protocol 2 as shown in *Figure 3.1* to allocate the resource. First the UP and the RP authenticate with each other (mutual authentication). Then the UP sends a signed request to the RP. The RP checks if this user is allowed – according to local security mechanisms – to access the resource. If the user is authorized the user and resource proxies negotiate together to create the process(es) temporary credential ($\mathbf{C_P}$). Finally the RP allocates the resource and passes ($\mathbf{C_P}$) to the newly created process(es).

The process credential ($\mathbf{C_P}$) facilitates the secure group communication by enabling the authentication between the UP and the processes and between the processes themselves if they exist in different domains. It also enables a process to acquire more resources by using protocol 3, as shown in

Figure 3.1, in which it first mutually authenticate with the UP. Then it passes a signed request to the UP that – after accepting the request – allocates the resource through Protocol 2. This technique may not be scalable because of the dependence on a single UP but the advantage is its simplicity and fine grained control over resource allocation by the processes.

Each user in the Grid has a unique global name and a permanent credential that verify this identity. However, in most cases the same user will be known by a different local name at each administrative domain. The RP maps this global name to a local name which is known by the local security mechanisms which translates between interdomain and intradomain security mechanisms. This mapping is simply a table containing the global name and the local name of the user. This table may be manually created and filled by the local administrator of the domain or filled automatically through Protocol 4 as shown in *Figure 3.1*. In this protocol the UP mutually authenticate with the RP and then send to it the local and global name. Then the user must login to the resource using its local mechanisms and starts a map registration process which will also pass the global and local names to the RP. Finally the RP validates the passed values, and if they match, it adds the global and local names to the map table.

This infrastructure relies on authentication and signature

verification techniques and does not require encryption. It does not dependent on any specific security mechanisms and can be implemented by multiple techniques. Standards such as X.509v3 may be used to encode credentials to support the uniform credentials/certification infrastructure requirement.

3.2 Evaluation of the Current Grid Security Infrastructure

One of the currently existing Grid Security Infrastructures was presented in previous s*ection*. The current implementation of this infrastructure in the Globus ToolKit [93] is based on the Public Key Infrastructure (PKI) [60]. Although it is an open architecture and could have been implemented using other approaches such as plain text passwords or Kerberos [8] among others [37]. This infrastructure focuses on providing an authentication service for the Grid and allowing local access control mechanisms at each site to be applied without changing them.

The example in *Figure 3.2* shows the steps of running a typical Grid application focusing on the interactions of the Grid Security Infrastructure (GSI) with the user and his/her application. This example will give better understanding of the underlying GSI in action, which on top of it the Grid Intrusion Detection Architecture (GIDA) was designed.

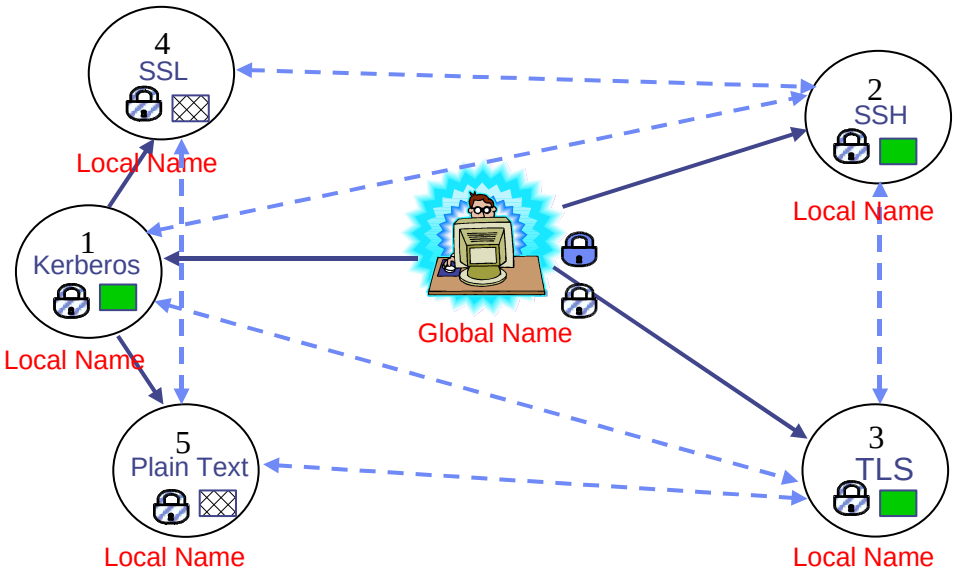

Figure 3.2: Grid applications, a typical example.

In this example the user – in the middle of the figure – tries to execute an application on the available suitable resources. Each group of resources – clusters, databases, super computers ... – that are under the same local administration and control, called an ***administrative domain,*** is represented by a circle. Each administrative domain has its own security policy that is enforced using any locally selected security mechanism, e.g., Kerberos, SSH, or plain text passwords, as shown inside the circles. This local security mechanism gives the administrative domain a completely independent security mechanism of other domains. This local control over resources allows only authorized persons to use them. In this example we assume that the user is authorized to use all the resources.

The user must first find and allocate suitable resources for his/her application among these authorized resources, either through a broker or a resource manager. To allocate a resource the user must have a local account at each administrative domain. This account may be created specifically for a particular user, a group account, or left anonymous for public use. Providing single-sign-on, which is one of the requirements of the GSI, is important so the user will not have to login at each domain separately using different mechanisms. Single-sign-on requires each user to have a *Global Name* that will be used in identifying that user at all domains and in each Grid operations. This global name is proved – at authentication process – by using the user credentials which is usually a certificate signed by a trusted certificate authority and a private key. These credentials are shown in the figure by the dark lock.

After authentication at an administrative domain using GSI mechanisms the global name is mapped to a *Local Name* – the name used locally to identify the user – then according to the local security policy the user is either authorized or denied to access the resource. It can be noticed that using this approach each Grid user has a unique name that identifies all his operations, and even if the local account is a group account or anonymous the logging at the Grid operations level is still possible using the users global names.

Now, after the user selects some suitable resources, the user can allocate these required resources – in this example these suitable resources where at domains labeled 1, 2, and 3 and allocation process is shown by the solid arrows – and run his application processes that are represented by dark rectangles inside the domains. These processes are solving the same problem in parallel so in most cases they need to communicate with each other – the dashed arrows – to cooperate and share results. This implies that they need to authenticate with each other somehow. The processes may need, while execution, to allocate more resources as in the case with the process at domain 1 need to allocate resources at domain 4 and 5, and then run other processes – shown by dashed rectangles – at these new domains to complete the application.

One trivial solution to allow these processes to authenticate, allocate resources, and perform any other tasks on behalf of the user is to give them the user's credentials. Although this solution is simple, it has two main drawbacks. First there will not be any control over the process. It can do whatever the user can do because it has a copy of the user’s private key. And the user can not control and limit what the application is allowed to do, because in a security context this control is enforced through the credentials. Second and more importantly, the process normally runs on remote resource, this means that a copy of the private key will exist at remote

sites. Sending the user credentials with each process will increase the probability of stealing and misuse these credentials because they will not be well protected. Misusing the credentials means misusing the grid resources! So to protect the grid resources the user credentials must be well protected. These are tow major security risks that makes this solution not applicable and introduced instead the concept of delegation.

Delegation allows the user to delegate or pass some of his/her rights to the processes allowing them to act on his/her behalf. This is done by creating temporary credentials – represented by the shaded locks – that allows the process to perform specific tasks for a limited period of time. The user can place any restrictions he/she wants on these temporary credentials, such as the period of time that they are valid, the sites that the application can communicate with and so on. This approach gives control upon the processes and reduces the risks of the user's credentials being compromised. Now the user can leave after he/she has login using his/her credentials and created the temporary credentials, and then come back later to check his/her application and collect results.

The objective of the intruder is either to gain access to a system (Authentication) or to increase the range of privileges accessible on a system (Authorization) [103]. This requires an outsider intruder to acquire information that should have been

protected. In this case this information is in the form of the user's credentials or private key. If the intruder managed to get this private key some way then he can login the system and perform all operations that the legitimate user had privileges to do.

Users' private keys can be protected in one of two ways [45]:

- **Access Control:** The private key is kept in a file accessible only by the private key owner account. For increased protection this file may also be encrypted using a password.

- **Smart Cards:** The private key may be stored on a smart card. This provides the best security but requires special hardware that is currently not widely used.

The Grid Security Infrastructure (GSI) assumes that no one - other than the owner of the private key - can gain access to the file containing the private key. Although the previous two ways are considered secure enough to protect the private key of the user, there still a small probability that flaws can occur and an intruder can gain access to this file. Techniques used to crack password protected systems such as trying default passwords or exhaustively trying all short passwords is not valid in GSI because with current technology it is not feasible to guess the user private key. The most important

requirement in the GSI as presented above is the single-sign-on and delegation. This is important because the user application may require the use of hundreds of resources located at different domains, and requiring the user to manually sign on each of these domains is impractical. This is also important because that the user's application may run for days or weeks, and we may not expect the user to sit in front of the computer and manually log on whenever a new resource is needed. The temporary credentials enhanced the security. Although if an intruder gain access to this temporary credential, temporary private key, is less dangerous than gaining access to the user's private key, an intruder is still able to do harmful or unauthorized work using this temporary private key.

Another risk can come from insiders. These are the legitimate users that misuse their privileges. Protection from insiders is more difficult because they can not be prevented using security mechanisms.

3.3 The Grid Research Project

The emerging Grid technologies have changed the way people think about computation by presenting new paradigms and application models. To help us better understand the Grid environment, we decided to join in a research project. The goal of joining such project is to better understand the Grid

infrastructure, the nature of Grid applications and environment, and specially to tackle the Grid security.

This work is a part of a joint research project between Ain Shams University (ASU) in Egypt and George Washington University (GWU) in USA to build a system for signature verification. The target of the project is to create a hand written signature verification system. This means that the user of the system will be able to check if a hand written signature he/she have (on a contract for example) is a genuine or forged signature.

This work presents a new approach to solve such problems using Grid approaches to increase performance and resource utilization while reducing the maintenance costs and security risks of the system. This is done by proposing a general architecture of the system, focusing on the advantages of using the Grid technologies over other techniques. The limitations of other possible solutions and the advantages of Grid solutions have make it a good paradigm for future applications. A testbed linking the two universities was created and used to test the proposed architecture and prove its applicability in real applications.

Section 3.3.1 briefly describes the problem of hand written signature verification. A proposal of different approaches to solve this problem is presented in section 3.3.2 showing the advantages and drawbacks of each and why the

grid approach is better. Section 3.3.3 discusses the testbed used for testing and the results of different experiments. Finally the general conclusions are presented in section 3.3.4.

3.3.1 The Signature Verification Problem

The problem of handwritten signature verification has four main components that can be described in the following points:

1. **The Database:** There are databases storing images of genuine signatures. The system can only verify signatures for persons having their genuine signatures stored in these databases. For each person there is one or more signatures set. Each set consist of a number of genuine signature images of that person. Each set also has some properties – meta data – describing its contents such as the number of signature images, the date these signatures where signed by the person, the conditions at which the person signed these signatures, the signature image resolution, and so on.

2. **Suspected Signature:** The user of the system should have a person's signature image that is needed to be verified by comparing it with the genuine signature images in one of that person's signatures sets.

3. **The Analysis Algorithm:** The system can accept one or more algorithm capable of analyzing suspected signature images and decide whether they are forged or not. These algorithms may have different characteristics and requirements such as complexity, execution time, accuracy, signature image format, number of valid signature images needed, and so on.

4. **The User's QoS:** According to the user's desired Quality of Service (QoS), the suspected person signature will be compared to one or more of his signature sets in the database – with different attributes and quality – by one or more analysis algorithm – with different requirements and accuracies – to achieve the desired user's goals.

3.3.2 The Signature Verification System Architecture

Although the problem sounds simple, it contain many complex hidden issues and trade-offs and many possible solutions. But generally these solutions can be classified into one of the possible three scenarios.

The Old Scenario

This is the simplest – but not the best – solution to this problem. It is simply to have the whole signature verification

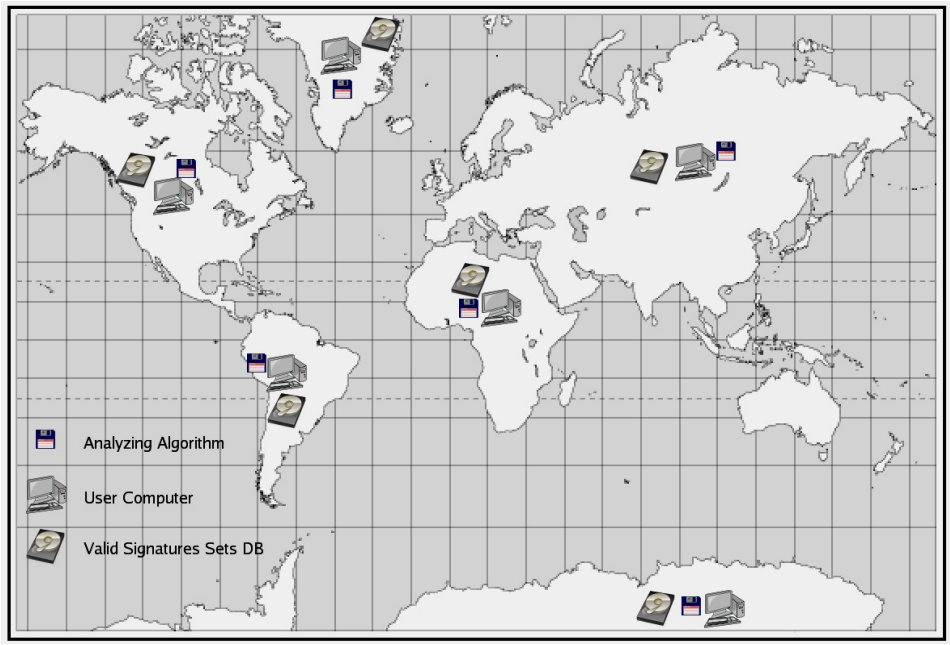

Figure 3.3: The old scenario.

system as one entity at each user that wants to use it as shown in *Figure 3.3*. Going back in time a couple of decades at the beginning of the computer revolution maybe it would the only possible solution. In this scenario any user of the system must have a database management system with a database filled with genuine signature sets for all possible persons that the user of the system may need to verify. The user must also have all the signature analyzing algorithms needed to analyze the signatures according to his QoS. A computer system capable of providing the needed computational power whenever it is needed by the algorithms to analyze the signatures must be available and dedicated for this purpose. This system must also

be regularly administrated and maintained by updating the database with new signature sets, checking for new algorithms and updated versions, and maintaining and upgrading the computer system hardware.

This scenario with its architecture and requirements introduced many problems. The database size may grow to be very large and hard to maintain by the user of the system when it is filled with millions of person signature sets. Such system – with replicating its components at each user – will have a very high maintenance cost and poor cost/performance ratio. This scenario contains security problems. The user of the system must implement security mechanisms to protect this database and to verify that its content was not compromised by any undetected attack on the system. The user must also verify the origin of the signature sets and the algorithms to assure that they are correct. Most importantly it is unsafe and very risky and to give a database with all genuine persons signatures to all users of the system, because there is no way to guarantee that none of these users will miss use this valuable information to create professional forged signatures that are hard to detect.

The Modern Scenario

With the widespread of the Internet and web enabled applications this problem may be solved in much elegant

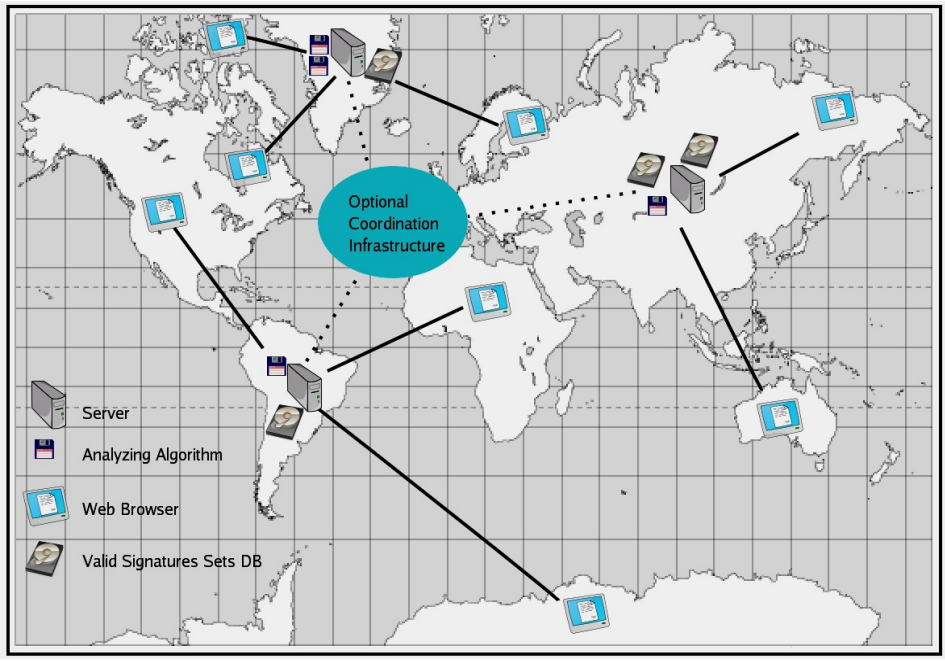

Figure 3.4: The modern scenario.

ways. Modern technologies may be used such as web services and client/server application. In one of the possible scenarios, as shown in *Figure 3.4*, the user acquires appropriate credentials(private keys, password, ...) to verify his identity. After that, using a web browser, the user browse to one of the system's home page then authenticate and login using his credentials. The user uploads the signature image that needs to be verified. The server analyzes the signature and sends back the result to the user.

In this scenario all what the user should have is a valid credential to prove his identity, a web browser, and of course

the signature image he wants to verify. This scenario sounds quite simple but only from the user (client) side. But from the server side there are several solutions with their problems. It could be just having multiple servers (mirrors) distributed around the world to divide the workload among them. Each server will maintain a complete system with a database of all valid persons signature sets, all analyzing algorithms, and the necessary computational power to handle the users requests. This is simple but has almost the same drawbacks of the old scenario.

An alternative solution for the server side is to have a real distributed system with a distributed database of valid signature sets, for example a database for each country, and having multiple processing nodes each specialized in one or more analyzing algorithms. This system will solve almost all the problems. Each valid signature set is maintained at a single database which eases the maintenance, reduces security risks, and reduces the database size. The cost is distributed among the database and computation servers. Each site will be responsible only for its part of the system not the whole system.

But on the other hand this architecture will require the implementation of a specialized infrastructure (shown by the circle in the middle of *Figure 3.4*) that is among others capable of:

1. Locating the available signature sets databases.

2. Searching these databases for a person's valid signatures sets.

3. Locating free analyzing nodes with appropriate algorithms.

4. Handling possible failure of any of these system components.

5. Implement appropriate security mechanisms to control access to databases and analyzing nodes.

6. Provide a mechanism to allocate and start computation on the analyzing nodes.

Such architecture is very complex and hard to be specially implemented for a specific application of the hand written signature verification system.

The Grid and Security

The Grid – from its definition – is used to coordinate and couple resources that are geographically distributed and administrated independently to create virtual organization that enables collaboration and seamless access to computational resources. Put in another way, the Grid simply implements the entire required infrastructure discussed in the previous

scenario (shown by the circle in the middle of *Figure 3.4*). The Globus ToolKit 3 [35][93] is one of the software toolkits that implements the Grid infrastructure. It provides basic services needed by any Grid enabled application. These services, as presented in Chapter 1, are resource management [50], information services [51], data management [98], and security[37] as shown in *Figure 1.2*. Security is important for the success of a Grid environment. The Globus ToolKit addressed the security issue through the Grid Security Infrastructure (GSI) that is a component of the toolkit and supports all other services as in *Figure 1.2*, It is currently based on public key infrastructure (PKI) and it requires all users and resources to have appropriate certificates to join the Grid environment. GSI also provide basic services such as encryption, single sign on, mutual authentication, among others.

The Grid Scenario

Using the Globus ToolKit 3 the scenario – as shown in *Figure 3.5* – will start by the user login using appropriate credential and creates the user proxy. The user will start the Globus enabled signature verification application and provide it with the signature image needed to be verified. The program will contact the monitoring and discovery service (MDS) – a part of the Grid information services – to find available valid signature sets and processing nodes with appropriate analyzing

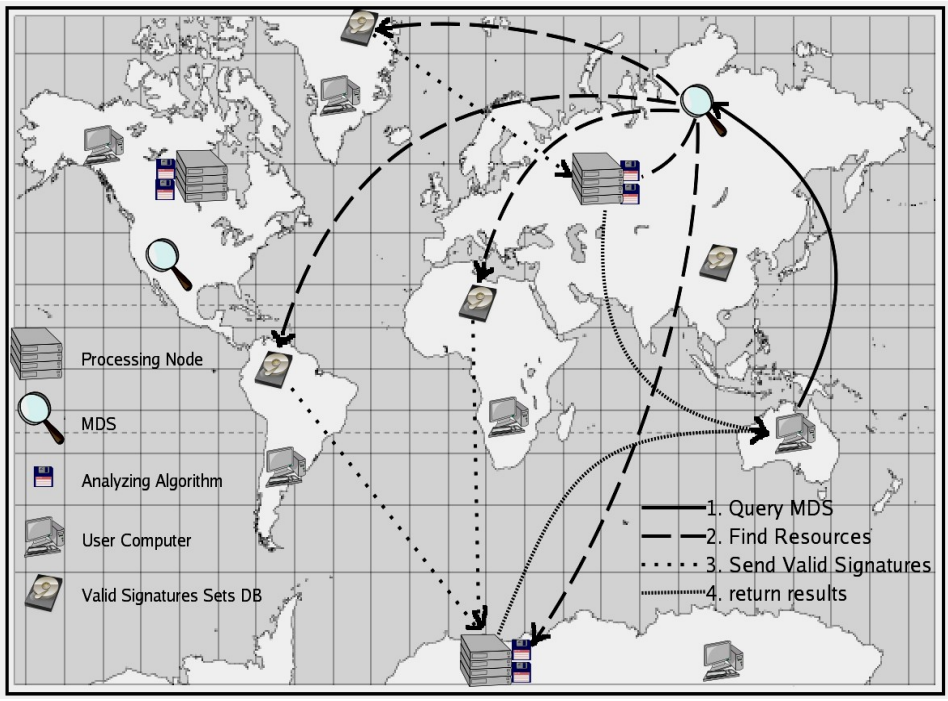

Figure 3.5: The Grid Scenario.

algorithms. According to the user QoS the application will pick one or more available processing node(s) and then send to them suitable valid signatures set and the suspected signature image. After processing ends the results are sent back to the user application and displayed.

In this scenario the user will have valid credentials to prove his identity, the signature image, the Globus infrastructure installed, and the Signature verification application. In this scenario we can notice the following:

1. It is a dynamic environment where valid

signatures sets can be added or removed and processing nodes can join or leave as they want. This environment is shared and coordinated through the Globus infrastructure.

2. The valid signature sets are protected by being kept at secured databases and only sent to trusted and authenticated analyzing nodes based on the choice of the local system administrators of the signature database.

3. The coordination process is seamless and all complexities are hidden by the Globus infrastructure. These complexities are in security, resource management, data management, and information services.

4. Signature algorithms are updated and improved locally and then discovered globally by MDS without requiring changing of the user application.

5. Within this large pool of resources the user can deliver the desired QoS by combining the appropriate pair of signature set and analyzing algorithm.

6. User authentication and secure signatures transfer over public insecure networks are done using the Grid Security Infrastructure GSI [37]

7. There can be different trust relationships among users, databases and processing nodes owners. The Grid Security Infrastructure supports these trust relationships. Although it is a one global system, each component (user, database, or processing node) works only with trusted components. This trust relation is defined locally by the decisions of the component's system administrators. For example a database may deny the request to send signatures sets to a processing node if it is not trusted. In the Globus ToolKit 3 this is done now using Public Key Infrastructure and hierarchical Certificate Authorities

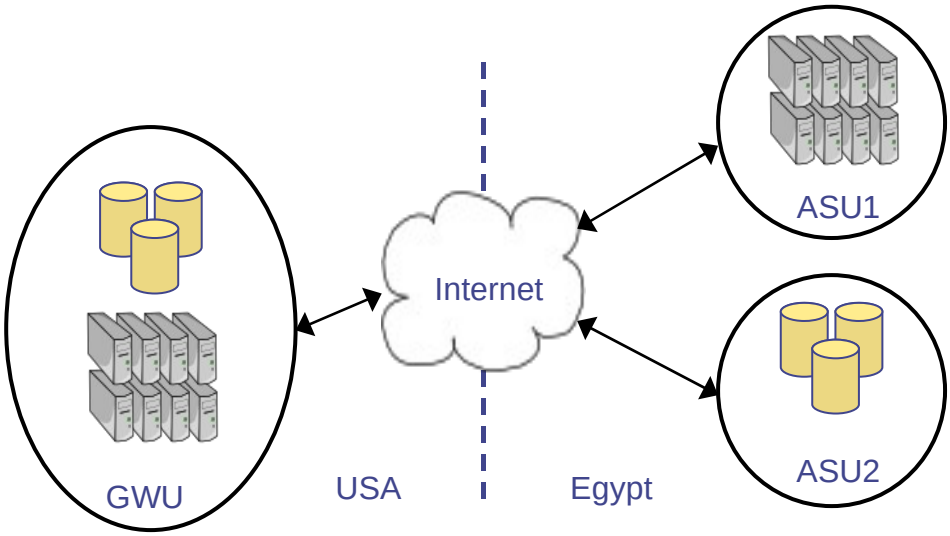

Figure 3.6: Layout of available resources used in implementation.

3.3.3 Project Results

To test the proposed Grid scenario a testbed was created consisting of three nodes as in *Figure 3.6*. The first one is called GWU and is located in USA at George Washington University. The other two nodes are called ASU1 and ASU2 and are located in Egypt at Ain Shams University. GWU contains both a signatures database and an eight node cluster for distributed signature analysis. ASU1 consists on only of an eight node cluster for distributed signature analysis. ASU2 contains only a signatures database. The three nodes where linked using the Globus ToolKit 3. The distributed hand written signature verification algorithm [63] is installed on the clusters at GWU and ASU1. A grid enabled application [65] was developed that enables the users of the system to locate available processing nodes with appropriate algorithms, locate and search the databases for a specific person's signatures sets, select the best signatures set and processing node for the user, upload the suspected signature, and finally return the result of analysis back to the user. The system user can be located anywhere in the world, all what is needed it appropriate infrastructure installed, the grid enabled application, and appropriate credentials to use the system.

The system total execution time was measured with respect to three parameters: The location of the user, the location of the signatures database node, and the location of

the analysis node, each of which can be – in this testbed – either in Egypt or in USA. According to the user QoS, the processing is done on the nearest available analyzing node to the signatures database node. The following scenarios can occur independently of the user's location:

1. Signatures are available on GWU and GWU is chosen for analysis.

2. Signatures are available on GWU and ASU1 is chosen for analysis.

3. Signatures are available on ASU2 and ASU1 is chosen for analysis.

4. Signatures are available on ASU2 and GWU is chosen for analysis.

EGYPT		***Database***		
		ASU1	***ASU2***	***GWU***
Processing	***ASU1***	X	0.27	1.27
	ASU2	X	X	X
	GWU	X	1.31	0.51

Table 3.1: Execution time of experiments (in minutes) when the user in Egypt.

USA		*Database*		
		ASU1	***ASU2***	***GWU***
Processing	***ASU1***	X	0.45	1.26
	ASU2	X	X	X
	GWU	X	1.14	0.25

Table 3.2: Execution time of experiments (in minutes) when the user in USA.

Table 3.1 and Table 3.2 summarize the results obtained from running the grid application, the first when the user is in Egypt, and the second when the user is in USA when the analysis is completed without any errors. The columns represent the node on which the signatures set was found and the rows represent the node on which processing was done. A cell with value X means that this case is not applicable due to the assumptions that the signatures do not lie on all nodes and that not all nodes can do analysis. The average execution time when the user was in Egypt was found to be 0.84 minutes and that when the user is in USA was found to be 0.775 minutes. This difference in the averages is attributed to the connection speed in Egypt site, as it is based on an ADSL line with upload to download ratio is 1:4.

Executing the same algorithm on a sequential machine

took about 34 seconds. The Grid enables the access to more powerful resources – the cluster in this case – to perform computations. From this point of view it is fare to compare the Grid enabled system performance that took 49 seconds in average to the sequential single machine system performance. It can be noticed that the delays due to communication was recovered by faster computation resulting in approximately the same performance.

3.3.4 Project Conclusions

The Grid infrastructure provided a seamless way to efficiently link and access different distributed computational resources. Providing an ideal development environment to create innovative applications such as the hand written signature verification system presented here. It enabled the efficient use of the available resources – databases, computations, and algorithms – by an intelligent Grid enabled application that used the underlying infrastructure to efficiently satisfy the desired user's QoS. The system was also secure by keeping the genuine signatures sets at secure databases away from the users and by using the Grid Security Infrastructure for authenticating users and transferring signature images. The different tests performed on the system showed good performance compared with single machine systems proving the applicability and advantages of using Grid paradigms for future applications.

Chapter 4

Intrusion Detection

Chapter 4: Intrusion Detection

This chapter reviews the field of intrusion detection by defining it and showing why it is needed. Then it illustrates the different techniques and approaches used to detect intruders.

4.1 Introduction

Early efforts in the area of networking and the Internet concentrated in developing efficient protocols, algorithms and hardware necessary to link computers with each other. They did not concern with security issues. After the rapid increase of the Internet size, security became a major issue to protect valuable information and resources. So passwords, encryption and other security techniques arose.

The new emerging security techniques came with three major drawbacks, witch created the need for intrusion detection. First drawback is that increasing the security level places constraints on communication and slows it down and decreases its usability, on the other hand decreasing the security level to make the communication more flexible increases the opportunity for penetration and unauthorized use of the linked computers. Second is that security systems may contain software bugs and holes that attackers can use to break the system, and no one can guarantee that a security system is one hundred percent free of bugs and holes. Third and most

important is that even if the security level was increased to its maximum levels and all bugs and holes were fixed, traditional security systems, some times called first layer of security, can not protect a system from insiders. An insider is defined as any legitimate person who is allowed, according to a security policy, to use the system.

Today's computer systems are vulnerable to both abuse by insiders and penetration by outsiders. Current security systems are not enough to protect systems from insiders and outsiders [103]. Passwords can be cracked, keys can be stolen, firewalls does not protect the system from insiders and outsiders can dig under the firewall, security systems contains bugs and holes that are impossible to fix in a feasible way were legitimate users can miss use their authority and privileges. Thus intrusion detection is viewed as a second line of defense as shown in *Figure 4.1*. Intrusion detection systems

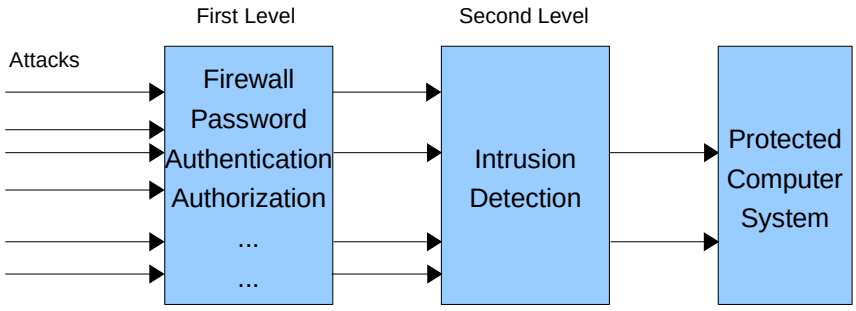

Figure 4.1: Different security levels to protect a computer system

are based on the assumption that the normal use of the system is different from malicious use [68]. They mainly analyze the auditing file and try to discover any suspicious user behavior, and take appropriate actions if such user is discovered.

To put it in another way computer security must address three fundamental needs:

- Prevention
- Detection
- Response

In general it is always better to prevent something bad from happening. This is what the first layer of security tries to achieve through fire walls, passwords, keys, encryption, and so on. If prevention is achieved then there is no need for detection and response. Unfortunately this is not the case for the reasons stated before, and it is impossible to achieve 100% prevention with the current technology. That is why detection and response is a vital part of any security system.

The objective of the intruder is to gain access to a system (Authentication) or to increase the range of privileges accessible on a system (Authorization) [103]. This requires the intruder to acquire information – keys, passwords, etc. – that should have been protected.

Intrusion detection may be defined as “the problem of identifying individuals who are using a computer system without authorization (i.e., 'crackers') and those who have legitimate access to the system but are abusing their privileges (i.e., the 'insider threat')” [55]. Another definition is “the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or bypass the security mechanisms of a computer or network” [71]. The research field of intrusion detection was first formalized with the publication of Anderson's seminal report in 1980 [68]. After that the first comprehensive model of an intrusion detection system was

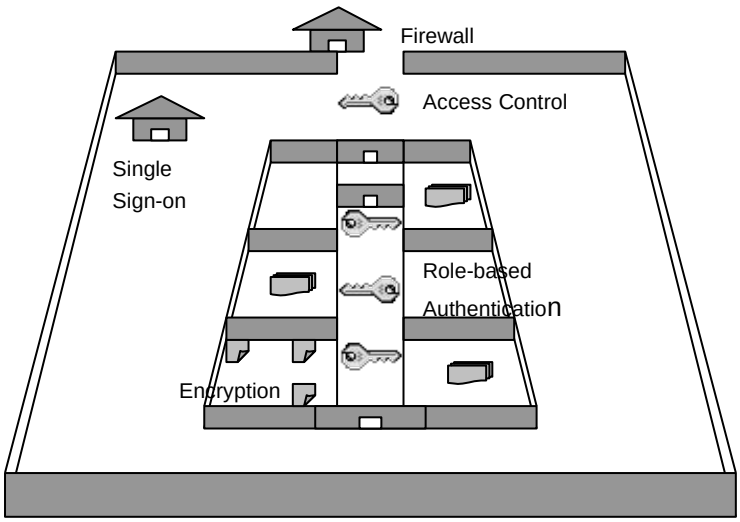

Figure 4.2: Access control mechanisms

introduced in 1987 by Denning [17].

To help in better understanding of what is intrusion detection all about, the two illustrations, shown in *Figure 4.2* and *Figure 4.3* form [70], compares computer security with all its layers with security mechanisms used in daily life to protect precious things in a building.

The first layer of computer security is basically access control mechanisms. They are analogous to traditional access control mechanisms as shown in *Figure 4.2*. The fence around the building is just like a firewall allowing only few to pass

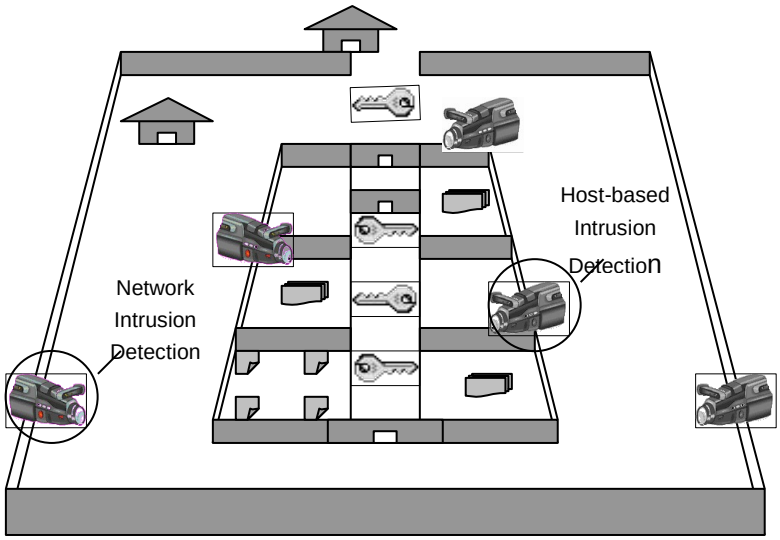

Figure 4.3: Video cameras, by analogy, typical to intrusion detection role in computer systems.

through. Gates with locks on them are just like passwords. Inside the building closed rooms are like restricted areas that only few people who have the right privileges can access.

Thieves can climb the fence, steal keys, break locks, or even masquerade as a person who is allowed to enter the building. Authorized persons can even misuse their privileges to do something wrong. That why, as shown in *Figure 4.3* cameras are needed as a second layer to increase security. They are used to watch what is going on outside and inside the building to ensure that everything is OK. That is exactly analogous to the rule of intrusion detection in a computer system where the building is the protected system. As analyzed below, cameras outside the building are network based intrusion detection, while inside cameras are host based intrusion detection.

4.2 The Anatomy of Intrusion Detection Systems

Intrusion detection systems try to detect, using different mechanisms and approaches, the difference in behavior caused by an intruder and take appropriate action to stop the intruder. To achieve this most intrusion detections employ a common architecture consisting of three major modules (gray boxes in *Figure 4.4*):

- **A data gathering module (Audit Collection):** This module gathers data that may contain evidence of intrusion

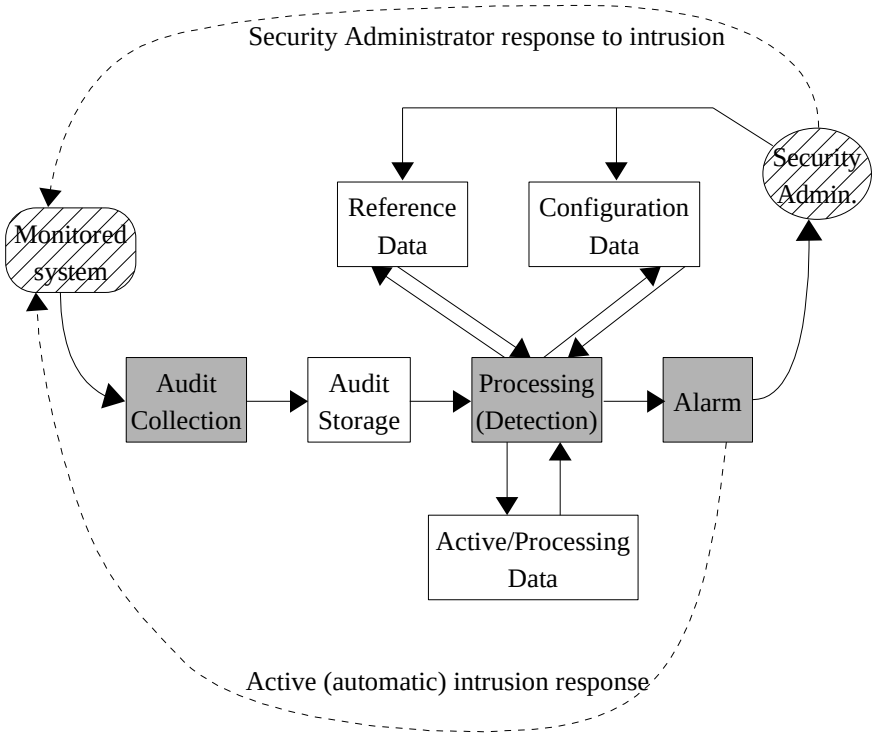

Figure 4.4: Organization of a generalized intrusion detection system

to be used in intrusion detection decisions. There are many sources of data that may be collected such as network activities, host security logs, keyboard input, command based logs, or application based logs.

- **An analysis module (Processing):** This module processes the gathered data to identify intrusive activity. There are many approaches to analyze the gathered data.

Earlier approaches were mainly based on the application of statistical methods to identify anomalous activity [69]. Many early systems [27][30][52][15] employed this method. Modern approaches uses a diverse range of classification methods to identify anomalous activities including, among others, rule induction [29][66][22], artificial neural networks [80][47][9], fuzzy set theory [92], classical machine learning algorithms [101][100], artificial immune systems [83][82], signal processing methods [61], and temporal sequence learning [91][2].

- **A response module (Alarm):** This module handles all the system alarms and takes decisions and performs appropriate actions according to the results of the analysis. This response may be automatic or just a notification to the security administrator to help in taking right actions in stopping the intrusion risk.

A general intrusion detection model should, in addition to the previous components, contain the following components [26] (white boxes in *Figure 4.4)*:

- **Audit Storage:** An important issue in any intrusion detection system is feature selection and data reduction. Feature selection is important because the inclusion of too much data will adversely impact the performance of the system, while the inclusion of too little data will reduce the

overall effectiveness of the system. Audit data must be stored somewhere so it is important to reduce the size of the data to save storage space. This storage may be for long time, months, or temporarily awaiting for processing.

- **Configuration Data:** This data sets the behavior of the intrusion detection system. It is configured by the security administrator to control and tune the operation of the system. This data may contain information such as when, where, and how to collect audit data and also how to respond to intrusion among others.

- **Reference Data:** To detect deviation from normal behavior, the analysis module must compare the gathered audit data to some reference data to detect intrusion. This reference data may be in the form of signatures of well known attack and/or profiles of normal behavior depending on the type of intrusion detection system as presented below. Profiles are usually updated by the analyzing module on regular basis when new observed information about normal behavior is gained. Signatures are usually updated by the security administrator when new attacks are discovered and their signatures detected.

- **Active/Processing data:** The analyzing module usually needs to store temporary results needed to complete the detection process. Examples of this data may include

information about partially fulfilled intrusion signatures or the percentage of matching between detected behavior and normal profiles at different time periods.

Intrusion detection systems implement the above components in different ways and using a wide range of approaches and techniques aiming to select the best combinations to improve the system performance. This wide variety led to different approaches to classify intrusion detection systems. The most significant and used approach is to classify Intrusion detection techniques either according to the source of the data used for the analysis into network based and host based intrusion detection systems [21][39], or according to the approach taken to analyze the data into misuse detection and anomaly detection [16].

4.3 Network vs. Host Based Intrusion Detection

The first step in any intrusion detection system is to collect data about the protected system and the users using it. This data reflects the status of the system and the operations that the users are performing. Depending on the way the data gathering module gets its data, the intrusion detection systems are classified to either network based intrusion detection or host based intrusion detection.

Network based intrusion detection systems get their data by installing a device on the network capable of monitoring all

network traffic and passed packets. They rely on raw network packets in their analysis. On the other hand, host based intrusion detection systems use log files created on each host, containing all the operations performed on the host, as the data source.

A Network Intrusion Detection System (NIDS) monitors the packets that traverse a given network link. Such a system operates by placing the network interface into promiscuous mode, affording it the advantage of being able to monitor an entire network while not divulging its existence to potential attackers. Because the packets that a NIDS is monitoring are not actually addressed to the host the NIDS resides on, the system is also impervious to an entire class of attacks such as the "ping-of-death" attack that can disable a host without ever triggering a host intrusion detection system. A NIDS is obviously of little value in detecting attacks that are launched on a host through an interface other than the network.

Network data has a variety of characteristics that are available for a NIDS to monitor: most operate by examining the IP and transport layer headers of individual packets, the content of these packets, or some combination thereof. Regardless of which characteristics a system chooses to monitor, however, the positioning of a NIDS fundamentally presents a number of challenges to its correct operation.

On a heterogeneous network, a NIDS generally does not

possess intimate knowledge of all of the hosts on the network and is incapable of determining how a host may interpret packets with ambiguous characteristics. Without explicit knowledge of a host system's protocol implementation, a NIDS is impotent in determining how a sequence of packets will affect that host if different implementations interpret the same sequence of packets in different ways [57].

A savvy attacker can exploit this property by sending packets that are designed to confuse a NIDS. Such attacks are referred to as insertion and evasion attacks based on whether they insert additional information into a packet stream that a NIDS will see and the target host will ignore or if they evade detection by forging data in such a way that a NIDS cannot completely analyze a packet stream.

Protocol ambiguities can also present a problem to a NIDS in the form of crud. Crud appears in a network stream from a variety of sources including erroneous network implementations, faulty network links, and network pathologies that have no connection to intrusion attempts [97]. If a NIDS performs insufficient analysis on a stream containing crud, it can generate false positives by incorrectly identifying this crud as being intrusive. While a NIDS therefore is in a very convenient position whereby it has complete access to all packets traversing a network link, its perspicacity is challenged due to ambiguities in network data

and its limited perspective of host system implementations and network topology.

On the other hand, host intrusion detection refers to the class of intrusion detection systems that reside on and monitor an individual host machine. There are a number of system characteristics that a Host Intrusion Detection System (HIDS) can make use of in collecting data including:

File System: Changes to a host's file system can be indicative of the activities that are conducted on that host. In particular, changes to sensitive or seldom-modified portions of the file system and irregular patterns of file system access can provide clues in discovering attacks.

- **Network Events:** An IDS can intercept all network communications after they have been processed by the network stack before they are passed on to user-level processes. This approach has the advantage of viewing the data exactly as it will be seen by the end process, but it is important to note that it will be useless in detecting attacks that are launched by a user with terminal access or attacks on the network stack itself.

- **System Calls:** With some modification of the host's kernel, an IDS can be positioned in such a way as to observe all of the system calls that are made. This can provide the IDS with very rich data indicating the behavior of a

program.

A critical decision in any HIDS is therefore choosing the appropriate system characteristics to monitor. This decision involves a number of trade-offs including the content of the data that is monitored, the volume of data that is captured, and the extent to which the IDS may modify the operating system of the host machine.

4.4 Anomaly Detection vs. Misuse Detection

Misuse and anomaly detection are two techniques that are used to analyze the data about a computer system and its users, gathered using either host based or network based techniques, to detect intrusion.

Anomaly detection process involves first characterizing the behaviors of individuals or systems and then recognizing behavior that is outside the norm [44], by trying to identify events that appear to be anomalous with respect to normal system behavior [16]. One advantage of Anomaly detection approach is that it does not require any historical knowledge of abnormal behavior or anomalous records. But the draw back of this is that intruders can slowly train the system to accept their malicious behaviors.

Anomaly detection is concerned with identifying events that appear to be anomalous with respect to normal system

behavior. A wide variety of techniques including statistical modeling, neural networks, and hidden Markov models have been explored as different ways to approach the anomaly detection problem [47][69][80]. Each of these anomaly-based approaches fundamentally relies upon the same principles: anomalous activity is indicative of an attempted attack and the correct set of characteristics can sufficiently differentiate anomalies from normal system usage. Developing an anomaly detection system therefore involves first establishing a baseline model that represents normal system behavior and against which anomalous events can be distinguished. The system then analyzes an event by considering it within this model and classifying it as normal or anomalous based on whether it falls within a certain threshold of the range of normal behavior or intruder, respectively.

The most appealing feature of anomaly detection systems is their ability to identify new and previously unseen attacks. Because the process of establishing a baseline model of normal behavior is usually automated, anomaly systems also do not require expert knowledge of computer attacks. This approach is not without its handicaps. However, as anomaly detection may fail to detect even attacks that are very well-known and understood if these attacks do not differ significantly from what the system establishes to be normal behavior. Anomaly based systems are also prone to higher numbers of false positives, as all anomalous events are

assumed to be intrusive although in reality a variety of other factors can produce behavior that appears anomalous (e.g., implementation errors) [71].

Misuse detection is another approach that involves identifying patterns of known “bad” behavior, while anomaly detection looks for patterns of activity that appear to be abnormal [72]. Misuse detection requires a database (knowledge base) of historically known attacks. The gathered data is searched for patterns and signatures of well known attack types stored in the knowledge base. The drawback here is that new attacks unknown to the system can not be detected.

The essence of misuse detection centers around using an expert system to identify intrusions based on a predetermined knowledge base. As a result, misuse systems are capable of attaining high levels of accuracy in identifying even very subtle intrusions that are represented in their expert knowledge base; similarly, if this expert knowledge base is crafted carefully, misuse systems produce a minimal number of false positives [71].

A less fortunate ramification of this architecture results from the fact that a misuse detection system is incapable of detecting intrusions that are not represented in its knowledge base. Subtle variations of known attacks may also evade analysis if a misuse system is not properly constructed. Therefore, the efficacy of the system relies heavily on the

thorough and correct construction of this knowledge base, a task that traditionally requires human domain experts.

4.5 Centralized vs. Distributed Intrusion Detection

Each component of an intrusion detection system may be either centralized – all functionality performed at a single location – or distributed. The major difference among intrusion detection systems is related to the centralization or distribution of the process of data-collection and/or data-processing. Usually the underlying monitored system has the major influence on whether to centralize or distribute a component.

Small systems, such as few PCs linked with a Local

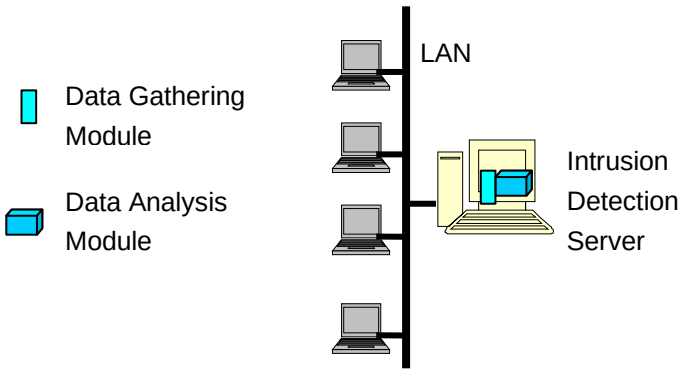

Figure 4.5: Centralized intrusion detection.

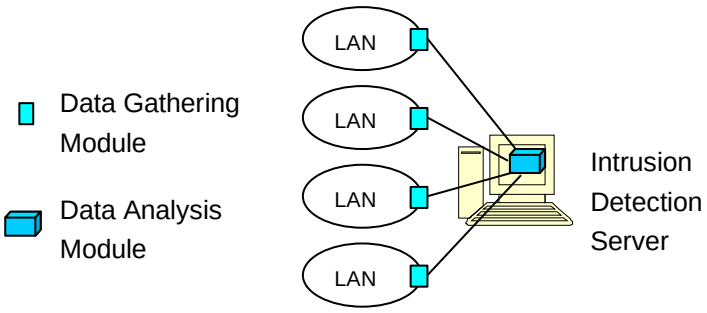

Figure 4.6: Simple distributed intrusion detection.

Area Network (LAN), as shown in *Figure 4.5* usually adopt a pure centralized approach. A powerful intrusion detection server is used that has the capability of both collecting data about the entire monitored system (the data gathering module) and performing all the required analysis on the collected data. Data reduction, data storage, and alarms among others are also the responsibility of the central server. Examples of centralized systems are [6][44][46].

When the size of the monitored system increase in larger organizations sometimes become hard or even impossible for a single centralized intrusion detection server to do all the work. In this case some of the intrusion detection system components must be distributed. The simplest solution is to distribute the data gathering module. That will gather the data, do necessary selection/reduction for the data, and then send it to a

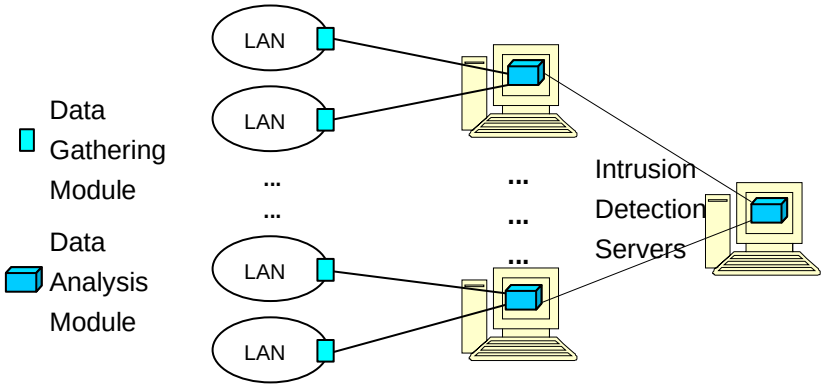

Figure 4.7: Hierarchical distributed intrusion detection

centralized analysis server for processing, as shown in *Figure 4.6*. A more complicated approach is to distribute the analyzing module, as shown in *Figure 4.7*. This is usually done together with the distribution of the data gathering module. The distributed analyzing modules are usually organized in a hierarchical manner. Examples of distributed intrusion detection systems are [41][58][77]. For Grid computing architecture, either class of distributed intrusion detection systems is more suitable due to the distributed nature of the Grid computing architecture.

4.6 Other Classifications and Attributes

There are other attributes that can be used in classifying intrusion detection systems. These attributes may include:

- **Detection Time:** Depending on the time needed to detect the intrusion from its beginning, intrusion detection

systems can be either real-time systems where intrusion are detected after they happen or non-real-time systems that depend mainly on audit analysis. Sometime they are called on-line and off-line intrusion detection system respectively.

- **Response Type:** After an intrusion is detected, there must be some actions taken to stop it. Depending on the type of this action in response to the intrusion, systems can be classified to either passive or active. Passive systems response by only notifying the security administrator who should take the appropriate actions to stop the intrusion. On the other hand, active systems response by performing appropriate actions depending on the type of intrusion. Such as, closing a session, suspending an account, shutting down a devise, or closing a port in the firewall. These actions are performed automatically without referring to the security administrator.

- **Granularity of Data-Processing:** This attribute contrasts between systems that analyze the data continuously, as it arrives, and those analyzing data as batches at some regular time intervals. This is related to the detection time mentioned above but they are not the same. A system may process data continuously but with certain delay making it non-real-time while other system may process data in small batches at real-time.

- **Intrusion Detection System Security:** This measures the ability to withstand attacks against the intrusion detection system itself. This area is not yet studied well. It is measured as a value on a high/low scale. Most systems don't address this issue.

- **Degree of Interoperability:** This attribute is concerned with to what degree an intrusion detection system can work in conjunction with other systems, accept audit data from other sources, and so on. This is the interoperability with other systems and not with the same system running on a different platform.

4.7 Problems of Traditional Intrusion Detection Systems

Centralized intrusion detection systems architectures such as [6][44][46] depends on a centralized server that is capable of monitoring and analyzing the entire network to detect intruders. These systems have one server with both DGM and DAM running on it. Centralized intrusion detection architectures are not suitable for Grid environments because reasons such as their poor scalability. A computational Grid size varies from a few numbers of computers in an organization to span the entire earth. Centralized architectures can not scale with this huge increase in grid size and the centralized server will not be able to analyze this huge amount

of data. Another problem happens if this server is attacked or failed for any reason, then the entire system will be affected and fails. Another reason is that, in the case of large Grid environments, it is not possible to find a centralized server that can monitor all the resources in the grid, because of the distributed nature of the networks linking its resources. The fact that the grid consists of resources controlled by different administrative domains makes another problem because it is not possible to find a single server that all these administrative domains can trust, agree to use and depend on it.

Distributed intrusion detection systems such as [41][58][77] are more suitable for Computational Grids. They have enhanced the scalability by distributing some of the components of the system, such as distributing the DGM while keeping the DAM centralized or in some systems distributed by taking a hierarchical form with a centralized control at the top. Distributed systems, although enhanced, are still not sufficient for Computational Grids. The components which are left centralized or components near the top of a hierarchy forms a performance bottle neck and a single point of failure. Intrusion detection systems designed for Computational Grids must address the important fact that the grid consists of different administrative domains that should not necessarily trust each other, agree to share information and work together to detect intruders or even believe each other if one of them found an intruder and warned the other administrative

domains. These complex trust relationships must be addressed by intrusion detection systems to be suitable for Computational Grids.

4.8 Conclusion

As described in this chapter, intrusion detection systems are becoming one of the main research areas in the field of computer security. The research in intrusion detection field covers a wide variety of systems, including centralized and distributed systems. Unfortunately, these systems have some limitations making them not 100% compatible with the Grid environment. This mismatch appeared because the capabilities of intrusion detection systems dose not satisfy some or all of the Grid characteristics and requirements, making them not suitable for implementation in Grid environments.

The rapid development of both intrusion detection systems and Grid computing architectures, motivated us to merge the two fields by designing and implementing a proposed intrusion detection model that will fulfill the needs of Grid computing systems.

Chapter 5

The Proposed Grid Intrusion Detection Architecture

Chapter 5: The Proposed Grid Intrusion Detection Architecture

Currently, the Grid Security Infrastructure (GSI) does not have any intrusion detection system that can avoid any of the previously presented security leaks. Hence, computational Grid environments needs intrusion detection system as a second line of defense after normal security mechanisms to protect themselves from intruders that will try to misuse the valuable resources made available by the Grid to any legitimate user from any location on earth. This is critical in the Grid environment because if the attacker masqueraded as a legitimate user or a user misused his privileges this will result in a high security risk not just at a single resource but at all the resources this user can access and use.

Special Grid characteristics and previous intrusion detection systems architectures were kept in mind while designing the Grid Intrusion Detection Architecture, or GIDA. The design was aiming to create an open, extensible, and general architecture that is compatible with the Grid and its requirements while tacking advantages of available Grid services and protocols as possible.

This chapter presents and examines the GIDA in details and its compatibility with the Grid is illustrated at the end of this chapter.

5.1 Problem Definition

Intrusion detection systems have many components as presented in *Chapter 4*. However two of these components are essential for GIDA. The First module is the ***Data Gathering Module (DGM)*** that is responsible for collecting data about the monitored system that may contain useful information that could give clues about intrusion. The second module is the ***Data Analysis Module (DAM)*** that analyzes the gathered data trying to detect any intrusion.

Taking into consideration the special features of the Grid architectures, implementing any of the traditional intrusion detection models to GSI will have some implications and incompatibilities. Therefore, these models should be customized to fit into the GSI new requirements.

A Grid intrusion detection system should protect the resources from attacks that could happen from the Grid users. These attacks could happen as a result of these resources installing the Grid infrastructure software and so joining a grid community. The Grid intrusion detection should not provide protection against normal Internet attacks that do not require a Grid environment, because its role is to protect Grid environments from Grid attacks. However normal attacks should be handled locally using local intrusion detection systems. There may be cooperation between local intrusion

detection systems and the Grid intrusion detection system in the form of a warning message or signal to warn the Grid intrusion detection system in the case of a local attack to prevent it from spreading to other Grid resources.

The Grid intrusion detection system must also be compatible with the Grid requirements and constraints presented earlier in *Section 1.1*. In other words it should be scalable to cope with growing size Grid environment and suit different small and large Grid environments. It should support heterogeneous resources and interoperate rather than replace existing systems. The Grid intrusion detection system should handle the dynamic nature of the grid and the fact that the failure is a rule not exception, so it should not be designed for a specific Grid organization. The design should keep in mind that the Grid is not owned by a single organization, so the detection should be cooperative and not subject to centralized control. It should also enable the delivery of different QoSs because each participant in a Grid environment may have different requirements and security levels. The design should use standard and open protocols as possible to enable interoperability and make expansions easy.

5.2 The Proposed Grid Intrusion Detection Architecture

The Grid Intrusion Detection Architecture GIDA

[62][64] was designed with all the previously mentioned problems in mind. GIDA is build on top of the Grid infrastructure [36] including the Grid Security Infrastructure GSI [37] which provides a uniform security infrastructure for Computational Grids and interoperates with the diverse intradomain security solutions. This means that GIDA uses the services and protocols provided by the Grid to build a new layer on top of the Grid layers to provide new services for

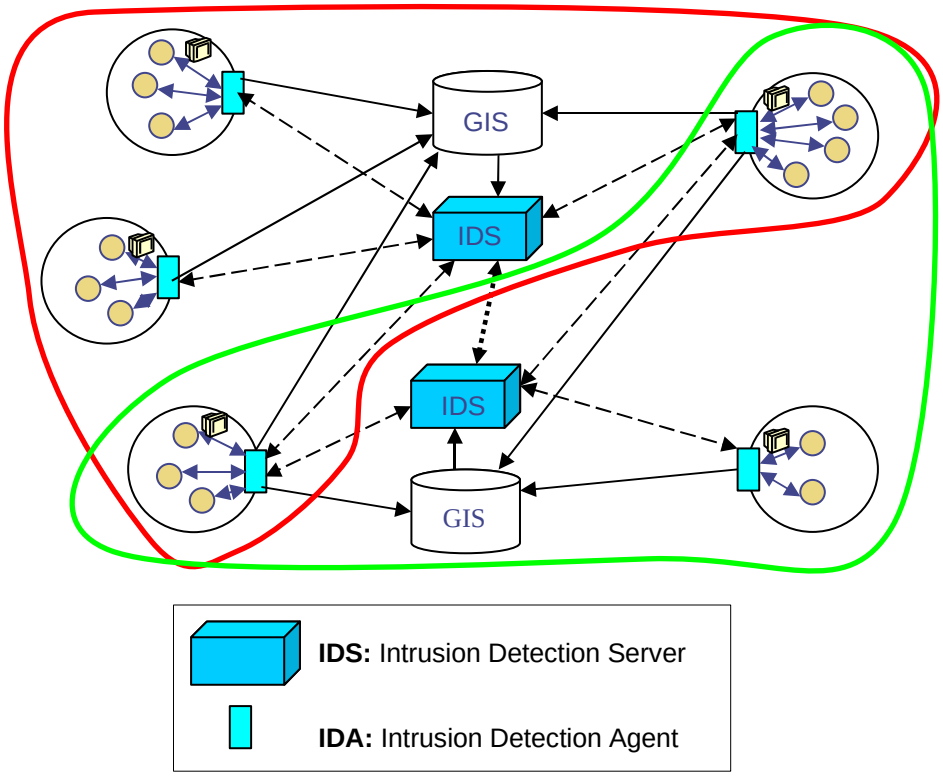

Figure 5.1: The proposed Grid intrusion detection architecture

intrusion detection.

The proposed GIDA is summarized in *Figure 5.1*. The main idea is to distribute the problem of the Grid intrusion detection among different distributed components that work together to detect intruders. The Grid environment can be considered as a virtual environment built on top of the Internet or any other type of network architectures. These Grid intrusion detection components will interact only with the Grid infrastructure, so they will exist in this virtual Grid environment and detect only Grid intruders. Meanwhile, these components will take advantage of the services provided by the Grid environment.

The GIDA has two main categories of components. The first one, corresponds to the data gathering module (DGM), called the **Intrusion Detection Agent (IDA)** which is responsible for gathering data about the users and resources from a specific administrative domain. The second component, corresponds to the data analysis module (DAM), is called the **Intrusion Detection Server (IDS)**, which is responsible for analyzing the gathered data and cooperate with other IDSs to detect intruders. Both of these modules, IDAs and IDSs, are distributed and not subject to centralized control as shown in *Figure 5.1*.

As stated above the Grid consists of resources owned by

different administrative domain, this is represented by the circles in *Figure 5.1*, each administrative domain will have an intrusion detection agent responsible for gathering data that is specific to this administrative domain and summarizing this data and converting it to a standard format. In other words this will deal with the heterogeneity of Computational Grid resources. Summarizing the gathered data will reduce the consumed network bandwidth when sending data to be analyzed but will require preprocessing at the administrative domain (client).

Each intrusion detection agent (IDA) will register with one or more intrusion detection server (IDS) this will increase the reliability, robustness and adaptability of the system. In the case of the failure of one IDS the administrative domain can still be protected against intruders if its IDA is registered with other IDS. Of course there is an overhead of this increased reliability and robustness in the form of the increased bandwidth consumption and the time needed by the IDS to analyze the data, this is due to the increase in the size of the data being transferred and more processing needed to analyze them. So there must be a trade off between the performance and its cost. The registration with multiple IDSs also allows the delivery of complex QoS if each one of these IDSs uses different approach to detect intruders with different properties.

After gathering information, these gathered information will be transferred from each intrusion detection agent IDA to all registered intrusion detection servers IDS. The Grid Information Service (GIS) [51] which is a major component of most Computational Grids can be used to store this information. If the GIS is not available or not applicable a special database can be implemented in the IDS to store the gathered information for analysis.

The Intrusion Detection Servers (IDS) will analyze the gathered information from the different IDAs and try to detect intruders. These IDSs must not be homogeneous. Each IDS can use a different approach to analyze the gathered data such as anomaly or misuse detection and each one can be designed and implemented using either neural network, statistical, data mining, or other technique for intrusion detection [79]. The key here is to use standard, open, general-purpose protocols between the IDSs that allow them to cooperate and work together.

When an IDS detects an intruder it should warn the other IDSs which in turn will signal the registered IDAs that will worn the local security to take an appropriate action. Because GIDA is build on top of the Grid infrastructure, GIDA can take advantage of the user's ***global name***. When an IDS detects an intruder it warns other IDSs using the user's global name. When an IDS receive a warning it has the choice

whether to accept or deny the warning according to the IDS QoS and trust relationship to the IDS sending the warning. If the receiving IDS chooses to accept the warning then the user's global name is mapped to the local name at each IDA registered with the IDS that accepted the warning. Using the local name the account can be disabled and appropriate actions should be taken. This example shows the benefit of building on top of the Grid infrastructure because using the global name of the user enabled interoperability between IDSs which would be otherwise difficult to implement.

The administrative domains can have local intrusion detection system that detects local intruders. This local intrusion detection system can cooperate with GIDA to help it find the intruders. When an intruder is detected locally, the IDA is warned which will then signal all the registered IDSs that are then responsible to warn other IDSs and signal other registered IDAs to take appropriate actions. In this case the local intrusion detection system will detect the intrusion based on the local name of the user that will be converted at the IDA to the Grid global name of that user. The global name will be used as in the previous example to warn other IDSs.

5.2.1 The Data Gathering Module

As mentioned before, the DGM is responsible for gathering data about the monitored system that may contain

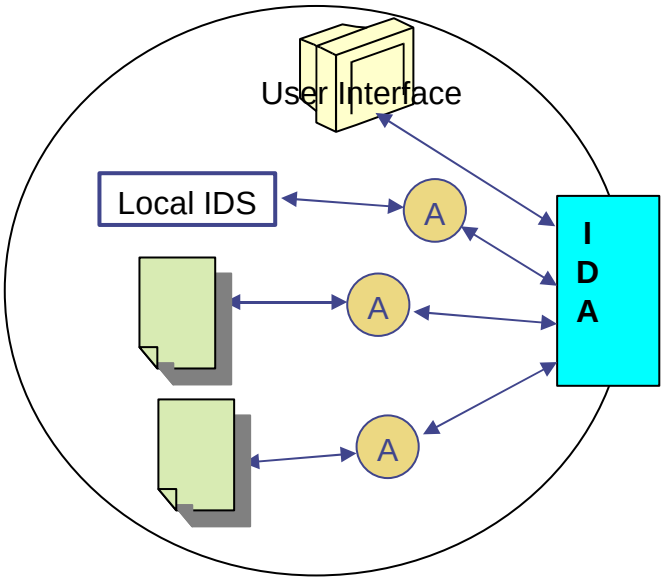

Figure 5.2: The Data Gathering Module (DGM)

useful information to detect intrusion. Taking a closer look at the DGM as shown in *Figure 5.2* will show its main components. The small circles labeled with "A" are agents that work for the IDA and are specialized in monitoring a specific resource or component of the system. Such as system log files, processes started, requests for processor time and storage pace, and so on. An agent can also be monitoring the local intrusion detection system to get data about locally detected intruders. These agents are ***system dependent***. They are created specially for each system depending on its hardware and software platforms. These agents are responsible to deal with the Grid heterogeneity. They are designed specially for each system but can be used to get information about that system in a standard

format.

The Intrusion Detection Agent (IDA) is the main component of the data gathering module. It uses the agents to get standard information about the monitored system. This information is summarized and formatted in a standard format to be ready for analysis. The IDA is also responsible for registration with one or more IDSs. The gathered data is then sent to all registered IDSs.

There is also a user interface for the DGM that is connected to the IDA and used to monitor and configure the agents and also to register with IDSs and to check the status of the system and to see if intruders are detected.

5.2.2 The Data Analysis Module

The Intrusion Detection Server (IDS) corresponds to the Data Analysis Module (DAM). It has two sub modules that are the analysis-and-detection module and the cooperation module. The IDS first receives the data from the registered IDAs and stores this data in the Grid Information Service (GIS) or a special database. Then the analysis and detection module analyzes the gathered data and queries the cooperation module to obtain results from other analysis and detection modules of other registered IDSs and uses them all to generate a final decision about the current users (intruders / normal).

The analysis can be done using any approach based on the decision of the implementer. The cooperation module should use standard and open protocols to enable interoperability. The proposed system has the ability to either use the Grid Information Service (GIS) protocols to locate and query other IDSs or to use Peer-to-Peer (P2P) protocols to locate and query other IDSs. Both approaches are distributed, scalable, and fault tolerant. In the case of P2P each IDS is considered as a peer in a community where each peer contributes with its analysis results in favor of being able to get other results. The only restriction here is that, an IDS only exchange information with other trusted IDSs according to a predefined trust relationship.

The GIDA can be viewed as abstract layers in a hierarchical manner as shown in *Figure 5.3*. The First layer is the specialized agents responsible for data gathering about the monitored system. The Second layer represents the IDAs. Each IDA is responsible for one administrative domain, it receives the data from the agents and prepare them and send them to be analyzed. In addition it may receive warning signals from IDSs. The Third layer has the IDSs which are responsible for the analysis of the gathered data.

Looking at the hierarchy in *Figure 5.3* it can be noticed that there is no head or root for this hierarchy. This is due to the fact that all components of the GIDA are distributed and

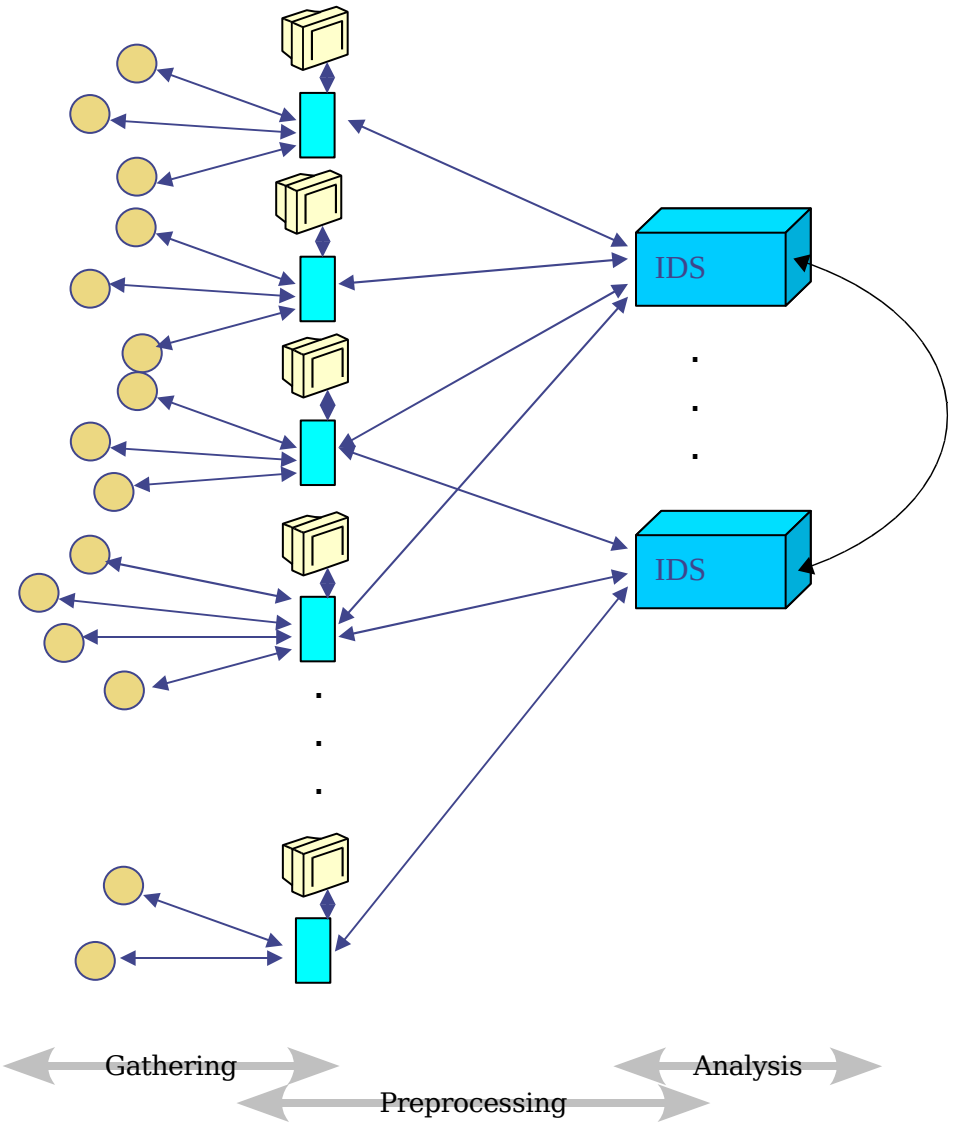

Figure 5.3: Hierarchical view of the proposed GIDA.

there is no single centralized control module to be placed at the top. The figure also shows the overlapping of the different

stages of data gathering, preprocessing, and analysis of the proposed GIDA.

The proposed GIDA is an extensible and open architecture that can be implemented in various ways. It meets the characteristics and requirements of Computational Grids, as elaborated in the next section. The detailed implementation of the proposed GIDA modules and protocols are illustrated in the next chapter.

5.3 GIDA Compatibility with the Grid

The GIDA should be compatible with the characteristics and constraints introduced by the Grid to ensure its successful implementation in a Grid environment. The GIDA implementation should be built on top the Grid Services – introduced in ***Section 1.1*** – as following:

- **Resource Management:** The agents should use the log files generated by the resource management services to gather useful information about user interactions with the resources. The IDAs and IDSs themselves can be considered as resources that are managed by the resource management services. For example an IDA can request registration to an IDS through a resource management service.

- **Information Services:** IDAs should use the Grid information services to find available IDSs, query their

capabilities, and check their status. IDSs should also use these services to insure that the registered IDAs are still available and not disconnected. IDSs may also use information services to locate and query other IDSs.

- **Data Management:** Data management services should be used to securely and efficiently transfer data from IDAs to IDSs and also to transfer results among IDSs.

- **Security:** The security issues should be managed using Grid Security Infrastructure (GSI). Each IDA and IDS should have a certificate that will be used in authentication between IDAs and IDSs, and also between IDSs themselves. The agents are considered local because they only work inside a domain and communicate only with this domain's IDA, so they do not need certificates. The security services are used to enforce trust relationships, because authentication will fail if the two participants do not trust each other.

The Grid characteristics and constraints were also introduced in *Section 1.1* and in the following the GIDA is checked to be compatible with them:

- **Heterogeneity:** The IDA, and its agents, deals with the heterogeneity of the Grid resources. For each class of resources – Clusters, Super Computers, Databases, and so on – special agents and an IDA is built to suit that resource

hardware and software. The IDSs are also heterogeneous so they can cover the needs of all resources.

- **Scalability:** The GIDA components are all distributed. Centralized components were avoided because they cause problems with scalability. Although distribution guaranties scalability but it does not ensure performance and efficiency of the system. This issue will be examined in the next chapter.

- **Dynamicity or Adaptability:** In a Grid environment the failure is the rule not the exception. With a large number of resources distributed among different administrative domains, each resource may join or leave in an uncontrolled manner. To handle this there are multiple IDSs so in the case of the failure or unavailability of an IDS other IDSs can do its job. Also to increase the fault tolerance, each IDA can register with multiple IDSs as presented before.

- **Multiple administrative domains and autonomy:** Resources in the Grid are controlled by different administrative domains and owned by different organizations. The autonomy of these sites must be protected. Not all domains agree to work with each other and share information. The GIDA addresses this problem through providing different trust relationships among participants.

- **No Centralized Control:** There is no centralized control in GIDA. The decision is made through cooperation among IDSs and each domain has the choice to accept or deny a warning signal.

- **Using Standard Protocols:** The GIDA is built on top of the Grid and uses standard, open, and general purpose protocols.

- **Deliver None Trivial QoS:** The heterogeneous nature if IDSs that enabled them to implement different intrusion detection techniques with different properties together with allowing an IDA to register with multiple IDSs has enabled each domain to deliver non trivial quality of service.

The above analysis shows that the design of the GIDA was made with the Grid and its properties in mind. The compatibility of the GIDA with the Grid makes it a good candidate for Grid intrusion detection systems. Following the GIDA can generate many systems with different characteristics and performance.

Chapter 6

The Proposed GIDA Implementation

Chapter 6: The proposed GIDA Implementation

This chapter presents a proposed implementation for the proposed Grid Intrusion Detection Architecture (GIDA) components. This implementation represents an instance of the general and open GIDA. This proposed implementation will use computer simulation to simulate different organizations of Grid environments including resources, users, and IDAs. The IDSs in this proposed implementation will be homogeneous and implemented using Learning Vector Quantization (LVQ) neural network for data analysis and simple protocols for cooperation between the IDSs.

The proposed GIDA needs to be tested and verified to prove its compatibility with the Grid environment and to prove its applicability in real world. The purpose of this implementation is testing and evaluating GIDA and give an insight into its properties, so the implementation should be simple and use basic and standard techniques to concentrate on the GIDA properties and to ensure that the focus is not diverged to complex techniques and fine tunings. The simple and well known techniques will also enable experts to predict the effect of changing them – with more efficient and more advanced techniques – on the GIDA.

6.1 Simulating the Computational Grid

Computer simulation has always been used as a cost effective solution for the evaluation, testing, and proving the effectiveness of new architectures and models before implementing them in real world applications [4]. Simulation also allows researchers to perform experiments repetitively using different combinations and arrangements in a controlled environment to find the most optimum solution in an effective way that would otherwise be both cost and time consuming.

Researchers in the field of Computational Grids face many problems in their research because of the special characteristics of Computational Grids. Below is a review of some of the main problems.

Most of the researchers do not have access to real Computational Grids or testbeds such as [10][76][78][99] to perform their experiments on it. This is due to the high cost and technical and organizational challenges needed to build a real Computational Grid.

Even those who have access to real Computational Grids face problems. Computational Grids contain expensive resources such as super computers, large clusters, other expensive devices such as electronic telescope, and so on. Dedicating a portion of these resources' time to researchers to perform their experiments increases research costs and in

many cases is infeasible and not applicable.

Computational Grid applications can take days or even weeks to complete and are complex in nature. Experimenting using real applications will waste lots of researchers' time and effort because of the long period of time needed to complete their execution and the time needed to build test applications to examine different application models and problem solving approaches.

Even if assuming that both the resources and test applications are available for the researchers and that the time and cost constraints are relaxed. It is very difficult to coordinate and control the experiment and gather information about it. This is due to the large size of Computational Grids and the fact that both the resources and users are geographically distributed and are owned by different administrative domains which makes the coordination between them very complex, creating a controlled environment very difficult because of their dynamic nature, and repeating the experiment and testing different combinations and different resource arrangements and scenarios with varying specifications and loads considered impossible. Testing the scalability is another problem which is limited by the size of the Computational Grid available to the researchers.

Because of the above problems most of the researchers have turned to simulate Computational Grids. Tools for

simulating Computational Grids have been developed and used in research including for example: GirdSim [59], SimGrid [25], ChickSim [12], Brics [28], and MicroGrid [49]. Researchers use this simulated Computational Grid to test their algorithms, models, and architectures. After they are well established and tested, they are implemented on real Computational Grids and retested only in the final phase. This reduces the time, cost, effort, and accelerates the research and give better results.

6.2 The Intrusion Detection Agent Implementation

The Grid Intrusion Detection Architecture (GIDA) can be divided into two main modules: the Intrusion Detection Agent (IDA) and the Intrusion Detection Server (IDS). For the purpose of validating and testing the GIDA, the IDAs and the Grid environment where simulated. This helps in overcoming problems similar to those presented in the previous section. The IDS is then tested using the data generated from the simulation.

Unfortunately most of the available Grid simulation tools are designed to solve problems related to resource management and scheduling ignoring security related requirements such as authentication, authorization, users' behavior, and managing trust relationships between different administrative domains. For these reasons a new Grid

simulation toolkit was developed that addresses security requirements which will be used to test the proposed Grid Intrusion Detection Architecture.

6.2.1 The Simulation Problem Definition

Testing and validating the proposed Grid Intrusion Detection Architecture requires performing experiments on a variety of resources and users arrangements with varying parameters including:

- Increasing the number of users and resources to test the scalability of the architecture by measuring its ability to detect intruders with this increase in grid size.

- Comparing between standard distributed intrusion detection systems having a single analyzing and detection module (one IDS) and the proposed architecture with increasing number of IDSs (decreasing the scope of each IDS).

- The effect of overlapping the scopes of the IDS (each resource can register with one or more IDS) and the ability of the architecture to detect intruders. In addition, study the effect of this overlapping with different degrees on the robustness and fault tolerance of the proposed architecture in case of one or more IDS have been compromised.

- The effect of trust relationships between different

administrative domains on the cooperation between different IDSs to detect intruders.

Performing these experiments on a real Computational Grid has many problems including:

- Controlling the number of users and resources to test the scalability.

- Log files containing data about users' actions can contain important data and sometimes are protected and not available to researchers.

- Most of the administrative domains will not agree to change their security policy and trust relationships to allow different experiments.

- Real Computational Grids are not controlled environment because of its dynamic nature, distributed large number of user and resources, and owned by different administrative domains. Because of this an experiment can not be repeated to test the effect of changing some parameters.

For these problems and others presented in the previous section, simulation becomes a good solution that overcomes these challenges and accelerates the research.

The simulation environment will allow performing the

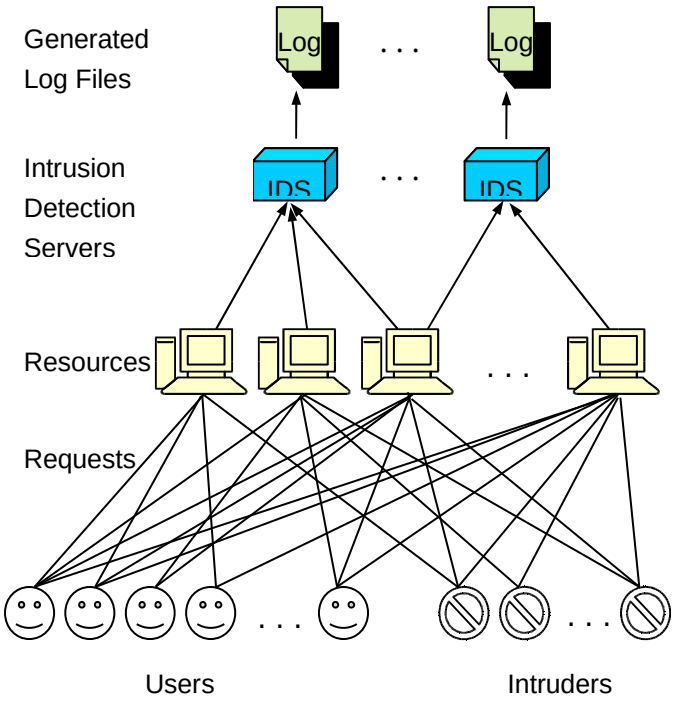

Figure 6.1: The simulated Grid and the IDA

required experiments. As summarized in *Figure 6.1* the simulated users will send requests to the available resources based on each user behavior. Intruders are also simulated and will also send requests to resources. The simulated resources will receive the users' requests and report them to the registered IDSs. The IDSs here are just dummy IDSs that do not perform any analysis but only dump the received data in a log file for later process by the real IDS. Each experiment will generate a dataset consisting of one or more log files. These

datasets are then used the process of testing the IDSs.

6.2.2 The Proposed Grid and IDA Simulator

The simulation environment consists of a set $\boldsymbol{U}=\{u_1, u_2, u_3, \ldots u_n\}$ of users, a set $\boldsymbol{I}=\{i_1, i_2, i_3, \ldots i_m\}$ of intruders, a set $\boldsymbol{R}=\{r_1, r_2, r_3, \ldots r_k\}$ of resources, a set $\boldsymbol{S}=\{s_1, s_2, s_3, \ldots s_p\}$ of services provided by the resources, and a set $\boldsymbol{D}=\{d_1, d_2, d_3, \ldots d_q\}$ of IDSs. In general the two inequalities should be true: $q<k<n$ and $m<n$. The value of p is related only to the number of services available in the simulated Grid environment. Each resource r_i will have a set $\bar{\boldsymbol{S}}_{r_i}\subseteq\boldsymbol{S}$ of available services, a set $\bar{\boldsymbol{D}}_{r_i}\subseteq\boldsymbol{D}$ of registered IDSs, and a set $\bar{\boldsymbol{U}}_{r_i}\subseteq\boldsymbol{U}$ of authorized users. Each user u_j will have a set $\bar{\boldsymbol{R}}_{u_j}\subseteq\boldsymbol{R}$ of resources that this user will use, and a set $\bar{\boldsymbol{S}}_{u_j}\subseteq\boldsymbol{S}$ of services that are provided by these $\bar{\boldsymbol{R}}_{u_j}$ resources.

Each user $u_j\in\boldsymbol{U}$ will pick a service $s_k\in\boldsymbol{S}_{u_j}$ and find a resource $r_i\in\boldsymbol{R}_{u_j}$ that have $s_k\in\boldsymbol{S}_{r_i}$ and sends a request for the service s_k to it. This will be repeated during the simulation. An intruder $i_l\in\boldsymbol{I}$ has the capability of requesting a service s_k from the resource r_i masquerading as the user u_j . Each resource r_i that receive a request from a user u_j will generate a record similar to the one shown in *Figure 6.2* that contains the user global name, service name, resource name, time of request, start time, end time, amount requested if

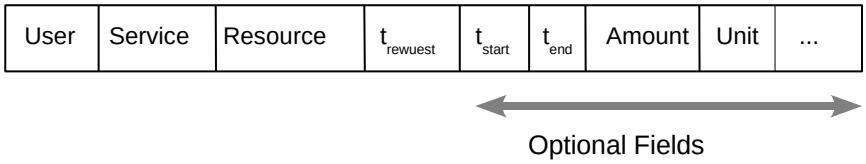

Figure 6.2: The log file record format

applicable, the unit used to measure the specified amount, and any other useful information. Then the generated record will be sent to all registered IDSs. Each IDS (dummy IDS) will receive records from resources in its scope and write it to a log file. After the simulation ends these log files will represent the dataset for the simulated experiment.

One major problem is how to simulate the user behavior. That is when and what service will a user pick to execute on which resource, in a way that is similar to users in real world where each user has his own unique behavior and habits. To do this each user u_j will have one or more profile. Each profile will consist of the following:

- The period of day (start time and end time) in which this profile is valid. For example there may be a profile for the morning another for the evening and so on.

- A general category of the profile. Such as administrator, programmer, scientist, and so on. This requires the services in the set S to be also classified to

similar categories.

- The average frequency or average number of request per unit time for this profile.

- An array **A** filled and shuffled with pairs *(a, b)* where $a \in \boldsymbol{R}_{u_j}$, and $b \in \boldsymbol{S}_{u_j}$ and belongs to the profile's category. Values μ and σ, which are the main parameters of normal distribution, are set for each array. These values are used to select a pair, using normally distributed random number, from the array to represent a user's request.

- A rate at which the profile will change to reflect changes in user behavior. Changes will be made by adding, removing, and swapping pairs from the array **A**, and by changing the values of μ and σ.

According to the current time *t* during the simulation the valid profile *p* of each user (one having *t* between its start time and end time) will be used to pick a resource and a service (by picking an element of the array using normal distribution) at a frequency equals to the profile average frequency, and the user behavior will be changing during the simulation time.

A trust relationship tree similar to the one shown in *Figure 6.3* is required to add trust management to the simulation system. This tree is similar to the relationship that exists between certificate authorities in systems that uses the

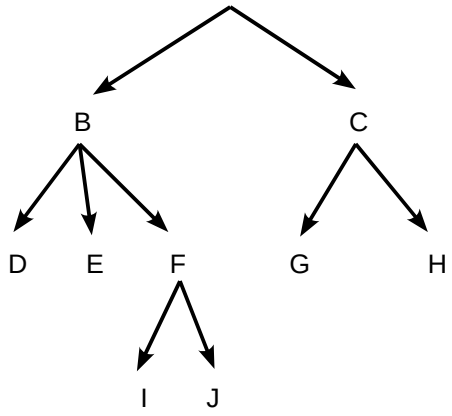

Figure 6.3: A trust relationship tree

Public Key Infrastructure PKI such as the Grid Security Infrastructure GSI [37]. Each entity in the simulation (users, resources, and IDSs) will have a trust level represented by one of the letters from the trust relationship tree. Each IDA can register with IDSs with the same level or above it in the trust tree; otherwise the registration will be refused. Similarly a user can request a service from resources with the same level or higher in the trust tree. For example in *Figure 6.3* a user with level J can request a service from a resource in level F or B, but not form a resource at level E or C. An IDA at level F can register with an IDS at level B but not with IDSs at level E or J. Trust relationships will also appear when talking about the cooperation between IDSs below.

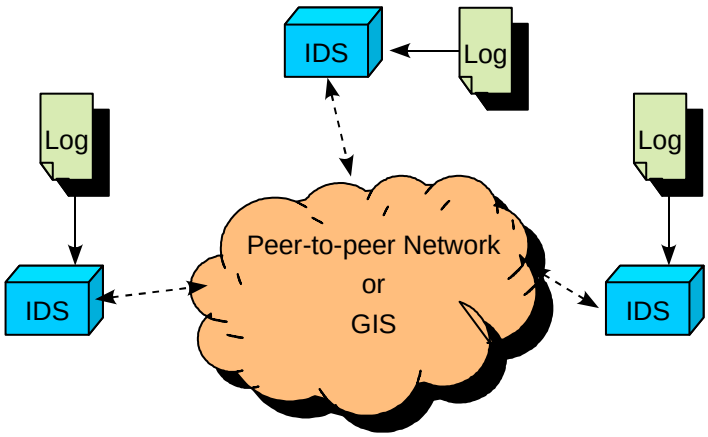

Figure 6.4: The analysis and detection module first utilize the simulated data, then cooperate through the cooperation module.

6.3 The Intrusion Detection Server Implementation

This component will analyze the data generated from the simulation with the goal of detecting intruders trying to compromise and misuse a Computational Grid. As shown in *Figure 6.4* the log file generated from the simulation step will be provided to the appropriate IDS. The IDSs will analyze this data using the analysis and detection module and cooperate with other IDSs using the cooperation module to detect the intruders.

The implementation of these two IDS components is described in the following subsections.

6.3.1 The Analysis and Detection Module

Intrusion detection systems are based on the assumption that the normal system use differs from malicious use [68]. Intrusion detection systems try to detect, using different mechanisms and approaches, this difference in behavior caused by an intruder and take appropriate action to stop the intruder. Intrusion detection techniques can be classified either according to the source of the data used for the analysis or according to the approach taken to analyze the data. In the first case it is classified into network based and host based intrusion detection systems [39][21]. In the second case it is classified into misuse detection and anomaly detection [16].

Network intrusion detection systems get their data by installing a device on the **network** capable of monitoring all network traffic and the passing packets. They rely on raw network packets in their analysis. On the other hand host based intrusion detection systems use log files created on each **host**, containing all the operations performed on the host, as the data source. In the context of Computational Grids the network intrusion detection has many disadvantages and problems. The following points summarize these disadvantages and problems:

- It is impossible to have a device installed on the grid capable of monitoring all the passing packets because of the large scale and distributed nature of the networks involved.

Even if this device is distributed on the network, moving the raw network packets to the IDS is not efficient so it must be preprocessed and summarized at each administrative domain before being sent to the IDS. This may add undesired overheads and complications to the systems.

- Because of security requirements in the grid most of the raw packets used are encrypted and this cause problems in network based intrusion detection.

- Analysis at a low level such as raw network packets makes higher level information, such as the global name of the user, not available or hard to discover.

- Network based intrusion detection systems sometimes analyze the raw network packets to guess what is the user is trying to do. While this information is already available in log files.

For these reasons it is recommended to implement the GIDA using host based intrusion detection. The data gathering module as presented before is responsible for gathering the data from the log files on each host (or administrative domain) and transferring it to the IDS. Because of these reasons areas labeled (1) and (2) in *Table 6.1* that use network based approach will not be used.

Misuse and anomaly detection are two techniques that

	Misuse	Anomaly
Network Based	(1) ✘	(2) ✘
Host Based	(3) ✘	(4) ✔

Table 6.1: Different approaches to intrusion detection

are used to analyze the gathered data to detect intruders. Misuse detection technique search the gathered data for **patterns** and **signatures** of well known attack types stored in a knowledge base [16] on the other hand anomaly detection technique tries to identify events that appear to be **anomalous** with respect to **normal** system behavior [16]. Currently the misuse detection technique can not be used in the context of the Computational Grids because the Computational Grids are still new and under research, signatures and patterns of attacks are not available and the creation of a knowledge base of well known attacks is currently impossible. In *Table 6.1* areas labeled with (1) and (3) which uses misuse detection techniques are not used in the context of Computational Grids.

From the previous analysis it is recommended to use

host based anomaly detection intrusion detection technique to implement the GIDA because it is suitable for Computational Grids, area labeled (4) in *Table 6.1*, it was also used to implement the Grid Intrusion Detection Architecture GIDA in this chapter.

Because the GIDA is an open architecture, the analyzing and detection module can be implemented using various techniques. Such as neural networks, statistical analysis, data mining, and so on. It is possible also to be implemented using different technique on different IDSs in the same Computational Grid. The implementation described in this chapter assumes homogeneous IDSs all using anomaly detection implemented with neural networks. The neural network used is the Learning Vector Quantization (LVQ). Properties of LVQ and reasons of choosing it are presented below.

6.3.2 The Learning Vector Quantization

Learning Vector Quantization (LVQ) [85] is a neurally inspired, nearest neighbor classifier based on Kohonen's work with self-organization [86][87][88]. Quantization can be defined as mapping a broad range of input values to a smaller number of output values. In LVQ, the input values can be thought of as the decision boundaries between a set of classes, and the output values are a predetermined number of nodes or

reference vectors. The strategy behind LVQ is to effectively train the reference vectors to define the Bayes Optimal decision boundaries between the classes. Correctly positioning the reference vectors in LVQ is accomplished in a supervised manner by presenting a training pattern to an input vector and adjusting the position of selected reference vectors in accordance with a set of learning rules, as will be described later in this section.

The training algorithms associated with LVQ attempt to adjust the position of the reference vectors so that each input pattern has a reference vector of the right category as its nearest neighbor. Classification, subsequently, is carried out using a nearest-neighbor method. Kohonen argues [88] that, asymptotically, the reference vectors approach the centroids of their resulting Voronoi tessellation. A Voronoi tessellation is a partition of $\Re^d$ into disjoint polytopes such that all training patterns within a polytope have the same reference vector as their nearest neighbor. Thus, the goal of LVQ is to approximate the boundaries of the Voronoi polytopes, therefore approximating the decision surfaces of a Bayesian classifier. Reference [3] provides a rigorous mathematical description of the LVQ process and proves convergence of the algorithm under certain asymptotic conditions.

There are several versions of LVQ reported in the literature [89]. While each LVQ algorithm attempts to

approximate the Bayes decision surfaces between classes, the learning rules associated with each LVQ algorithm are slightly different. In this initial research, the LVQ algorithm referred to as OLVQ1 was used. OLVQ1 is an enhancement of the original LVQ1 algorithm which is described below.

The LVQ1 Algorithm

Assume that a number of "codebook vectors" m_i (free parameter vectors) – here representing the legitimate users behavior – are placed into the input space – all possible behaviors – to approximate various domains of the input vector x by their quantized values. Usually several codebook vectors are assigned to each class – single user – of x values, and x is then decided to belong to the same class to which the nearest m_i belongs. Let

$$c = \min_i \{\|x - m_i\|\} \tag{1}$$

define the nearest m_i to x , denoted by m_c .

Values for x that approximately minimize the misclassification errors in the above nearest-neighbor classification can be found as asymptotic values in the following learning process. Let $x(t)$ be a sample of input and let the $m_i(t)$ represent sequences of the m_i in the discrete-time domain. Starting with properly defined initial values, the

following define the basic LVQ1 process:

$$m_c(t+1) = m_c(t) + \alpha(t)[x(t) - m_c(t)]$$

if x and m_c belong to the same class,

$$m_c(t+1) = m_c(t) - \alpha(t)[x(t) - m_c(t)]$$

if x and m_c belong to different classes,

$$m_i(t+1) = m_i(t) \text{ for } i \neq c .$$

Here $0 < \alpha(t) < 1$, and $\alpha(t)$ may be constant or decreasing monotonically with time. In the above basic LVQ1 in is recommended that α should initially be smaller than 0.1.

The optimized-learning-rate LVQ1 (OLVQ1)

The basic LVQ1 algorithm is now modified in such a way that an individual learning rate $\alpha_i(t)$ is assigned to each m_i . The discrete-time learning process will be as follows. Let c be defined by *Equation (1)*. Then:

$$m_c(t+1) = m_c(t) + \alpha_c(t)[x(t) - m_c(t)]$$

if x is classified correctly,

$$m_c(t+1) = m_c(t) - \alpha_c(t)[x(t) - m_c(t)] \quad (2)$$

if the classification of x is incorrect,

$$m_i(t+1) = m_i(t) \text{ for } i \neq c \ .$$

The optimal values of $\alpha_i(t)$ are determined by the recursion:

$$\alpha_c(t) = \frac{\alpha_c(t-1)}{1 + s(t)\alpha_c(t-1)}$$

Where $s(t) = +1$ if the classification is correct and $s(t) = -1$ if the classification was wrong.

6.3.3 Using LVQ for implementing IDSs

The LVQ neural network was used to implement the analysis and detection module part of all the IDSs as shown in *Figure 6.5*.

The LVQ was used for the following main reasons:

- The LVQ is similar to the SOM neural network and both are widely used for classification. LVQ is used in this context to classify user's behavior to either legitimate or intruder based on the previous user behaviors that were used to train the LVQ neural network. LVQ was also used in other intrusion detection systems such as [44].

- In the Grid environment LVQ take advantage over SOM because it is supervised and uses the available global user names as class labels during classification process. This

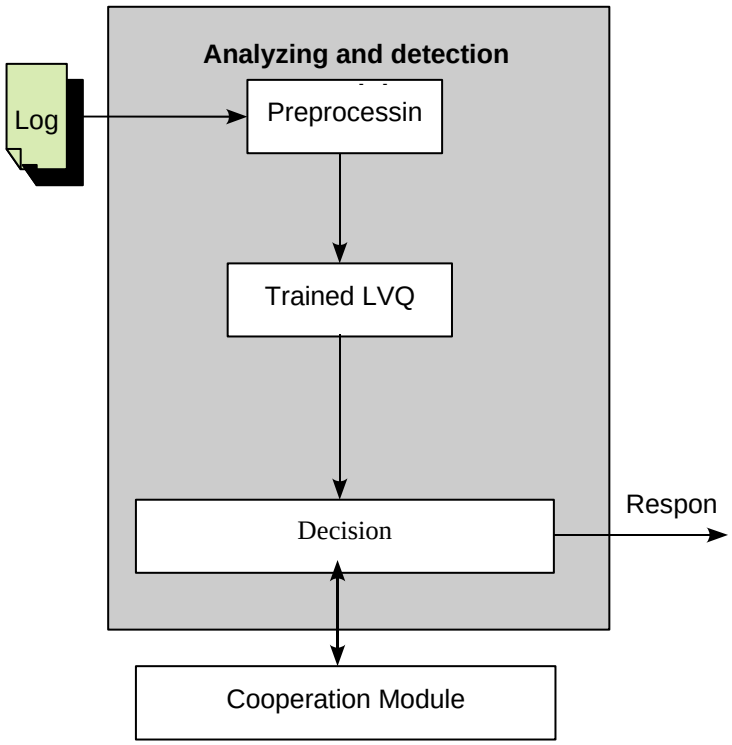

Figure 6.5: The analyzing and detection module

supervision helps improving the classification process.

- It is important to note this methodology does note require the generation of masquerader data. Training does not require the inclusion of malicious records. The system is trained based-on legitimate user data, and then measures the deviation of subsequent users from these profiles. Thus, the problem of negative data is obviated.

Before applying the data in the log files – generated from the simulation – to the LVQ it must be first reprocessed *Figure 6.5*. Preprocessing is done by grouping user's actions in the log files in windows. Each window will contain actions of the same user. The number of actions within a window can be specified by a fixed number of actions or by a specified period of time or both. The global user name is used as a label for the window. Each window represents an input **vector** used to train the LVQ and the label is used in the supervised learning process.

After training the LVQ is ready for detection. User's actions from the testing log file is also grouped in windows and applied to the neural network that will try to classify it to either normal or intruder. The decision and response are made based on this classification and the results of other IDSs classifications that are known through the cooperation module.

A challenge that all developers of anomaly detection based intrusion detection classifiers must address is feature selection/data reduction. Clearly, the inclusion of too much data will adversely impact the performance of the system, while the inclusion of too little data will reduce the overall effectiveness of the system. In addition, most anomaly detection approaches must address the problem of conceptual drift [43]. In this domain the problem of conceptual drift manifests itself in that a user s behavior changes over time. An

effective anomaly-based intrusion detection system should adapt to this change while still recognizing intrusive actions and not adapting to those.

Because the behavior of the users change slightly over time, the system after a period of time will give a high rate of false positive alarms (normal users classified as intruders). To overcome this problem the neural network must be retrained. The retraining can be manual based on the decision of the system administrator or automatic at a fixed period of time.

6.3.4 The Cooperation Module

Each IDS has a scope. This scope is defined by the administrative domains (resources) that chose to register with this IDS as was shown in *Figure 5.1*. The analyzing and detection module will make its decisions about users based on the data available in its scope. Because other information that exists outside its scope is invisible and not available for use. This will produce poor results because other important events may occur in the scope of other IDSs. Here arises the important rule of the cooperation module. It is responsible for distributing the intrusion detection problem among IDSs. Instead of having one IDS, there will be several IDSs each responsible for a portion of the Grid environment. The cooperation module will be responsible for sharing the results obtained at each IDS among the other IDSs. This sharing will

be achieved through a protocol that defines how will an IDS query and share its results with other IDSs.

This sharing can be either implemented using peer-to-peer techniques [19] or by using the Grid Information System (GIS) to share the results. Both techniques are distributed, does not rely on a central server and it supports the dynamic nature of Computational Grids.

The protocol used in this implementation is simple. Each IDS has a subset of users that are in its scope. For each user, the IDS will query other IDSs (peers) for their results of this user. Then the received results will be used together with the local result to decide whether the user is intruder or normal. The decision is made based on the number of miss classifications in a certain period of time crossing a predetermined threshold value. When an intruder is detected at an IDS, a warning will be sent to other IDSs to take appropriate actions.

This protocol can be enhanced by adding weights to the received results from other IDSs based on the characteristics of the sending IDS. This weight can be calculated for each user based on the length of history used to train the LVQ, the number of the users' records, and the scope of the IDSs among others.

The trust relationships between different entities add a

constraint on the protocol described above. As shown in *Figure 6.3*. An IDS can request from another IDS only if it has a trust level equal to or lower than the trust level of the requested IDS with respect to the trust relationship tree.

Chapter 7

Experimental Results

7.1 Evaluation Parameters and Test Approach
7.2 Data preprocessing
7.3 Number of IDSs
7.4 Number of users
7.5 Number of resources
7.6 Number of intruders

Chapter 7: Experimental Results

The analysis of the results from different test cases is important to give better understanding of the GIDA and the factors affecting its performance. This understanding will also help to fine tune an implementation of the GIDA to best fit a specific Grid environment.

This Chapter starts with listing the measurement parameters that will be used to evaluate the performance of the Presented GIDA implementation. Then shows and examines the values obtained for these parameters in different test cases each trying to illustrate different aspects of the GIDA implementation.

7.1 Evaluation Parameters and Test Approach

The performance of the proposed GIDA implementation is measured by the following five main parameters:

- **False positive percentage:** This parameter reflects the percentage of normal Grid users, which are miss classified by the GIDA implementation as intruders, from the total number of legitimate users in a specific Grid environment.

- **False negative percentage:** This parameter reflects the percentage of intruders that are miss classified by the

system as normal users from the total number of intruders in an environment.

- **Training time:** This measures the time (in minutes) needed to train the LVQ neural network. The software package used in the experiments is the LVQ_PAK version 3.1 [90] running on a PC with Pentium VI 2.8GHz processor and 512MB RAM. The Linux distribution called “Fedora Core 1” was used as the operating system.

- **Detection duration:** This measures the time duration (in minutes) needed by the GIDA implementation to detect that there is an intrusion occurring. It is the duration from the start of the attack until it was detected.

- **Recognition percentage:** This is measured after training the LVQ neural network with test data to check how the accuracy level of the LVQ to correctly classify and recognize users with the absence of intruders. Ideally this should be 100% but because similarity and overlapping of the users' behaviors, some users are miss classified.

The value of these parameters is affected by the environment and conditions in which the system is running. In this variable environment there are **controllable** issues such as the **data preprocessing** applied to raw data before using it to train the LVQ and the **number of IDSs** that exist in a certain

Grid environment. On the other hand there are **uncontrollable** issues in a Grid environment such as the **number of users**, **number of resources**, and **number of intruders**. The designers and administrators of a deployed GIDA implementation in a certain Grid environment must adapt the controllable issues to best fit the uncontrollable issues to give the best possible performance. The remainder of this chapter presents different experiments each concentrating on the effect of one issue of the environment on the performance of the presented GIDA implementation. The results from these experiments will both help in proving the applicability and suitability of GIDA to Grid environment and help administrators of GIDA implementations to fine tune and control its performance.

7.2 Data Preprocessing

The records in the log files are preprocessed (*Figure 6.5*) before applying them to the LVQ neural network. This preprocessing is done by grouping several records or events for the same user in a single victor that will be later applied to the input neurons of the LVQ. A window manages the process of grouping records together. Records for the same user that lies in the same window are grouped together. The records inside a window are selected depending on the type of the window used in the preprocessing. The window can take several forms (*Figure 7.1*) depending on how it is controlled.

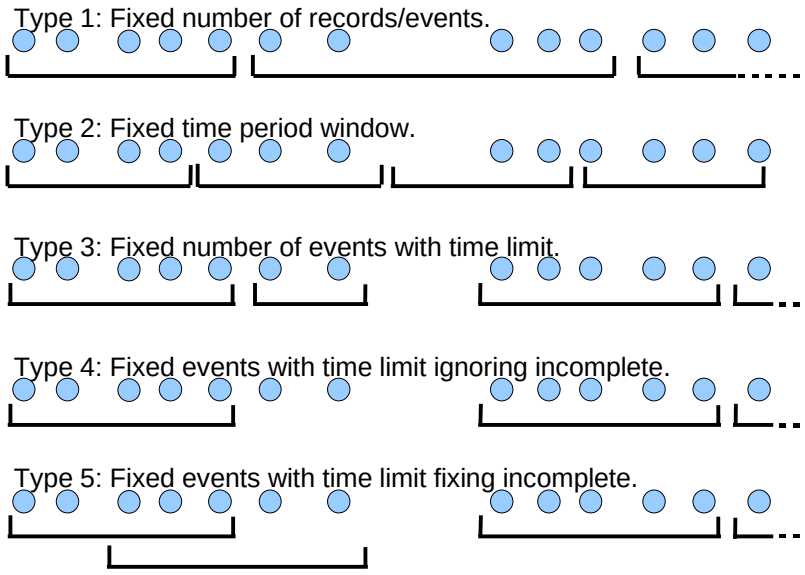

Figure 7.1: Different possible types of windows

It can be one of the following:

- A window can be controlled by a fixed number of records in each window (Type 1). For example if the size of the window is ten, then each ten records for the same user are grouped together.

- A window can be controlled by a fixed time period for the window regardless of the number of records in this period (Type 2). For example if the window has a fixed period of five minutes then all the user's records that happened in these five minutes are grouped together.

- Another possibility is to have a mix of the two previous types which will create a hybrid window with both size and time limits. In this case the number of records is determined depending on which limit is reached first (Type 3). For example a window may have a fixed size of ten records and a period of five minutes. In this case if ten records are generated in less than five minutes then they are grouped together and new window starts, on the other hand if five minutes passes while having less than ten records then they are also grouped and new window starts.

- The hybrid approach can have a problem when a window has a few number of records. Because these windows will not have enough information to reflect user behaviors. This problem can be fixed either by ignoring incomplete windows (Type 4), or by fixing incomplete windows by overlapping it with the previous window and adding some of its records (Type 5) so it will have enough information about the user.

The results in *Figure 7.2* are for fixed size window (Type 1) starting from 5 records to 40 records. *Figure 7.3* shows the results for fixed duration window (Type 2) starting from 100 seconds to 2400 seconds. The hybrid window (Type 5) results are shown in *Figures 7.4, 7.5, and 7.6* for different combinations of size and duration. Note that in the hybrid approach increasing the window duration will make it similar

to Type 1, while increasing the window size will make it similar to Type 2.

Generally the experiments in *Figures 7.2 to 7.6* test the effect of increasing the number of records in the window on the system performance. This increase in the number of records in the window may happen because of increasing its size, duration, or both depending on the type of window used. The following general behavior was conducted from these experiments as a result of increasing the window capacity:

- The false positive percentage is reduced as a result of increasing the window capacity. This is normal as more records in a window means that it will contain more information about the user and reduces the probability of miss classification.

- On the other hand, this increase in the capacity minimized the effect of malicious records because they are grouped with many legitimate records. This resulted in an increase in the false negative percentage which is not desired.

- Increasing the capacity means that completing the window will need longer time. This resulted in increasing the detection duration time which is not desired because the LVQ must wait until the window is complete before it can tell wither it match its user's behavior or not.

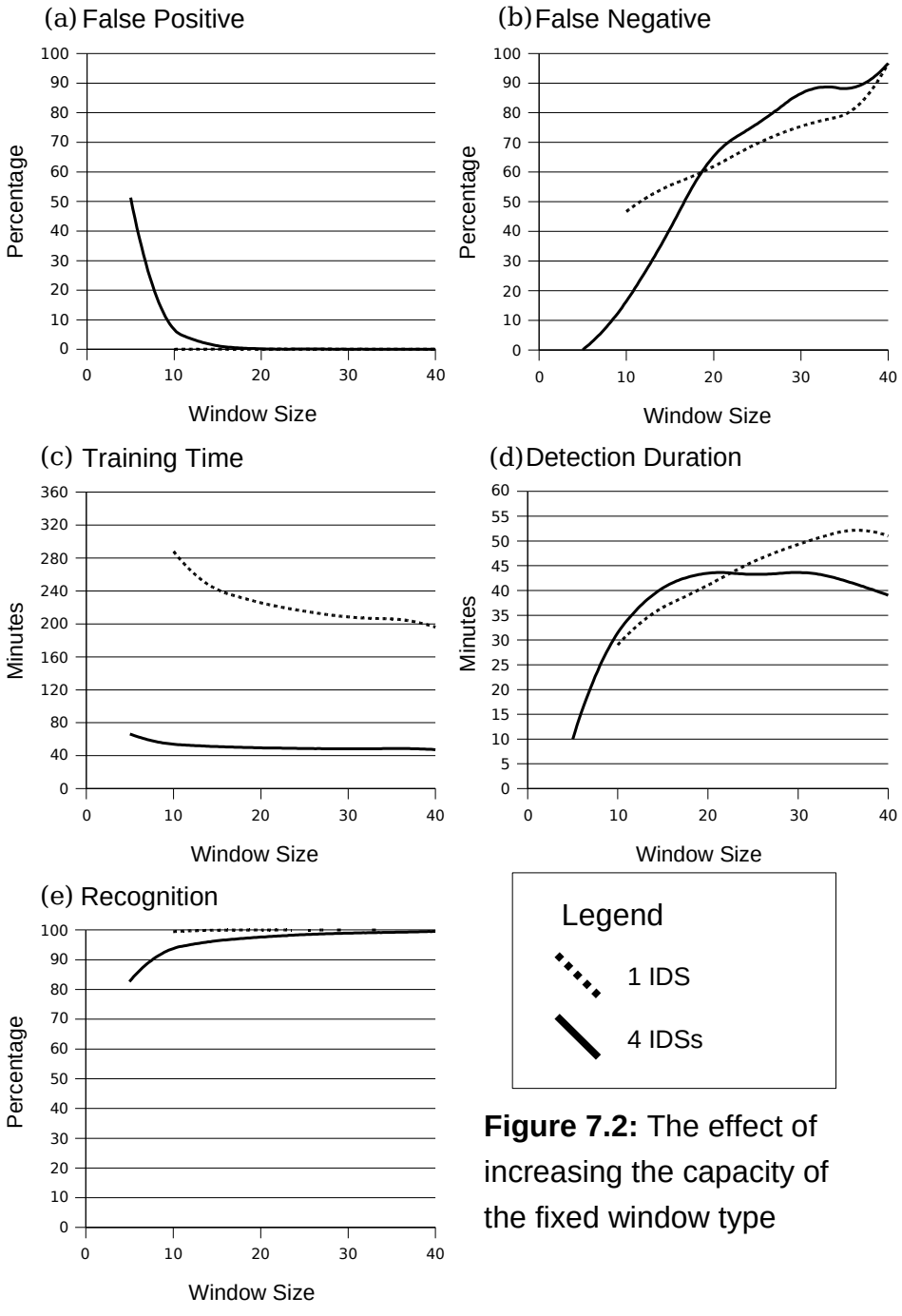

Figure 7.2: The effect of increasing the capacity of the fixed window type

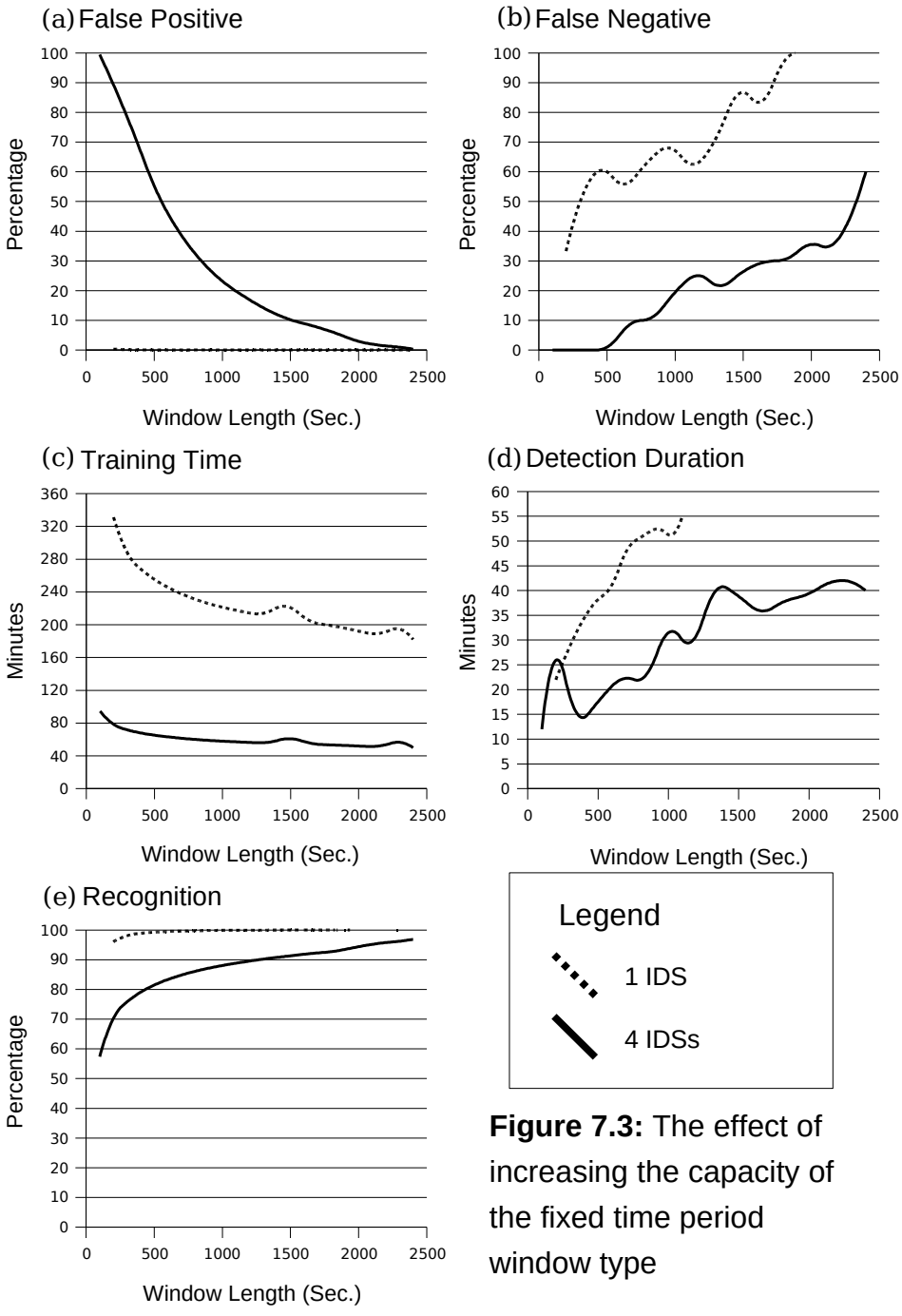

Figure 7.3: The effect of increasing the capacity of the fixed time period window type

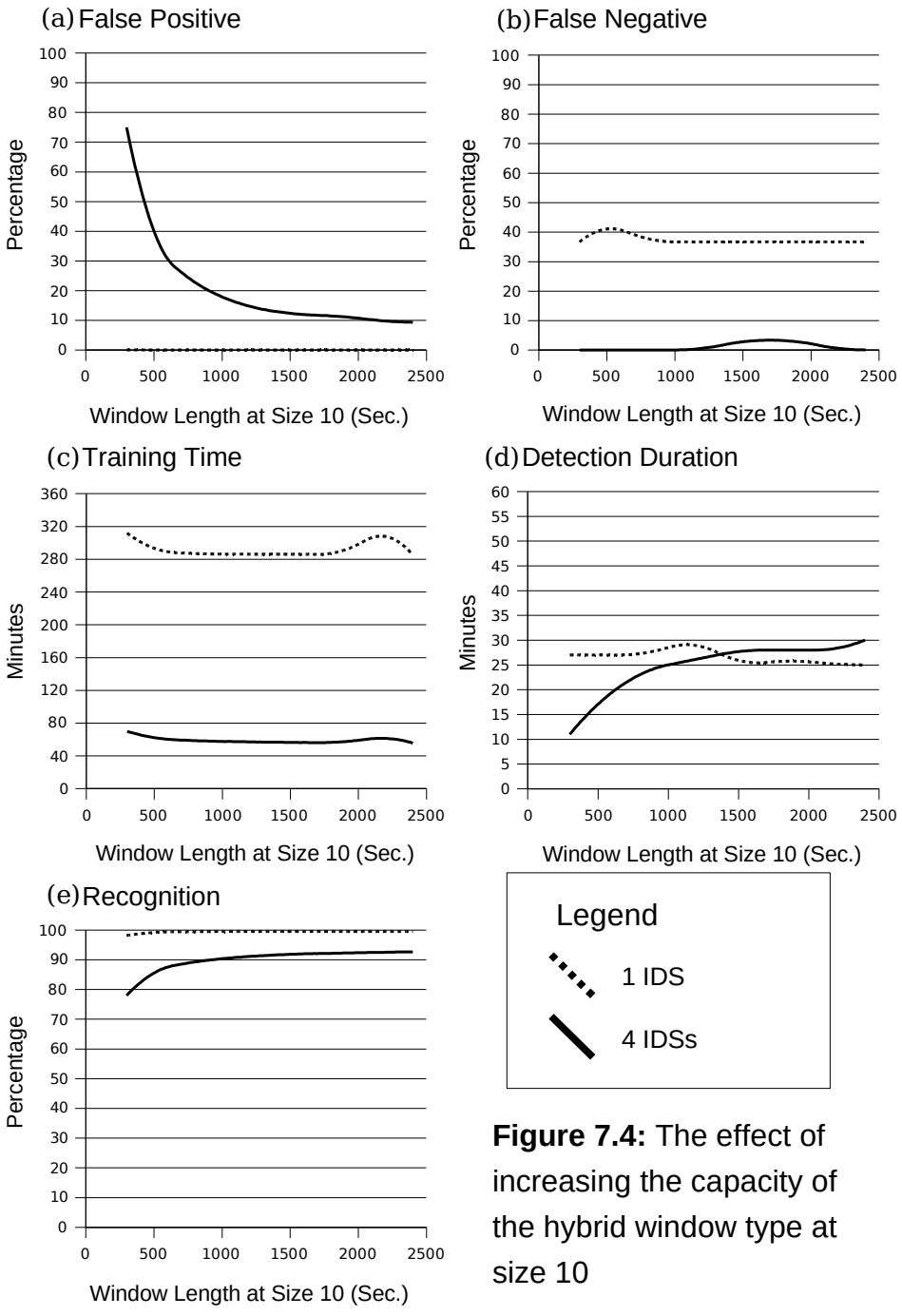

Figure 7.4: The effect of increasing the capacity of the hybrid window type at size 10

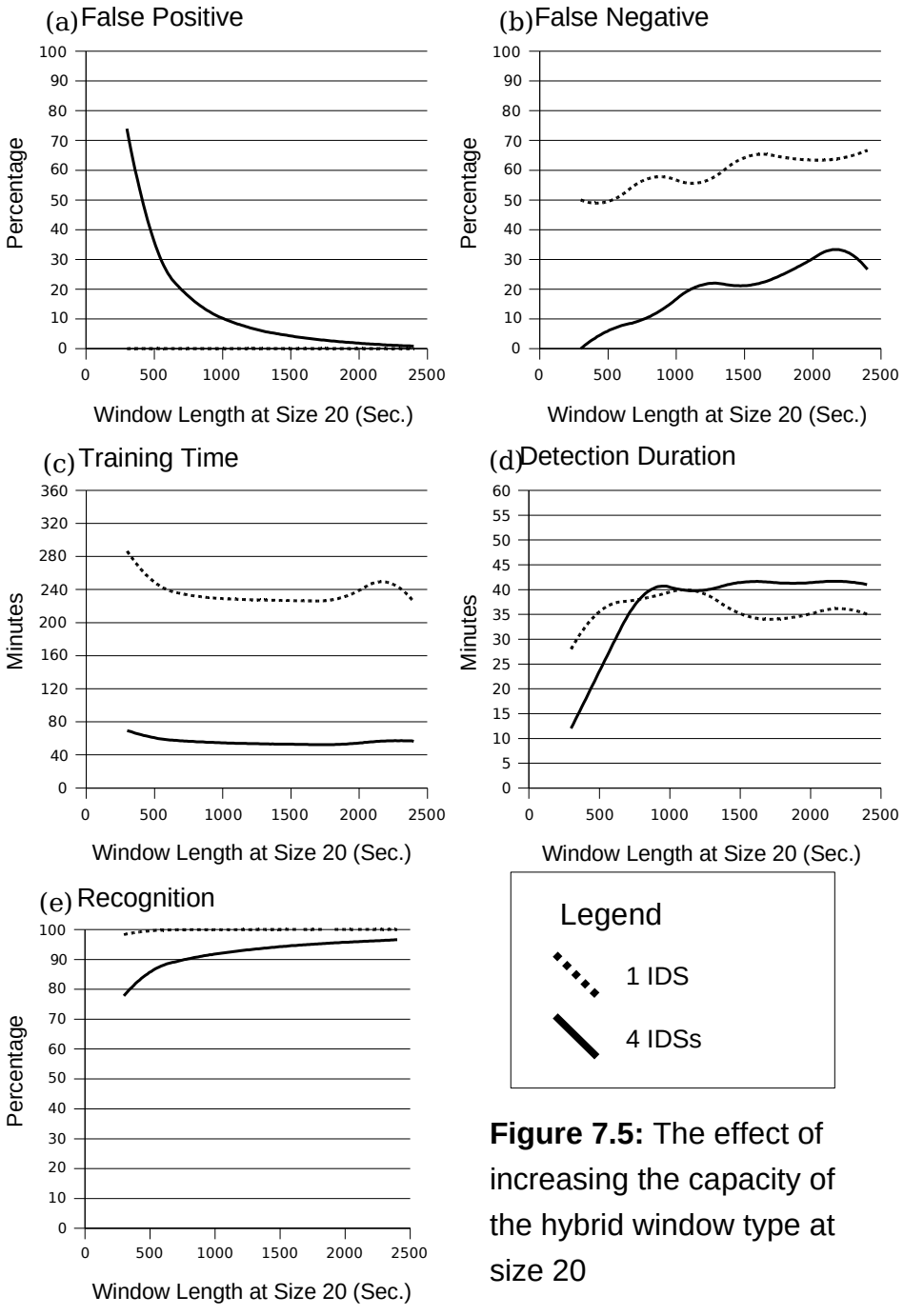

Figure 7.5: The effect of increasing the capacity of the hybrid window type at size 20

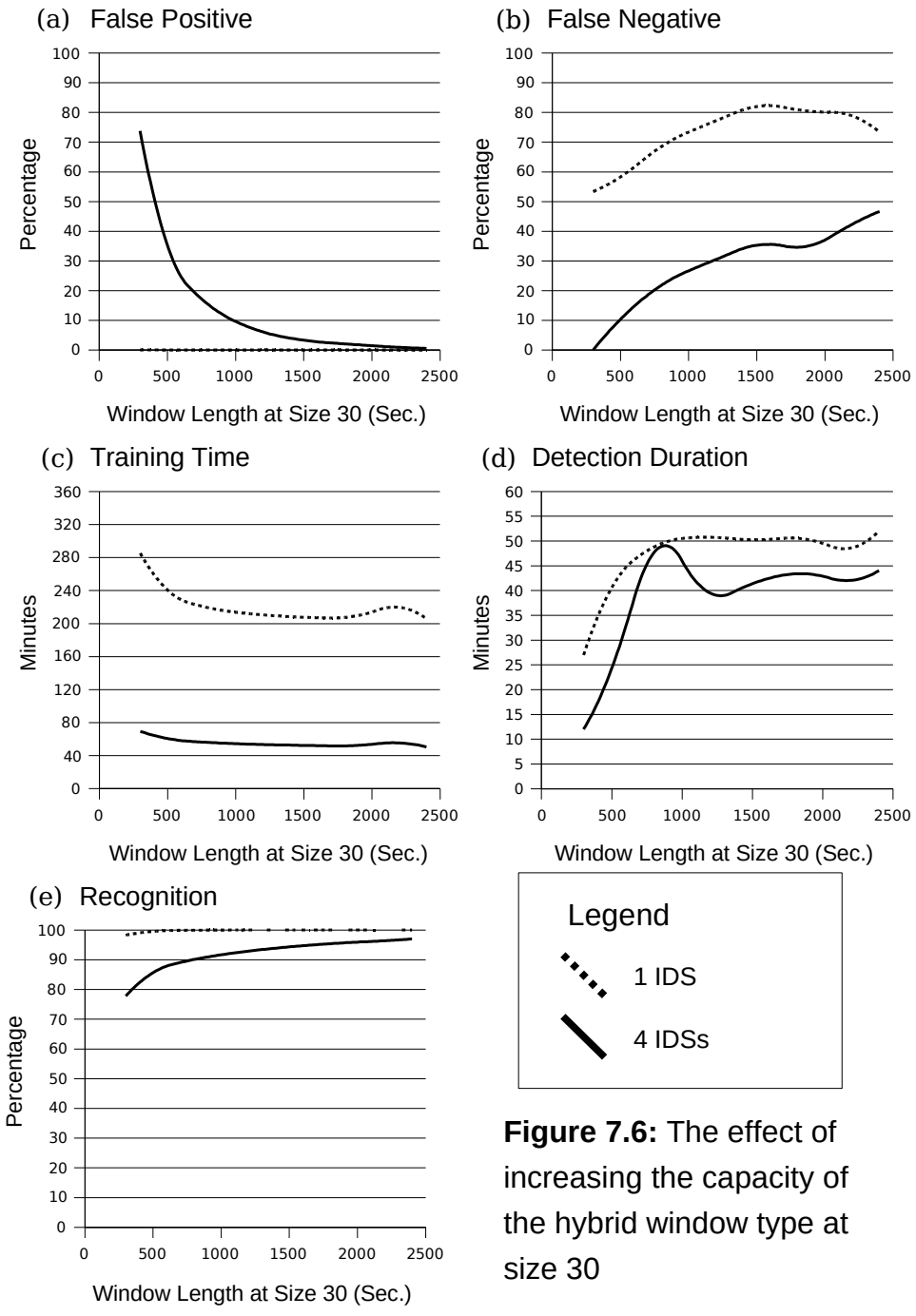

Figure 7.6: The effect of increasing the capacity of the hybrid window type at size 30

- Increasing the capacity of the window means that there will be less number of total windows or victors that will be used in training the LVQ. This resulted in reducing the training time.

- More capacity means more information about the user and better recognition. So the recognition of the LVQ increased by increasing the capacity.

The increase of window capacity had positive and negative effect on different parameters so there must be an optimization to keep these parameters at the desired value. Increasing the number of IDSs resulted in fewer records available for each IDS which means that Type 1 windows will span large period of time while Type 2 window will have small capacity compared with having only one IDS. Both of these two extremes have disadvantages as presented. The hybrid window (Type 5) gave better results because it kept the number if records in the window at the desired value even when using multiple IDSs and it also kept the detection duration stable.

For Example, increasing the number of IDSs from one IDS to four IDSs has different effect on the false negative percentage depending on the window type. The Type 1 fixed window with 10 records has decreased it from 47% to 7%. The Type 2 window with 900 seconds duration has decreased it from 70% to 20%. While the Type 5 Hybrid window with 10

records and 900 seconds duration has the best effect by reducing it from 37% to 0%. This difference shows the advantage of using the Hybrid window in a distributed environment for the data preprocessing.

7.3 Number of IDSs

This is one of the most important issues examined because it shows the scalability of the system and proves the possibility of distributing the intrusion detection problem among multiple IDSs. The experiments show the effect of increasing the number of IDSs from 1 to 24 for Grid environments having 50, 200, and 350 users (*Figure 7.7*). The following results were conducted from the experiments:

- Increasing the number of IDSs increased the percentage of false positive (*Figure 7.7.a*). This is because less information is available to each IDS about the user behavior that makes it more prone to errors. This is not desired but it is less critical than an increasing in false negative percentage, because false positive does not means intrusion but it is only annoying because legitimate users' access to resources may be denied if they are falsely detected as intruders.

- Meanwhile increasing the number of IDSs decreases the percentage of false negative (*Figure 7.7.b*), because among the few and narrow user actions monitored at an

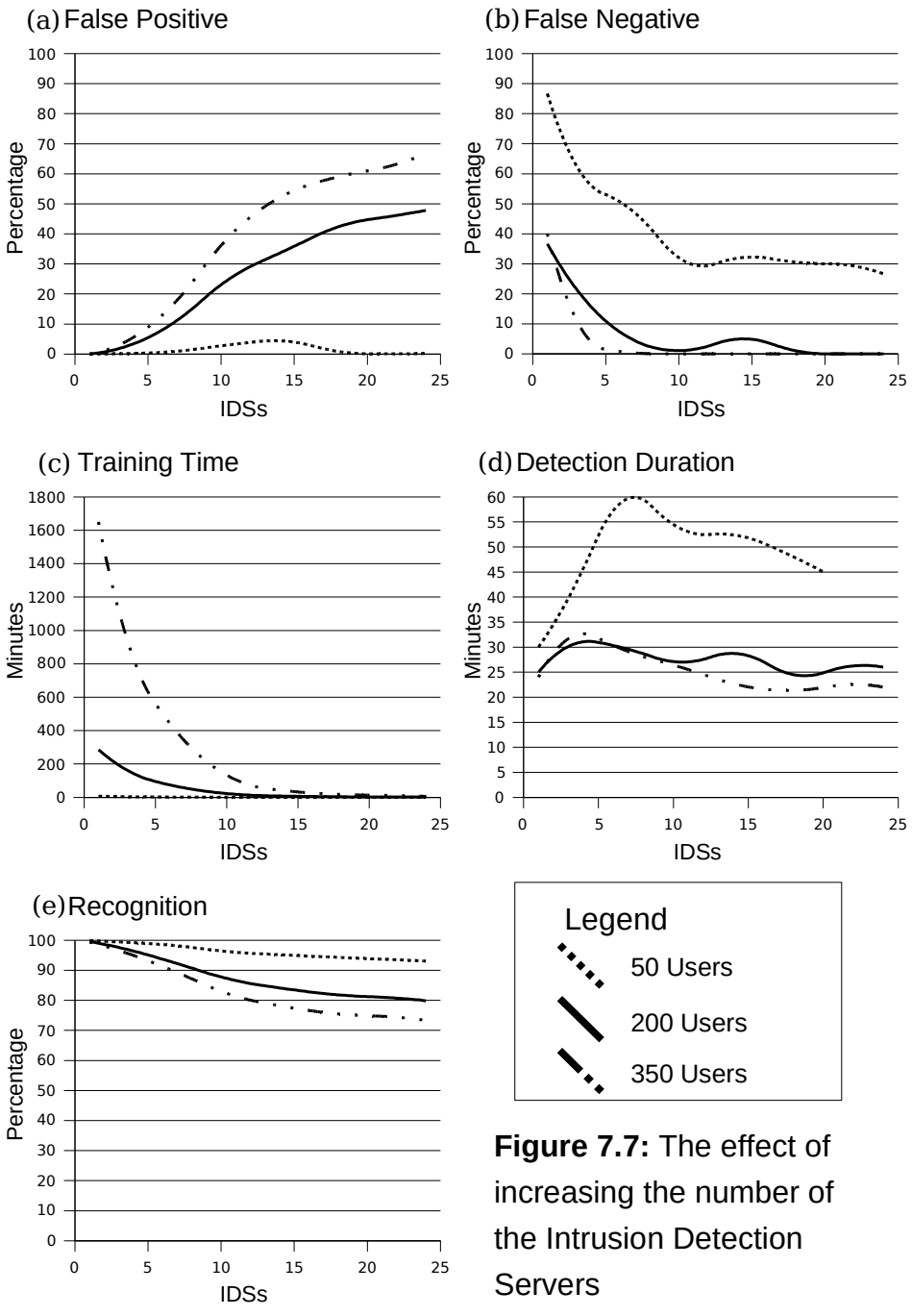

Figure 7.7: The effect of increasing the number of the Intrusion Detection Servers

IDS, detecting a deviation form them is easier than detecting deviation from many diverse actions. This is a very important issue because false negative is critical because it means that an intrusion is not detected, and in this case it gives great advantage to distributed IDSs over a centralized IDS.

- Increasing the number of IDSs has a great effect on reducing the training time (*Figure 7.7.c*). This is because dividing the training data set among multiple IDSs reduces the number of input vectors used for training in the complex LVQ algorithm. This is also an advantage of the distributed system over the centralized one and shows great scalability. The size of possible input data size is bounded by the available memory on the computer used for training. So after a certain size distributing the problem will be the only solution.

- The increase of the number of IDSs only slightly decreased the LVQ recognition percentage (*Figure 7.7.e*). This reduction is reflected in the increase of the false positive percentage. The reason for this reduction in recognition is the smaller size of data used in training. This reduction is small compared with the advantages gained in training time and false negative percentage.

- The detection duration was kept at an average of 25 minutes (*Figure 7.7.d*). This is achieved because of using

the hybrid window (Type 5) approach.

These results show that distributed intrusion detection is applicable in Grid environments and also gave better results than centralized system in all critical cases. This trade of between false positive and false negative percentages exists in all intrusion detection systems. The number of IDSs must be carefully chosen to deliver the desired values of false positive and negative percentages.

For example, in the experiments with 200 users, doubling the number of IDSs by increasing their number form two to four affected the false positive percentage by increasing its value by *8.6%* from 0.8% to 9.4% respectively. The good news is that this increase in the number of IDSs stronger effect on the false negative percentage by decreasing its value by *16.7%* from 20% to 3.3% which is more important parameter for intrusion detection. Also the training time decreased by 61% from 142 to 55 minutes. Doubling the number of IDSs form 2 to 4 has a weak effect on the detection duration and the recognition rate of the LVQ. The detection duration has only decreased by *2* minutes from 32 minutes to 30 minutes. This is because the detection duration is mostly affected by the properties of the window used as introduced in the previous section. The recognition rate has only decreased by 5.3% from 98% to 92.7%.

7.4 Number of users

This is another important issue that measures the scalability of the system in accepting larger number of users. The experiments start with 50 users up to 400 users for Grid environments with 1, 4, and 8 IDSs (*Figure 7.8*). The following was conducted from the experiments:

- Increasing the number of users slightly increased the false positive percentage (*Figure 7.8.a*). This is not desired but not very critical.

- The false negative percentage was reduced with the increase of the number of users and distribution gave even better results (*Figure 7.8.b*). This result support scalability of the system.

- Centralized systems with one IDS was not scalable as training time increased exponentially, multiple IDSs kept training time low and gave much better results (*Figure 7.8.c*) that also supports the scalability. This is because of the complexity of the LVQ algorithm which is ***O(#nodes²)*** and because more users need more nodes to keep accuracy high.

- The detection duration was kept at an average of 25 minutes (*Figure 7.8.d*). This is also achieved because of using the hybrid window (Type 5).

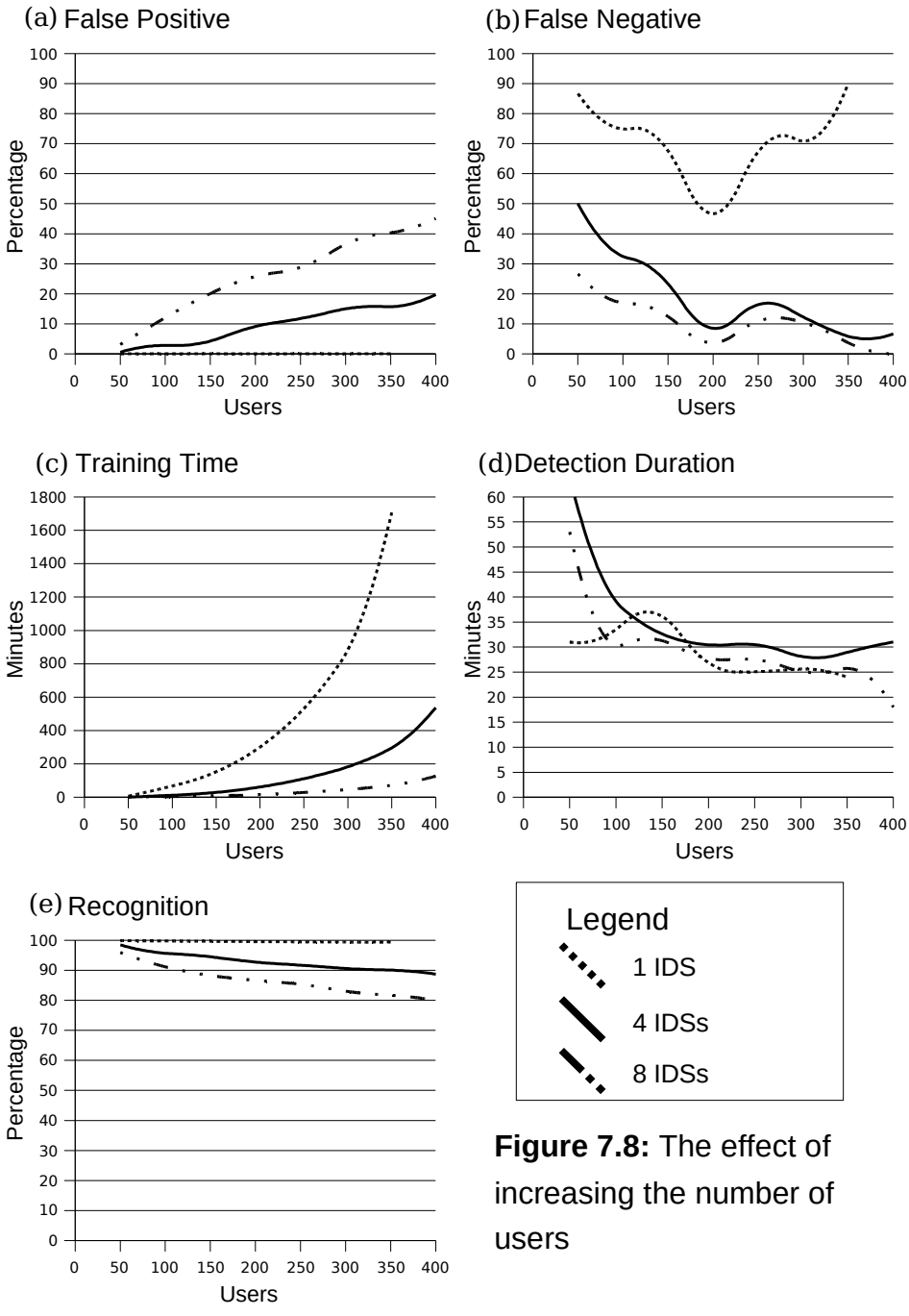

Figure 7.8: The effect of increasing the number of users

- The recognition percentage was slightly reduced with the increase of the number of users.

These results showed that the GIDA is scalable and can support a growing number of users, and also showed the limitation of centralized systems for facing growing Grid environments.

For example, in the experiments with four IDSs, doubling the number of users from 100 to 200 users increased the false positive percentage only by *3.3%* from 96% to 92.7% but fortunately meanwhile decreased the false negative percentage by *30%* from 33.3% to 3.3%. Doubling the number of users has increased the training time by *800%* from 6.9 minutes to 55.2 minutes! This shows the importance of increasing the number of IDSs in supporting scalability. The recognition percentage and the detection duration parameters were slightly affected by the doubling of the number of users. The recognition percentage decreased from 95.9% to 92.7%. The detection duration decreased from 42 minutes to 30 minutes.

7.5 Number of resources

The effect of the number of resources on the system is a bit tricky, because it indirectly affects the performance of the system unlike the number of users or number of IDSs. The following experiments study the effect of increasing the

number of resources in a Grid environment form 20 to 160 resources or administrative domains (*Figure 7.9*):

- Increasing resources reduced the false positive percentage dramatically (*Figure 7.9.a*). This is because the users have wider variety of resources to choose from to perform their tasks. This gave them better distinct behavior depending on which resource did a particular user chose. This is a great advantage and can be used to cure the increase in the false positive percentages when increasing the number of users or number of IDSs. Although this parameter is not controllable, but it is likely to naturally increase the number of resources when increasing the size of a grid environment and support the scalability.

- The increase in the number of resources has very slightly increased false negative percentage compared with the decrease in the false positive percentage (*Figure 7.9.b*).

- The training time also reflects the advantage and scalability of distributed approach over the centralized one (*Figure 7.9.c*).

- The detection duration kept stable around 25 minutes (*Figure 7.9.d*).

- The increase of the number of resources has increased the LVQ recognition percentage (*Figure 7.9.e*). This is

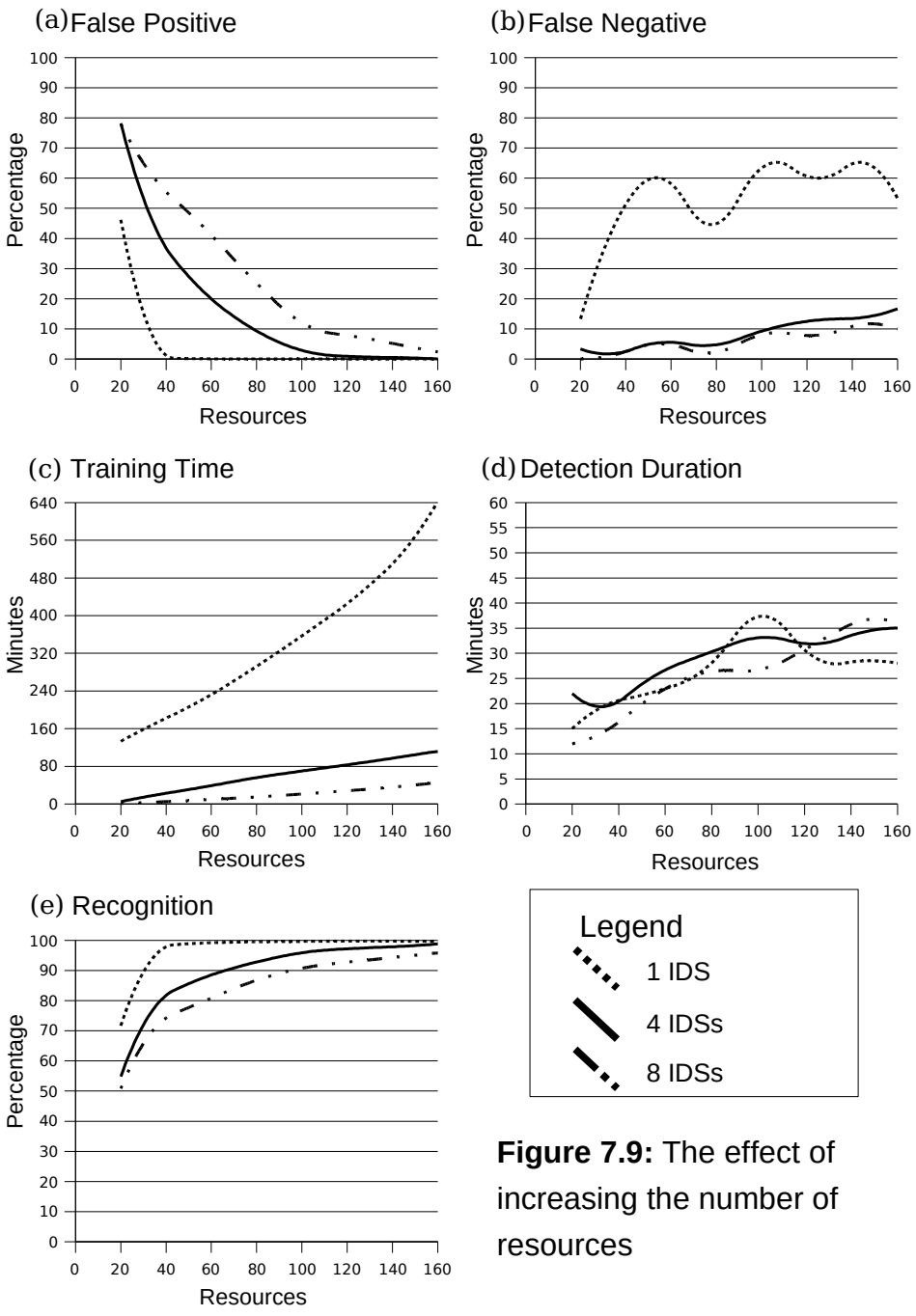

Figure 7.9: The effect of increasing the number of resources

reflected on the reduction of the false positive percentage.

The increase of the number of resources has positive effect on the system performance and can help overcome problems of increasing the number of users and/or IDSs. This is an advantage because this support scalability of GIDA to manage an increasing number of resources. Unfortunately the number of resources in not controllable and it is forced by the protected Grid environment.

For example, in the experiments with four IDSs, increasing the number of resources from 60 to 120 has a strong effect on the false positive percentage by decreasing it by *22.1%* from 22.7% to 0.6%. This is a great advantage as the increase in the false negative percentage was only from 6.7% to 13.3% and the increase in the training time was only from 34 minutes 86 minutes. The detection duration and the recognition percentage parameters were slightly affected by the doubling of the number of resources. The detection duration increased from 26 minutes to 31 minutes. The recognition percentage increased from 87.5% to 97.4%.

7.6 Number of intruders

The effect of increasing the number of intruders attacking the system was tested in the following experiments. The number of attackers was increased from 5 to 40 (*Figure 7.10*). Increasing the number of intruders only slightly

increased the percentage of the false negative (*Figure 7.10.b*). This increase did not affect the other system parameters.

For example, in the experiments with four IDSs, increasing the number of intruders from 10 to 20 has only affected the false negative percentage by increasing it from 3.3% to 11.7%. Other parameters were almost not affected. False positive percentage decreased only from 9.4% to 8%. The detection duration decreased only from 30 minutes to 28 minutes. Recognition percentage almost remained the same by increasing from 92.7% to 93%. Also the training time almost remained the same by decreasing from 55.2 minutes to 54.8 minutes.

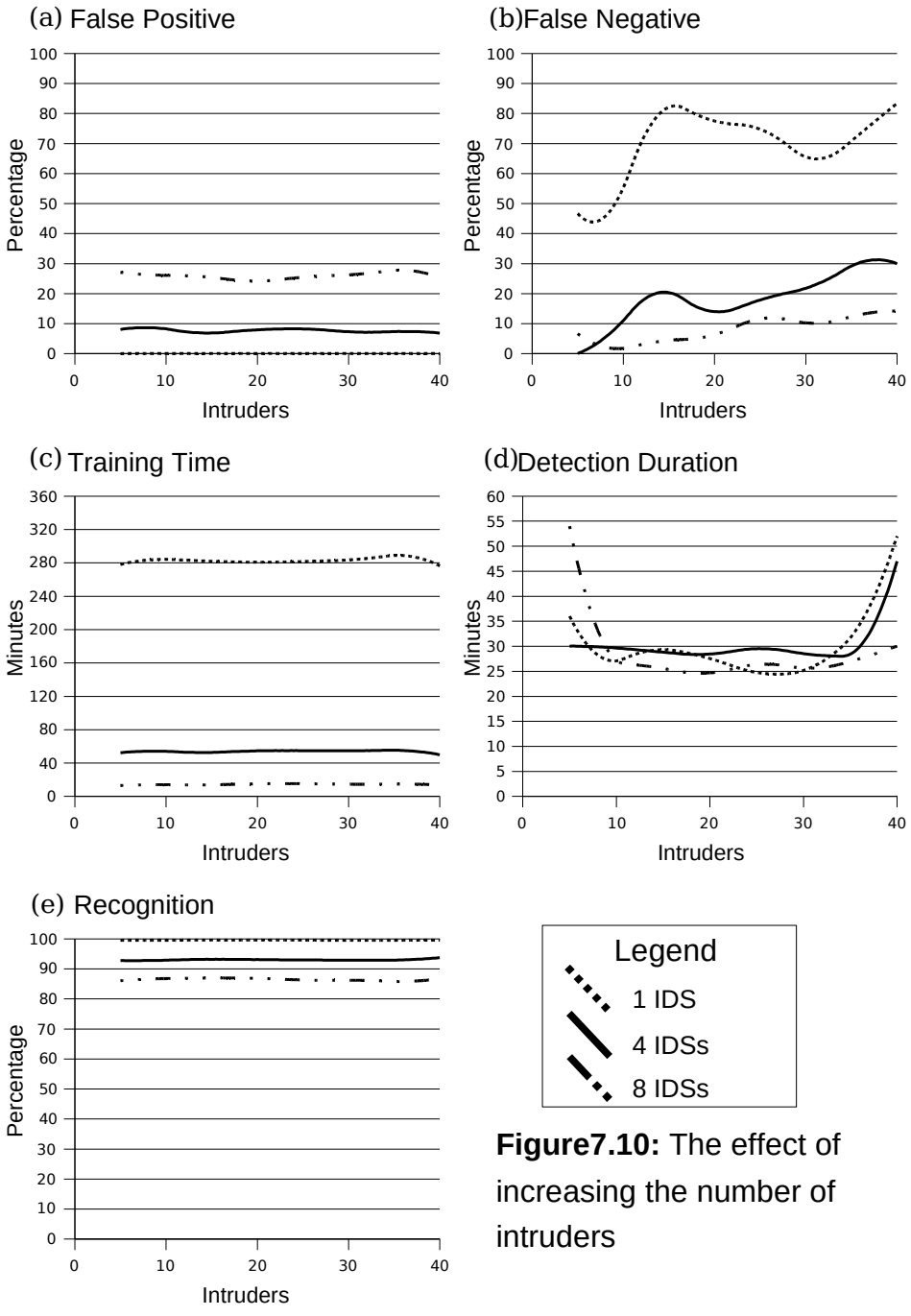

Figure7.10: The effect of increasing the number of intruders

Chapter 8

Conclusions and Future Work

Chapter 8: Conclusions and Future Work

This chapter presents the final conclusions gained from the study of intrusion detection in Grid environment through this thesis. It also suggests future directions to expand and further investigate this work.

8.1 The Grid Environment

Security is an important issue for the future of the Grid. As Grid technologies improve and real Grids start to appear, security will be more critical to protect the Grid resources in large collaborations and commercial applications. The study present in C*hapter 1* and C*hapter 2* for the Grid architecture and some Grid projects showed that security is a vital issue for the success of any Grid environment and was addressed by all studied Grid projects. This is mainly because the Grid technologies provide easy and seamless access to resources that is almost always critical and important so can attract many people who may want to misuse and abuse the system. These misusers may either be insiders or outsiders.

Unfortunately all the projects concentrated on providing basic security services such as authentication, authorization, single sign on, encryption and so on. But they did not address the possible attack of the insiders who are legitimate users but can misuse their privileges, or possible penetration from the

outsiders because of system bugs or security holes. These problems are the concerns of the intrusion detection field. The intrusion detection techniques, in spite of their importance as a second line of defense, were not applied to Grid environments or addressed by various Grid project and considered a contribution of this research.

8.2 The Grid Intrusion Detection Architecture

The study of the current intrusion detection system presented in *Chapter 4* showed that these systems are not suitable and directly applicable to Grid environments because they do not address the special characteristics and requirements of this new Grid environment. A Grid Intrusion Detection Architecture (GIDA) was presented that was designed to be suitable and applicable in Grid environments. The GIDA design was found to support heterogeneity, scalability, adaptability and multiple administrative domains which are the basic characteristics of Grid environment. GIDA is also not subject to centralized control, uses standard, open, general-purpose protocols, and can deliver nontrivial qualities of service which are the basic requirement of Grid systems.

The general conclusion in this part is that GIDA is an open, flexible, and Grid compatible architecture that can be used as a guide line to design many Grid intrusion detection systems that have various properties and performance.

8.3 The Grid Simulator

Simulation is important and has many advantages in studying new systems. There are many simulators for Grid environments that are used in research. But unfortunately these simulators were mainly designed to test resource scheduling and management not security and intrusion detection. A new simulator was implemented – inspired by available Grid simulators – to address issues related to security and intrusion detection. The simulator was used to simulate different Grid environments and generate data that was later analyzed by a prototype implementation of a simple IDS. This simulator may be used by other researchers to test their implementations of the GIDA or even in designing a new architecture.

8.4 Results Summary

The implementation of GIDA proved the applicability of such architecture in Grid environments through the results presented in *Chapter 7*. The distributed system with multiple IDSs was shown to be scalable and much better that centralized system with one IDS. The training time was sharply reduced when the problem was divided among IDSs and also the false negative percentage was reduced. The system also showed scalability to accept growing number of users. The number of resources reduced the false negative percentage. A good selection of number of IDSs and resources

in a Grid environment can help in improving Intrusion detection performance. The hybrid approach for the window, used in preprocessing, was shown to best suit different environments by keeping enough information in each window for neural network to correctly classify the users and also to keep detection duration at acceptable levels.

The main issues affecting the system have been presented to help in deciding the value of different parameters to increase the performance of the system in different Grid environments. This work also helped in better understanding the problem of intrusion detection in Grid environments and in building future systems. Also in fine tuning Grid intrusion detection systems.

8.5 Future Work

This work could be continued in many directions such as completing missing parts, improving the current system, and trying different approaches and mechanisms. To begin with, the proposed architecture itself can be improved by adding more components, details, standards, and guide lines based on experience with the current architecture to provide the best support to the designers of Grid Intrusion Detection Systems.

From the point of view of the proposed prototype implementation it is possible to try different algorithms for the

LVQ neural network and try to fine tune it with different values for its parameters. It is also suggested to try other neural networks and even other approaches than neural networks. After this step a heterogeneous system should be created with different techniques of intrusion detection used in the IDSs. After the application of the Grid techniques in real life problem it is suggested to create a knowledge base with signatures of known Grid attacks to enable the use of misuse intrusion detection along with the anomaly intrusion detection technique presented in this work to create a complete intrusion detection system.

Study of the effect of different trust relationships between participants must be studied to understand their effect on the system. Also the overlapping of the scopes of different IDSs will affect the system performance and reliability and should be analyzed.

The cooperation protocol must also be revisited and improved after the use of heterogeneous IDSs, complex trust relationships, and IDSs scope overlapping. This is because these issues increase the complexity of the system and thus the complexity of the cooperation between the IDSs. Also these two issues will raise a question about their effect on different QoSs and how these QoSs can be selected and measured.

The simulator presented in this work is also subject to improvements to support other problems than security to

increase its usability. Trust relationships should be added and its performance enhanced and evaluated.

Finally these systems should be implemented and tested on real grids because this is the only way to prove their success in protecting Grid environments and increasing the security level.

Published Work

Published Work

[1] M. Tolba, I. Taha, and A. Al-Shishtawy, **"An Intrusion Detection Architecture for Computational Grids"**. First International Conference on Intelligent Computing and Information Systems, June 2002.

[2] M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy, **"A Secure Grid Enabled Signature Verification System"**. Second International Conference on Intelligent Computing and Information Systems, Cairo, Egypt, March 2005.

[3] M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy, **"Distributed Intrusion Detection System for Computational Grids"**. Second International Conference on Intelligent Computing and Information Systems, Cairo, Egypt, March 2005.

[4] M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy, **"GIDA: Toward Enabling Grid Intrusion Detection Systems"**. Cluster Computing and Grid 2005, Cardiff, UK, 9 - 12 May 2005.

http://dsg.port.ac.uk/events/conferences/ccgrid05/wip/schedule/Paper20.pdf

[5] M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy, "**Intrusion Detection System for the Grid"**. The 2005 International Conference on Grid Computing and Applications (GCA'05). Las Vegas, Nevada, USA, 20 - 23 June 2005.

References

References

[1] A. Grimshaw, and W. Wulf, **The Legion Vision of a Worldwide Virtual Computer.** Communications of the ACM 1997; 40(1)

[2] A. Kosoresow, and S. Hofmeyr, **Intrusion Detection via System Call Traces.** IEEE Software, vol. 14, pp. 24-42, 1997.

[3] A. LaVigna, **Nonparametric Classification Using Learning Vector Quantization.** Ph. D. Thesis, University of Maryland, 1989.

[4] A. Law, and W. Kelton, **Simulation Modeling and Analysis.** 3rd Edition, Mc. Graw Hill, 2000

[5] A. Oram, (ed.), **Peer-to-Peer: Harnessing the Power of Disruptive Technologies.** O'Reilly, 2001

[6] A. Seleznyov, V. Terziyan, and S. Puuronen, **Temporal-Probabilistic Network Approach for Anomaly Intrusion Detection.** 12th Annual Computer Security Incident Handling Conference, Chicago, USA, 2000.

[7] B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, **Secure, Efficient Data**

Transport and Replica Management for High-Performance Data-Intensive Computing. IEEE Mass Storage Conference, San Diego, CA, USA, 2001.

[8] B. C. Neuman and T. Ts'o, **Kerberos: An Authentication Service for Computer Networks.** IEEE Communications Magazine, 32(9):33-28, September 1994.

[9] B. Rhodes, J. Mahaffey, and J.Cannady, **Multiple Self-Organizing Maps for Intrusion Detection.** Presented at Proceedings of the 23rd National Information Systems Security Conference, Baltimore, MD, 2000.

[10] B. Segal, **Grid Computing: The European Data Grid Project.** IEEE Nuclear Science Symposium and Medical Imaging Conference, Lyon, France, October 2000.

[11] C. Shirky, **What Is P2P... And What Isn't.** http://www.openp2p.com/pub/a/p2p/2000/11/24/shirky1-whatisp2p.html

[12] **ChicSim: The Chicago Grid Simulator.** http://people.cs.uchicago.edu/~krangana/ChicSim.html/

[13] Computer Science and National Research Council Telecommunications Board. **Realizing the Information Future. The Internet and Beyond.** National Academy

Press, 1994. 19.

http://www.nap.edu/readingroom/books/rtif/.

[14] D. Anderson, S. Bowyer, J. Cobb, D. Gedye, W. T. Sullivan, and D. Werthimer, **A New Major Seti Project Based on Project Serendip Data and 100,000 Personal Computers.** In Astronomical and Biochemical Origins and the Search for Life in the Universe, Proc. of the Fifth Intl. Conf. on Bioastronomy, 1997.

[15] D. Anderson, T. F. Lunt, H. Javitz, A. Tamura, and A. Valdes, **Detecting Unusual Program Behavior Using the Statistical Components of NIDES.** SRI International, Menlo Park, CA, Tech Report SRI-CSL-95-06, May 1995.

[16] D. Brown, B. Suckow, and T. Wang, **A Survey of Intrusion Detection Systems.** CSE 221: Fall 2001 Projects, Department of Computer Science, University of California,San Diego, USA, 2001.

[17] D. Denning, **An Intrusion-Detection Model.** IEEE Transactions on Software Engineering, vol. 13, pp. 222-232, 1987.

[18] D. Erwin, **Unicore Plus Final Report.** (2003) http://www.unicore.org/forum/documents.htm.

[19] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, **Peer-to-peer computing.** Technical Report HPL-2002-57, HP Lab, 2002.

[20] E. Adar, and B. Huberman, **Free Riding on Gnutella.** First Monday, 5 (10). 2000.

[21] E. Spafford, and D. Zamboni, **Data collection mechanisms for intrusion detection systems.** CERIAS Technical Report 2000-08, CERIAS, Purdue University, 1315 Recitation Building, West Lafayette, IN, June 2000.

[22] G. Helmer, J. Wong, A. Vasant Honavar, and L. Mille, **Intelligent Agents for Intrusion Detection and Countermeasures.** Presented at IEEE Information Technology Conference, Syracuse, NY, 1998.

[23] G. Pfister, **In Search of Clusters.** Prentice Hall PTR, ISBN: 0138997098; 2nd edition, January 1998.

[24] **Global Grid Forum Home Page.** 2002. http://www.gridforum.org/

[25] H. Casanova, **Simgrid: A Toolkit for the Simulation of Application Scheduling.** Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18,

2001, Brisbane, Australia.

[26] H. Debar, **An Introduction to Intrusion-Detection Systems.** IBM Research, Zurich Research Laboratory, Ruschlikon, Switzerland, 2000.

[27] H. Javitz, and A. Valdes, **The SRI IDES Statistical Anomaly Detector.** Presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1991.

[28] H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, **The MicroGrid: a Scientific Tool for Modeling Computational Grids.** Proceedings of IEEE Supercomputing (SC 2000), Nov. 4-10, 2000, Dallas, USA.

[29] H. Teng, K. Chen, and S. C. Lu, **Adaptive Realtime Anomaly Detection Using Inductively Generated Sequential Patterns.** Presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Los Alamitos, CA, 1990.

[30] H. Vaccaro, and G. Liepins, **Detection of Anomalous Computer Session Activity.** Presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Location TBD, 1989

[31] I. Foster, **Grid Today.** Daily News and Information for the Global Grid Community. July 22, 2002: VOL. 1 NO. 6.

http://news.tgc.com/msgget.jsp?mid=286185&xsl=story.xsl

[32] I. Foster, **The Grid: A New Infrastructure for 21st Century Science.** Physics Today, 55 (2). 42-47. 2002.

[33] I. Foster, and A. Iamnitchi, **On Death, Taxes and the Convergence of Peer-to-Peer and Grid Computing.** http://people.cs.uchicago.edu/ anda/papers/foster grid vs p2p.pdf, 2002.

[34] I. Foster, and C. Kesselman (Eds), **The Grid: Blueprint for a New Computing Infrastructure.** Morgan Kaufmann, 1999.

[35] I. Foster, and C. Kesselman, **The Globus Project: A Status Report.** Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop, pp. 4-18, 1998.

[36] I. Foster, C. Kesselman, and S. Tuecke, **The Anatomy of the Grid.** Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001.

[37] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, **A**

security architecture for computational grids. In Fifth ACM Conference on Computers and Communications Security, November 1998, 83-91.

[38] I. Foster, C. Kesselman, J. Nick and S. Tuecke, **The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.** Globus Project, 2002. www.globus.org/research/papers/ogsa.pdf.

[39] Internet Security Systems, **Network- vs. Host-based Intrusion Detection: A Guide to Intrusion Detection Technology.** Technical Whitepaper. http://documents.iss.net/whitepapers/nvh_ids.pdf/ (2003).

[40] J. Almond, and D. Snelling, **UNICORE: Uniform access to supercomputing as an element of electronic commerce.** Future Generation Computer Systems 1999, 15:539 548.

[41] J. Balasubramaniyan, J. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, **An Architecture for Intrusion Detection using Autonomous Agents.** Department of Computer Sciences, Purdue University, Coast TR 98-05; 1998.

[42] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S.

Tuecke, **GASS: A Data Movement and Access Service for Wide Area Computing Systems.** Sixth Workshop on I/O in Parallel and Distributed Systems, Atlanta, GA, USA, May 5, 1999.

[43] J. C. Schlimmer, **Concept Acquisition Through Representational Adjustment.** In Department of Information and Computer Science. Irvine: University of California, 1987

[44] J. Marin, D. Ragsdale, and J. Surdu, **A Hybrid Approach to Profile Creation and Intrusion Detection.** in Proceedings of DARPA Information Survivability Conference and Exposition, Anaheim, CA, 12-14 June 2001.

[45] J. Novotny, S. Tuecke, V. Welch, **An Online Credential Repository for the Grid: MyProxy.** 10th IEEE Symp. On High Performance Distributed Computing, 2001.

[46] J. Ryan, M. Lin, and R. Miikkulainen, **Intrusion Detection with Neural Networks. AI Approaches to Fraud Detection and Risk Management.** Papers from the 1997 AAAI Workshop (Providence, Rhode Island), pp. 72-79. Menlo Park, CA: AAAI.

[47] J. Ryan, M. Lin, and R. Miikkulainen, **Intrusion**

Detection with Neural Networks. Presented at Proceedings of the 10th Advances in Neural Information Processing Systems Conference, Denver, CO, 1998.

[48] **JXTA Home Page.** http://www.jxta.org/

[49] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, and U. Nagashima, **Performance Evaluation Model for Scheduling in a Global Computing System.** The International Journal of High Performance Computing Applications, vol. 14, No. 3, pp. 268-279, 2000.

[50] K. Czajkowski, I. Foster, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, **A resource management architecture for metacomputing systems.** Technical report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1997.

[51] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, **Grid Information Services for Distributed Resource Sharing.** Proc. 10 th IEEE Symp. On High Performance Distributed Computing, 2001.

[52] L. Lankewicz, and M. Benard, **Real-time Anomaly Detection Using a Nonparametric Pattern Recognition Approach.** Presented at Proceedings of the of 7th Computer Security Applications conf., San Antonio, TX,

1991

[53] M. Baker, R. Buyya, and D. Laforenza, **Grids and Grid technologies for wide-area distributed computing.** Software Practice and Experience, 2002

[54] M. Baker, R. Buyya, and D. Laforenza, **The Grid: International Efforts in Global Computing.** Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR'2000), Italy, 2000.

[55] M. Biswanath, T. Heberlein, and K. Levitt, **Network Intrusion Detection.** IEEE Network, 8, PP. 26-41, May/June, 1994.

[56] M. Chetty, and R. Buyya, **Weaving Computational Grids: How Analogous are they with Electrical Grids?** Journal of Computing in Science and Engineering (CiSE) 2001; (July-August).

[57] M. Handley, V. Paxson, and C. Kreibich, **Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics.** 10th USENIX Security Symposium, Washington, D.C., 13-17 August 2001.

[58] M. Huang, and T. Wicks, **A Large-scale Distributed**

Intrusion Detection Framework Based on Attack Strategy Analysis. Web proceedings of the First International Workshop on Recent Advances in Intrusion Detection (RAID'98).

[59] M. Murshed, R. Buyya, and D. Abramson, **GridSim: A Grid Simulation Toolkit for Resource Management and Scheduling in Large-Scale Grid Computing Environments.** 17th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2002), April 15-19, 2002, Fort Lauderdale, FL, USA.

[60] M. Thompson, A. Essiari, S. Mudumbai, **Certificate-based Authorization Policy in a PKI Environment.** ACM Transactions on Infomation and System Security (TISSEC), Volume 6, Issue 4, pp: 566-588, November 2003.

[61] M. Thottan, and C. Ji, **Proactive Anomaly Detection Using Distributed Intelligent Agents.** IEEE Network, vol. 12, pp. 21-27, 1998.

[62] M. Tolba, I. Taha, and A. Al-Shishtawy, **An Intrusion Detection Architecture for Computational Grids.** First International Conference on Intelligent Computing and Information Systems, June 2002.

[63] M. Tolba, I. Taha, M. Al Shandawely, **Building a Grid-Enabled Distributed System to Solve Signature Verification Problem Based on Improving QoS.** *Second International Conference on Intelligent Computing and Information Systems*, Cairo, Egypt, March 2005.

[64] M. Tolba, M. Abdel-Wahab, I. Taha, and A. Al-Shishtawy, **Distributed Intrusion Detection System for Computational Grids.** Second International Conference on Intelligent Computing and Information Systems, March 2005.

[65] M. Tolba, M. Abdel-Wahab, I. Taha, A. Anbar, **Fault Tolerant Scheduling for Grid Enabled Signature Verification System.** *Second International Conference on Intelligent Computing and Information Systems*, Cairo, Egypt, March 2005.

[66] N. Habra, B. L. Charlier, A. Mounji, and I. Mathieu, **ASAX: Software Architecture and Rule-based Language for Universal Audit Trail Analysis.** Presented at Proceedings of the European Symposium on Research in Computer Security, Brighton, England, 1992.

[67] **Napster Home Page.** http://www.napster.com/, 2001.

[68] P. Anderson, **Computer Security Threat Monitoring and Surveillance.** Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, April 1980.

[69] P. Helman, and G. Liepins, **Statistical Foundations of Audit Trail Analysis for the Detection of Computer Misuse.** IEEE Transactions on Software Engineering, vol. 19, pp. 886-901, 1993.

[70] P. Proctor. **Practical Intrusion Detection Handbook.** Prentice Hall PTR. 1st edition August-2000

[71] R. Bace, and P. Mell, **NIST Special Publication on Intrusion Detection Systems.** 16 August 2001.

[72] R. Bace, **Intrusion Detection.** MacMillan Technical Publishing 2000.

[73] R. Butler, D. Engert, I. Foster, C. Kesselman, S. Tuecke, J. Volmer, and V. Welch, **A National-Scale Authentication Infrastructure.** IEEE Computer, Vol 33, No 12, pp 60-66, 2000.

[74] R. Buyya, **The Gridbus Toolkit: Enabling Grid computing and business.** http://www.gridbus.org.

[75] R. Buyya and S. Venugopal, **The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An**

Overview and Status Report. Proceedings of the First IEEE International Workshop on Grid Economics and Business Models (GECON 2004, April 23, 2004, Seoul, Korea), 19-36pp, ISBN 0-7803-8525-X, IEEE Press, New Jersey, USA

[76] R. Buyya, K. Branson, J. Giddy, and D. Abramson, **The Virtual Laboratory: Enabling On-Demand Drug Design with the World Wide Grid.** Proceedings of the IEEE International Symposium on Cluster Computing and the Grid, May 21-24, 2002.

[77] R. Gopalakrishna, **A framework for distributed intrusion detection using interest driven cooperating agents.** Paper for Qualifier II examination, Department of Computer Sciences, Purdue University, May 2001.

[78] R. Stevens, P. Woodward, T. DeFanti, and C. Catlett, **From the I-WAY to the National Technology Grid.** Communications of the ACM, 40(11):51--60, November 1997.

[79] S. Badr, **Security Architecture for Internet Protocols.** Ph.D. dissertation, Military Technical Collage, Cairo, Egypt, 2002.

[80] S. C. Lee, and D. V. Heinbuch, **Training a Neuralnetwork Based Intrusion Detector to Recognize**

Novel Attacks. Presented at IEEE Workshop Information Assurance and Security, West Point, NY, 2000.

[81] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, **A Directory Service for Configuring High-Performance Distributed Computations.** Proc. 6th IEEE Symposium on High-Performance Distributed Computing, Portland, OR, US, 1997

[82] S. Forrest, S. A. Hofmeyr, and A. Somayaji, **Computer Immunology.** Communications of the ACM, vol. 40, pp. 88-96, 1997.

[83] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, **A Sense of Self for Unix Processes.** Presented at Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, CA, 1996.

[84] T. Howes, M. Smith, and G. Good, **Understanding and deploying LDAP Directory services.** MTP edition 12 Wahl, M., Howes, T, Kill, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1998.

[85] T. Kohonen, **Learning Vector Quantization.** In M. Arbib, editor, The Handbook of Brain Theory and Neural Networks. pages 537--540. MIT Press, 1995.

[86] T. Kohonen, **Learning Vector Quantization for Pattern Recognition.** Technical Report, TKK-F-A601, University of Technology, Helsinki, 1986.

[87] T. Kohonen, **Self-Organization and Associative Memory.** Springer-Verlag, Berlin, 1987.

[88] T. Kohonen, **The Self-Organizing Map.** in Lau, C. (ed.) Neural Networks: Theoretical Foundations and Analysis. IEEE Press, NY, 1992.

[89] T. Kohonen, G. Barna, and R. Chrisley, **Statistical Pattern Recognition with Neural Networks: Benchmarking Studies.** IEEE International Conference on Neural Networks, San Diego, CA, pp. 61-68, 1988.

[90] T. Kohonen, J. Hynninen, J. Kangas, K. Laaksonen, and J. Torkkola. Lvq pak, **The Learning Vector Quantization Program Package.** http://www.cis.hut.fi/research/lvq_pak, 1995.

[91] T. Lane, and C. E. Brodley, **Temporal Sequence Learning and Data Reduction for Anomaly Detection.** ACM Transactions on Information and System Security, vol. 2, pp. 295-331, 1999.

[92] T. Y. Lin, **Anomaly Detection - A Soft Computing Approach.** Presented at New Security Paradigms

Workshop, Little Compton, Rhode Island, 1994.

[93] **The Globus Project™ Home Page.** http://www.globus.org/

[94] **The GridBus Home Page.** http://www.gridbus.org/

[95] **The Legion Home Page.** http://legion.virginia.edu/

[96] **The UNICORE Home Page.** http://www.unicore.org/

[97] V. Paxson, **Bro: A System for Detecting Network Intruders in Real-Time.** Computer Networks, 31, pp. 2435-2463, Dec. 1999.

[98] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, **The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets.** The Journal of Network and Computer Applications, 2001.

[99] W. Johnston, D. Gannon, and B. Nitzberg, **Information Power Grid Implementation Plan: Research, Development, and Testbeds for High Performance, Widely Distributed Collaborative, Computing and Information Systems Supporting Science and Engineering.** Technical report, NASA Ames Research Center, http://www.nas.nasa.gov/IPG, 1999.

[100]W. Lee, and S. J. Stolfo, **A Framework for Constructing Features and Models for Intrusion Detection Systems.** ACM Transactions on Information and System Security, Vol. 3, November, 2000

[101]W. Lee, R. Nimbalkar, K. Yee, S.Patil, P. Desai, T. Tran, and S. Stolfo, **A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions.** in Recent Advances in Intrusion Detection (RAID 2000), Third International Workshop, Toulouse, France, October 2-4, 2000, vol. Vol. 1907, H. Debar, L. Mé, and S. F. Wu, Eds. Berlin: Springer-Verlag, 2000, pp. 49-65.

[102]W. Roush, A. Goho, E. Scigliano, D. Talbot, M. Waldrop, G. Huang, P. Fairley, E. Jonietz, and H. Brody, **10 Emerging Technologies That Will Change The World.** Technology Review, MIT, 106:33--49, Feb 2003.

[103]W. Stallings, **Network and Internetwork Security - Principles and Practice.** Prentice Hall, 1995.

ملخص الرسالة

يعتبر توفير القدرة الحسابية الكافية لحل المشاكل المختلفة بقدرة وكفاءة عالية هي الهدف الأساسي للعاملين في أي مجال يتطلب العمل بدقة وتوفير الوقت و المال. ظهر مجال البيئات الحسابية الشبكية ليسد الفجوة الموجودة بين التكنولوجيا المتاحة و الطلب المتزايد للقدرة الحسابية. توفر الشبكة الحسابية بيئة حسابية قوية وذلك عن طريق ربط الموارد الموزعة لتمكين التجميع و المشاركة بسهولة وذلك لخلق مورد حسابي أكثر قوة. والجدير بالذكر أن مصطلح الموزع هنا لا يشير فقط إلى الأماكن الجغرافية بل أيضا إلى الإدارة التي قد تغطي المنظمات المتعددة.

إلى جانب أهمية القضايا الأمنية المختلفة التي طُرقت مع بداية مجال البيئات الحسابية الشبكية يعتبر كشف التطفل من أهم العناصر لأي نظام أمني حديث لأنه يُعتبر خط دفاع ثاني ضد الثغرات الأمنية و أيضا ضد المستخدمين الذين يسيؤون إستخدام حقوقهم.

تتعرض هذه الرسالة لدراسة مشكلة كشف التطفل في البيئات الحسابية الشبكية باعتبارها احد القضايا الأمنية المهمة. يقدم البحث طرازا مرنا مبنيا على التعاون و توزيع العمل لكشف التطفل في البيئات الحسابية الشبكية. هذا العمل مبني على أساس دراسة مشاريع البيئات الحسابية الشبكية و نظم كشف التطفل الحالية لتصميم طراز يتناسب مع البيئة الحسابية الشبكية المتاحة و يستفيد منها.

وقد تم تنفيذ نموذج للطراز المقترح من أجل الإجازة و المراجعة

لاكتساب معلومات أكثر و خبرة في حل مشكلة كشف التطفل في سياق البيئات الحسابية الشبكية. يعمل النموذج المقترح بطريقة كشف تطفل موزعة و متجانسة و التي تستخدم الشبكات العصبية المعروفة بأسم (Learning Vector Quantization) للتصنيف وذلك لاكتشاف التطفل إذا حدث.

قد تم استخدام نظم النمذجة و المحاكاة لفحص هذا النموذج في عدة بيئات حسابية شبكية ذات تنظيمات و طرازات مختلفة من خلال محاكاة هذة البيئات باستخدام برنامج لمحاكاة البيئة الشبكية. وقد تم تطوير برامج المحاكاة هذه لتتناسب مع دراسة الأمن و كشف التطفل. اظهرت النتائج إمكانية تطبيق النّظام المقترح في البيئات الحسابية الشبكية و متفوقا علي نظم مركزية غير موزعة. وقد تم أيضا دراسة العناصر المختلفة التي قد تؤثر على أداء نظام كشف التطفل المقترح. فقد أدت زيادة عدد المستخدمين و أيضا عدد خادمات كشف التطفل إلى نقصان نسبة الخطأ السلبي (False Negative Percentage) الذي يعتبر من أهم عناصر تقييم جودة أنظمة التطفل. وقد دعمت هذه النتائج إثبات إمكانية تطبيق النظام في البيئات الحسابية الشبكية. ومما زاد هذا الدعم أن زيادة عدد الموارد، وهو طبيعي مع زيادة حجم البيئة الحسابية الشبكية ، أدي إلى نقصان نسبة الخطأ الإيجابي (False Positive Percentage). وقد أدى ذلك إلى التغلب على زيادة طفيفة طرأت على هذه النسبة بسبب زيادة عدد المستخدمين و خادمات كشف التطفل.

جامعة عين شمس
كلية الحاسبات والمعلومات
قسم الحسابات العلمية

مشكلة السرية في بيئة حسابية شبكية ديناميكية مفتوحة وغير متجانسة

رسالة مقدمة لإستيفاء الحصول علي درجة الماجستير
في الحاسبات والمعلومات

إعداد

أحمد محمد عبد الغفور الششتاوي

بكالوريوس الحاسبات والمعلومات
معيد بقسم الحسابات العلمية
جامعة عين شمس

تحت اشراف

الأستاذ الدكتور/ محمد فهمي طلبة

أستاذ بقسم الحسابات العلمية
نائب رئيس الجامعة لشؤون التعليم والطلاب
جامعة عين شمس

الأستاذ الدكتور/ محمد سعيد عبد الوهاب

أستاذ بقسم الحسابات العلمية
العميد السابق لكلية الحاسبات والمعلومات
المشرف على قسم نظم المعلومات
جامعة عين شمس

الدكتور/ إسماعيل عبد الحميد طه

أستاذ مساعد بالكلية الفنية العسكرية بالقاهرة

القاهرة ٢٠٠٦