2009-01-01

Hybrid Layered Intrusion Detection System

Varsha Sainani
University of Miami, v.sainani@umiami.edu

UNIVERSITY OF MIAMI

HYBRID LAYERED MULTIAGENT INTRUSION DETECTION SYSTEM

By

Varsha Sainani

A THESIS

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Master of Science

Coral Gables, Florida

May 2009

UNIVERSITY OF MIAMI

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Science

HYBRID LAYERED MULTIAGENT INTRUSION DETECTION SYSTEM

Varsha Sainani

Approved:

SAINANI, VARSHA (M.S., Electrical and Computer Engineering)
Hybrid Layered Multiagent Intrusion Detection System (May 2009)

Abstract of a thesis at the University of Miami.

Thesis supervised by Professor Mei-Ling Shyu.

No. of pages in text. (83)

The increasing number of network security related incidents has made it necessary for the organizations to actively protect their sensitive data with network intrusion detection systems (IDSs). Detecting intrusion in a distributed network from outside network segment as well as from inside is a difficult problem. IDSs are expected to analyze a large volume of data while not placing a significant added load on the monitoring systems and networks. This requires good data mining strategies which take less time and give accurate results. In this study, a novel hybrid layered multiagent-based intrusion detection system is created, particularly with the support of a multi-class supervised classification technique. In agent-based IDS, there is no central control and therefore no central point of failure. Agents can detect and take predefined actions against malicious activities, which can be detected with the help of data mining techniques. The proposed IDS shows superior performance compared to central sniffing IDS techniques, and saves network resources compared to other distributed IDSs with mobile agents that activate too many sniffers causing bottlenecks in the network. This is one of the major motivations to use a distributed model based on a multiagent platform along with a supervised classification technique. Applying multiagent technology to the management of network security is a challenging task since it requires the management on different time instances and has many interactions.

To facilitate information exchange between different agents in the proposed hybrid layered multiagent architecture, a low cost and low response time agent communication protocol is developed to tackle the issues typically associated with a distributed multiagent system, such as poor system performance, excessive processing power requirement, and long delays. The bandwidth and response time performance of the proposed end-to-end system is investigated through the simulation of the proposed agent communication proto-col on our private LAN testbed called Hierarchical Agent Network for Intrusion Detection Systems (HAN-IDS). The simulation results show that this system is e±cient and extensible since it consumes negligible bandwidth with low cost and low response time on the network.

I dedicate my thesis and my work to most important people in my life, My Parents and My Late Grandmother.

Acknowledgments

I would like to extend my sincere gratitude and appreciation to my mentor and Chairman of the Committee, Dr. Mei-Ling Shyu, for her guidance, support, motivation and suggestions throughout my work.

I would also like to thank Dr. Moiez Tapia of the Department of Electrical and Computer Engineering and Dr. Shu-Ching Chen, of the School of Computer Science at the Florida International University for accepting my request to serve on my thesis Committee and for their helpful suggestions and support.

I would like to thank the members of AOML, NOAA who always stand by me, especially Dr. Sundaraman GopalKrishnan, Dr. Frank Marks, Dr. Robert Rogers, Dr. Xuejin Zhang, and Mr. Thiago Quirino for their cooperation.

I would also like to thank my friends and colleagues whom I have met and known while attending the University of Miami, in particular Mr. Guy Ravitz, Miss Zifang Huang and Mrs. Lin Lin.

Finally, I extend my utmost gratitude to my parents for their unconditional support in all my decisions and enormous encouragement. Without them this work would not have been possible.

I would also like to thank my best friends for their love and patience with which they have been bearing with me.

University of Miami *VARSHA SAINANI*

May 2009

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Intrusion Detection System

The growing importance of network security is shifting security concerns towards the network itself rather than being just host-based. Security services must be evolving into network-based and distributed approaches to deal with heterogeneous open platforms and support scalable solutions. Intrusion detection is the process of identifying network activity that can lead to a compromise of security policy. Intrusion Detection Systems (IDSs) must analyze and correlate a large volume of data collected from different critical network access points. This task requires an IDS to be able to characterize distributed patterns and to detect situations where a sequence of intrusion events occurs in multiple hosts. As network-based computer systems play increasingly vital roles in modern society, they have become the target of intrusions by the hackers. In addition to intrusion prevention techniques such as user authentication and authorization, encryption, and defensive programming, IDSs are often used as another wall to protect computer systems.

An intrusion is defined as "*any set of actions that attempt to compromise the integrity, confidentiality or availability of information resources*" [Heady90]. Network intrusion detection model is concerned with events that alter the usual mode of operation of an installation, having a negative impact on its network system. Often data and other information losses may be produced. Examples of such events are virus attacks, worms, zombies, trapdoors, denial of services etc. from malevolent sources. The current state of information technology and communication technology allows for a dramatic change in the mode of management of these undesired events. Today it is possible to analyze the system and network properties in real time. On this basis a decision maker can evaluate the current situation (what is happening?) as well as its short term evolution (what may happen if?) within different scenarios, and elaborate potential action plans to apply (what to do?), so that an adequate real time risk management is performed.

As accuracy is the essential requirement for an IDS, its extensibility and adaptability are also critical in today's network computing environment. There can be multiple weak points for intrusions to take place in a network system. For example, at the network level, malicious IP packets can crash a host, and at the host level, vulnerabilities can occur in system software which can be exploited to execute an illegal root shell. Since malicious activities at different intrusion points are normally recorded in different data sources, an IDS often needs to be extended to incorporate additional modules that specialize in certain components of the network systems. Therefore, IDSs need to be adaptive in such a way that frequent and timely updates are possible.

Building an effective IDS acquires enormous knowledge-based supports. System builders rely on their intuition and experience to select the statistical measures for anomaly detection. Experts first analyze and categorize attack scenarios and system vulnerabilities, and hard-code the corresponding rules and patterns for misuse detection. Because of the manual and ad hoc nature of the development process, current IDSs have limited extensibility and adaptability. Some of the recent research and commercial IDSs have started to provide built-in mechanisms for customization and extension. These systems also handle a fixed set of network traffic events.

The two main intrusion detection techniques are *misuse detection* and *anomaly detection*. Misuse detection systems, for example, IDIOT [Kumar95] and STAT [Ilgun95] use patterns of well known attacks or weak spots of the system to match and identify known intrusions. Misuse detection techniques in general are not effective against novel attacks that have no matched rules or patterns yet. On the other hand, *anomaly detection* systems observe flag activities that deviate significantly from the established normal usage profiles as anomalies or in other words as intrusions. Anomaly detection techniques can be effective against unknown or novel attacks since no prior knowledge about specific intrusions is required. However, anomaly detection systems tend to generate more false alarms than misuse detection systems because an anomaly can just be a new normal behavior [Garuba08], [Klusch03]. Some IDSs like IDES and NIDES use both anomaly and misuse detection techniques.

As an illustration, network intrusion detection, one of the current research focuses, is in dire need of powerful classification approaches to meet its requirements in both accuracy and operational merits, as it becomes a necessary and vital real-time application in the Internet. Network intruders have found the Internet the perfect environment to develop various algorithms, to cause serious damage to people, corporations, and society. Therefore, the investigation of adequate and practical solutions to network security problems with both high detection rate and favorable operational benefits are increasingly needed to safeguard the network systems and the inherently crucial information residing in them in real-time.

This thesis aims to develop a more systematic and automated approach for building IDSs. We have developed a set of tools that can be applied to a number of tasks such as capturing data, extracting features, classifying them into known and unknown attack categories, and ultimately stopping the ongoing malicious activity. We take a data-centric point of view and consider intrusion detection as a data analysis process. The central theme of our approach is to apply data mining techniques to the extensively gathered data to compute a model that accurately captures the actual behavior and patterns of the intrusions and normal activities. This approach significantly reduces the need to manually analyze and encode intrusion patterns, as well as the guesswork in selecting statistical measures for normal usage profiles. The resultant model is more effective because it is computed and validated using a large amount of network data.

1.2 Agent Technology

As we know, computers are not very good at knowing what to do: every action a computer performs must be explicitly anticipated, planned for, and coded by a programmer. If a computer program ever encounters a situation that its designer did not anticipate, then the result is not usually good. For most part, we accept computers as obedient, literal unimaginative servants. For many applications, it is entirely acceptable. However, for an increasing large number of applications, we require systems that can decide for themselves what they need to do in order to satisfy their design objectives. Such computer systems are called agents.

The definition presented in [Weiss01] for agents is "An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives."Agents (adaptive or intelligent agents and multi-agent systems) constitute one of the most prominent and attractive technologies in Computer Science at the beginning of this new century. Agent and multi-agent system technologies, methods, and theories are currently contributing to many diverse domains. These include information retrieval, user interface design, robotics, electronic commerce, computer mediated collaboration, computer games, education and training, smart environments, ubiquitous computers, and social simulation. They are not only very promising technologies, but also emerging as a new way of thinking, a conceptual paradigm for analyzing problems, designing systems, and dealing with complexity, distribution and interactivity, and perhaps a new perspective on computing and intelligence. Agent-based computing has been a source of technologies to a number of research areas, both theoretical and applied.

Agent capabilities are more useful when integrated with data mining strategies when it comes to identify patterns and do respective jobs associated with them automatically like in weather analysis and prediction, identification of theft or sales patterns in commercial or financial organizations, identification of network patterns for intrusions, etc.

1.3 Multiagent Systems

Agents operate and exist in some environment which typically is both computational and physical. The environment might be open or closed, and it might or might not contain other agents. Although there are situations where an agent can operate by itself, the increasing interconnection and networking of computers have made such situations rare, and in the usual state of affairs, the agent interacts with other agents.

Multiagent systems (MAS) are the best way to characterize or design distributed computing systems. Information processing is ubiquitous. There are computer processors seemingly everywhere, embedded in all aspects of our environment. The large number of processors and the myriad ways in which they interact make distributed computing systems the dominant computational paradigm today.

Multiagent environments, in general, depict following additional characteristics [Weiss01]:

1. Multiagent environments provide an infrastructure specifying communication and interaction protocols.
2. Multiagent environments are typically open and have no centralized designer.
3. Multiagent environments contain agents that are autonomous and distributed, and may be self-interested or cooperative.

Multiagent systems can be distributed or centralized. However, in multiagent systems, information involved is necessarily distributed and resides in information systems that are large and complex in several senses [Weiss01]:

1. they can be geographically distributed,
2. they can have many components,
3. they can have a huge content, both in the number of concepts and in the amount of data about each concept, and
4. they can have a broad scope, i.e., coverage of a major portion of a significant domain.

Also, the components of the system are dynamic and their content is changing so rapidly that it is difficult for a user or an application program to obtain correct information, or for the enterprize to maintain consistent information. Multiagent systems can exist with both centralized and decentralized structures.

In centralized approaches, in order to achieve intelligent coordinator agent can be designed, that is responsible for detecting interdependencies between the local agent's activities at successive levels of abstraction. Once the knowledge to reason at the upper levels is elicited, the centralized approach provides a model of predictable behavior where all possible cases of inconsistencies are all analyzed a priori and are taken into account by the upper level modules. However, the bottleneck of this type of models is precisely the knowledge elicitation of the different inconsistencies: in many cases it is difficult to identify the precise way in which methods and domain models need to be integrated in order to solve a problem. Once such a centralized model is built, the maintenance process is complex because, if additional lower level models are introduced, a sequence of changes has to be produced in the upper level models to take into account the potential modification of the situations produced by the new element.

This approach is contrasted by a decentralized stance, where no such special agents exist and agents interact laterally: agents are endowed with knowledge to discover inconsistencies between their intended actions and interchange messages to mutually adapt their local decisions, so as to converge on one or several sets of consistent local control plans. The former coordination model leads to hierarchical integration of control plans as determined by the upper level functions, while in the latter this integration emerges from agent interactions as implied by the agents' social knowledge. From an abstract point of view both approaches seem feasible. However the first seems more reliable with respect to operation, while the second appears more adequate from a design perspective. The decentralized approach promises systems that are easier to build, because the model needs to be defined very accurately only at the local level, where it is more feasible to felicitate the knowledge to solve the specific problems of each agent.

The mode of operation of the agent model is given by a simple reasoning cycle. It contains the following steps:

1. The perception subsystem captures percepts and messages from other agents and updates the information model accordingly;

2. The conversation agenda is updated and reordered in accordance with the social strategic knowledge. As a result of the selection of some conversation, new tasks are added to the task agenda;

3. The motivation is matched against the information model and eventually more new tasks are created on the task agenda;

4. Using the local strategic knowledge the task agenda is reordered and some tasks are chosen for execution; For every task two approaches are to be followed:

    (a) The local problem-solving approach where, using the knowledge about relation between tasks and methods, a method is chosen for execution. Usually basic methods are preferred to compound methods, and the latter are given priority over social methods;

    (b) The delegation approach, if in the previous process no method is available in the internal problem-solving knowledge to cope with a task. In this case, the agent consults its acquaintance models and identifies a collection of agents that may perform the required tasks. The then assigns the task to the most adequate agent;

5. The action subsystem performs actions and sends messages as indicated by the intelligence subsystem in the information model.

The rationale for interconnecting computational agents and expert systems is to enable them to cooperate in solving problems, to share expertise, to work in parallel on common problems, to be developed and implemented modularly, to be fault tolerant through redundancy, to represent multiple viewpoints and the knowledge of multiple experts, and to be reusable.

1.4 Agent Communications

In a MAS, agents need to communicate to achieve their goals in a global coherent manner. Communication languages and protocols are thus needed to enable the agents to coordinate their individual actions, behavior, exchange information and knowledge.

Software agents suggest a paradigm for software development that emphasizes autonomy, adaptability, and cooperation, both at the design time and runtime. This approach seems appealing in a world of distributed, heterogeneous systems [Cao04] [Khasteh06]. An agent communication language (ACL) that allows the agents to interact while hiding the details of their internal workings will result in an agent community with the capability of tackling the problems that no individual agent could. The most popular Agent Communication Languages (ACLs) are: FIPA-ACL (by the Foundation for Intelligent Physical Agents, a standardization consortium) and KQML (Knowledge Query and Manipulation Language). FIPA Agent Communication specifications deal with Agent Communication Language (ACL) messages, message exchange interaction protocols, speech act theory-based communicative acts and content language representations.

The Knowledge Query and Manipulation Language (KQML) is one of the most commonly used agent communication languages due to its versatility and generality of purposes [Froese03]. KQML supports multiagent communication through an extensible set of reserved primitives called *performatives* that represent communicative acts. In proposed architecture, KQML was adopted as the default ACL [Weiss01]. To make agents understand each other they have to not only speak the same language, but also have a common ontology. An ontology is a part of the agent's knowledge base that describes what kind of things an agent can deal with and how they are related to each other. An example of a framework that implements a standard agent communication language (FIPA-ACL) is Jade [Jade09].

1.5 Data Mining

There is a tremendous explosion in the amount of data that organizations generate, collect and store. Managers are beginning to recognize the value of this asset, and are increasingly relying on intelligent systems to access, analyze, summarize, and interpret information from large and multiple data sources. These systems help them make critical business decisions faster or with a greater degree of confidence. Data mining is a promising new technology that helps bring business intelligence into these systems. While there is a plethora of data mining techniques and tools available, they present inherent problems for end-users including complexity, required technical expertise, lack of flexibility and inter-operability, etc. These problems can be mitigated by deploying software agents to assist end-users in their problem solving endeavors.

Data mining, a process of analyzing data to identify patterns or relationships for autonomously extracting useful information or knowledge, has been increasingly developed to provide solutions for uncovering useful and/or unexpected information from large volumes of data in various research areas, such as multimedia data analysis, visualization, biomedicine, market analysis, homeland defense, threat assessment systems, intrusion detection, credit card fraud detection, and other applications.

Data mining can also be described as the process of extracting hidden patterns from data. As more data is gathered, with the amount of data doubling every three years, data mining has become an increasingly important tool to transform this data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection, and scientific discovery. Data mining can be applied to data sets of any size. However, while it can be used to uncover hidden patterns in data that have been collected, obviously it can neither uncover patterns which are not already present in the data, nor can it uncover patterns in data that have not been collected.

Among various data mining techniques, classification is important as it is one of the key techniques in most data mining applications. It is the act of distributing things into classes or categories of the same type or characteristics. In other words, it is a basic cognitive process of arranging things or objects into classes or categories. A classification procedure is learned from given training data and then tested on test data. For multivariate data, a classification procedure predicts different output patterns. Classification techniques have been applied to numerous research areas including market analysis, homeland defense, threat assessment systems, intrusion detection systems, credit card fraud detection, face recognition, etc. Generally speaking, there are two broad types of classification procedures:supervised and unsupervised classification [Laskov05]. Supervised classification can be defined when the output patterns and their range of values are given. As its counterpart method, unsupervised classification can be defined when none or only part of the output pattern information is known before the classification process

Supervised classification takes into use a class quantity that is high enough to distinguish a class from other class types. In supervised classification, classification accuracy or detection rate is most important performance evaluation measure of a classifier. On the other hand, operation merits, another important performance evaluation measure, refer to the usage benefits of the classifier in terms of how fast it executes, how much memory it consumes, how large the programming code is, being lightweight, and so on. For instance, a classifier may have a high detection rate or high accuracy, but is slow to execute and/or requires a considerable amount of storage space to save, for instance, generated decision rules; whereas, another classifier may execute faster and use some form of heuristic that does not require so many rules to be stored. All these merits taken together give the overall performance merit for a classifier.

When relatively little information is known about the data before classification, unsupervised classification is usually employed . In addition, no human effort is required to provide the fore knowledge of the class labels of the data set. This is the reason behind, clustering algorithms being used to aggregate and classify data instances into classes while performing unsupervised classification. Unsupervised classification is highly efficient and useful when real-world applications are considered because usually there is very less class-related information available for them.

1.6 Contributions

The major contributions of my thesis work fall in the design and implementation of the communication protocol which facilitates the creation of a real multiagent system in the domain of network intrusion detection. It also adds the capability of real time exchange. In the domain of proposed communication protocol it facilitates exchange of information in terms of KQML messages. Another contribution is in bringing together all the available tools which are implemented in various programming languages for the testbed. As it will be described in the following chapters the network testbed set-up was a big challenge as all the tools though readily available were discrete and functional only in themselves irrespective of other parts. Each of these tools like feature extractor, traffic generator, C-RSPM classifier, etc. were written using different programming languages and irrespective of the presence of the other tools. Finally, having the testbed realized and ready, the next challenge was to encode the entire architecture calling these tools and at the same time integrate it with the communication protocol. In this thesis, the architecture has been successfully integrated on the testbed, along with the communication protocol and the classification algorithms.

1.7 Scopes and Limitations

There are some scopes and limitations in the current proposed MAS. First, the proposed architecture is limited to a small subnet and as for now does not take into account any other existing network. Second, no fault tolerance capabilities are added to the proposed architecture, i.e., the failure of one layer will eventually fail the entire system. Third, at this time, only one ongoing attack is considered in the experiments. Finally, the proposed architecture has its scope only until the solution is decided and informed, and it does not consider problem resolving time.

1.8 Outline of the Thesis

This thesis is organized as follows. In Chapter 2, a literature review of the existing intrusion detecting methods and their applications is presented with a focus on their performance such as accuracy and operational benefits. Chapter 3 describes the proposed MAS and its architecture including a detailed discussion of setting up the testbed and simulating real network conditions. Chapter 4 supports the proposed MAS with several experiments. The LAN testbed Setup, the Relative Assumption Modeling method, and Feature Extraction techniques are also introduced in this chapter.

Chapter 5 shows and discusses the results and analysis obtained from the experiments. Chapter 6 ornaments our work with an existing real application in the field of hurricane data analysis and collection. Chapter 7 concludes this thesis by highlighting the achievements of the proposed MAS and by providing some possible future research directions.

CHAPTER 2

Literature Review

In the last couple of years, while the cost of information processing and Internet accessibility has fallen greatly, network systems have played an increasingly critical role in modern society. The popularity of the web-based applications leads to the interconnection of almost all the computers in the world in a global network that facilitates communications, among people. At the same time, as increasingly sensitive data are being stored and manipulated through the Internet along with the fact that various intrusions are bringing serious damage to people, corporations, and society as a whole, network security has become an extremely vital issue that has strongly attracted both researchers and commercial organizations.

Recently, multiagent systems have become popular since they promise to provide high-level interactions in intricate applications for modern computing and information processing systems, a very good example being an Intrusion Detection System (IDS).

2.1 Types of Agent Architectures

As multiple agents have to operate and exist in an environment which is both computational and physical, there are situations where an agent can operate by itself but the increasing interconnection and networking of computers have made such situations rare. In usual state of affairs, the agent has to interact with other agents [Weiss01]. There have been a number of multiagent architectures developed in the literature, such as the *Logic Based Architecture*, *Reactive Architecture*, *Belief Desire Intention Architecture (BDI)*, and *Layered Architecture.*

In logic based architectures decision making is realized through logical deductions [Barringer89], [Muller95]. Reactive architectures employ direct mapping from situation to action for implementing decision making [Georgeff87], [Kaelbling86]. While in belief-desire-intention architecture decision making depends upon the manipulation of data structures representing the beliefs, desires, and intentions of the agent, [Rao91], [Rao91a], [Rao92], [Rao92a], [Rao93], [Rao96], [Rao96a]. Finally, in layered architectures decision making is realized via various software layers, each of which is more-or-less explicitly reasoning about the environment at different levels of abstraction [McCarthy69], [Weiss01]. For our purpose, we adopt the layered architecture since it is an approach to design subsystems handling many subproblems of an application.

2.2 Layered Architectures

A number of different layered architectures exist, including *Touring Machines*, *InteR RaP*, *ATLANTIS*, etc.,[Sycara03]. Layered architectures can be further broadly categorized into the following two types depending on the direction of the flow control: *Horizontal* and *Vertical.* In *Horizontal* architectures, each interaction focuses on assigning central control and each layer performs independently. In other words, each layer comprises of one or more agents. Each layer gets an input and gives an output. If there are n layers and each layer has possible m actions, there are m^n

interactions. This type of architecture has conceptual simplicity, but at the same time it lacks coherence and needs a mediator function to decide which layer has the control. The *Touring Machine* is an example of horizontal architecture which consists of reactive, planning, and modeling layers along with a control subsystem. The use of such central control reduces the autonomy.

On the other hand, in the *Vertical* architectures, the inputs and outputs are handled by only one layer at a time and the control flows through each layer until the output is achieved. Under the same definitions of m and n as mentioned previously, it has the complexity of $m^2(n-1)$, which is less than the horizontal approaches. The vertical architectures include a natural decomposition of the functionality and pragmatic approach to solutions. The *InteR Rap* system is a good example for vertical layered architectures [Weiss01]. It constitutes of cooperation, planning, and behavior layers. However, this type of architectures is not fault tolerant as the failure of one layer will pull down the entire system. In addition, how to handle the interaction between layers is also a challenging issue.

2.3 Reactive Architectures

Reactive architecture approach was developed by *Rodney Brooks*. It stresses on two characteristics. The first one is that an agent's decision-making is realized through a set of task accomplishing behaviors. Each of these behavior modules is intended to achieve some particular task. The second characteristic is that many behaviors can 'fire' simultaneously. There are many existing studies on reactive architectures such as *Brook's behavior languages*, *Agre's* and *Chapman's PENGI* system [Agre87], [Agre96], *Situated Automata*, and *Pattie Maes's Agent Network Architecture* [Maes89], [Maes90], [Maes90a], [Maes91]. *Brook's* [Brooks86], [Brooks90], [Brooks91], [Brooks91a], reactive architecture brings simplicity, economy, computational tractability, robustness against failure, and elegance. A major selling point of purely reactive systems is that the overall behavior emerges from the interaction of the component behaviors when an agent is placed in its environment. However, the term 'emerges' suggests that the relationship between individual behaviors, environment, and the overall behavior is not easy to realize [Chalupsky02].

Due to the advantages and disadvantages of each architecture, it gives rise to the concept of creating hybrid architectures and brings out the best from two or more architectures. There exist hybrid architectures such as *Procedural Reasoning System (PRS)*, *IRMA*, and *GRATE*, to name a few. Almost all of these architectures constitute of a number of components that are put together in order to solve tasks. The layered architecture approach has yet not been employed for hybrid architectures [Assal04].

2.4 Distributed Agent Architectures

IDSs have undergone rapid development in both power and scope in the last few years. There are various types of architectures for IDSs that can be summed up into four main categories: monolithic, hierarchic, agent-based, and distributed systems [Edmund87], [Edmund88], [Edmund90]. However, much improvement could be done for these architectures, as the nature of the artificial attacks keeps changing. Recently, the agent concept has been widely used in distributed environments because it provides many favorable characteristics including scalability, adaptability, graceful degradation of service, etc. as compared to the non-agent based systems [Briggs95]. Most of the distributed agent-based IDSs introduced more traffic into their residing network, and therefore the communication protocol between various entities is also an important aspect that has to be considered [Conry91]. At the same time, most of the agent-based IDSs require comparatively high processing power in local machines to run the agents and other supportive software. Hence, a lightweight agent system with low network traffic generation requirements is needed. This can be accomplished with the use of appropriate data mining strategies.

One of the well known examples of applying distributed agent design methodology in the intrusion detection domain is the Distributed Intrusion Detection System (DIDS). DIDS attempts to build a distributed system based on monitoring agents that reside at every host in the network. A centralized data analysis component called the DIDS director agent is solely responsible for the analysis of the network traffic data collected by each monitor [Corkill82]. Distributed systems present both advantages and disadvantages. On one hand, the system utilizes the real-time traffic information from various sources, in the form of data from various host monitors or to assess the security status of its residing network. However, on the other hand, the system's scalability is poor for large networks as an increasing number of hosts monitoring the network also significantly increases the work load of the DIDS director agent . Additionally, the data flow between host monitors and the director agent may generate significantly high network traffic overheads.

2.5 Agent Communication

KQML and FIPA, the two languages differ primarily in the details of their semantic frameworks. Both languages assume a basic non commitment to a reserved content language and both the languages have the same syntax. That is, a KQML message and a FIPA ACL message look syntactically identical except,for their different names for communication primitives. We have seen the emergence of a multitude of applications and systems built around ACLs, over the past few years. Most of these systems, are multiagent systems that use an ACL for interagent communication, or APIs to facilitate the incorporation of speaking capabilities into an application. All the systems mentioned here use some variant of KQML as their ACL.

Infosleuth [Bayardo97], [Nodine97], is a project that emphasizes the semantic integration of heterogeneous information in an open dynamic environment. These communicating agents which were initially written in Java use simple services for authentication, monitoring, and visualization of the agent's interaction. An integral part of these architectures is the ontology of agent, which assists with the semantic integration of the information. Infosleuth agents engage in conversations rather than in single-message exchanges.

Knowledgeable Agent-Oriented System [Bradshaw97], is another work for Boeing project aimed at providing an infrastructure for agent development. Kaos relies heavily on object-oriented technology; it uses, for example, a CORBA-based message delivery mechanism. It also emphasizes persistent interaction between agents that take into account not only the particular communication primitive but the content of the message and the applicable conversation policies. This system allows the design of agents that support specialized suites of interactions.

Infomaster [Genesereth97], is another information integration system from Stanford that uses ACL, the KQML variant with Knowledge Interchange Format (KIF) as its content language. The resulting language does not observe the distinction between the content layer and the message layer. Infomaster integrates structured information sources, giving the illusion of a centralized, homogeneous information system.

JAFMAS [Chauhan97], [Chauhan98], supports directed (point-to-point) communication as well as subject-based, broadcast communications.

Jackal [Cost98], is another Java package that allows applications written in Java to communicate via an ACL. KQML is currently being used for this package. Jackal strongly emphasizes conversations between agents. It provides a flexible framework for designing agents around conversations and includes extensive support for registration, naming, and control of agents.

2.6 Classification

Data mining techniques such as classification can be useful for both misuse detection and anomaly detection. In network intrusion detection, classification can be applied to classify network data consisting of malicious behaviors, and several existing approaches such as RIPPER, Naive Bayes, and multi-Bayes classifiers have been successfully used to detect malicious virus code. Various other intrusion detection algorithms using classification techniques have been developed for this purpose which are described in [Ghosh99], [Lee99].

Nowadays, very few effective unsupervised classification methods capable of adapting to various domains have been proposed and developed. Unsupervised classification usually requires a combination of clustering and supervised classification algorithms to be employed. Unsupervised classification is relatively faster and less expensive, while most of the existing unsupervised classification algorithms, especially those which do not require any known class related information such as the number of classes and the maximum number of instances in each class, suffer from a lack of high classification accuracy [Lark95a],[Lark95b], and broad effectiveness in applications with data sets with different inter-class variability [Mukkamala02].

During the past decade, supervised classification has become an essential tool that has been applied successfully in diverse research areas [Angiulli05], [Kim05], [Tseng05], [Yin06]. Its main goal is to determine a measure from instances belonging to the same class that is large enough to reject instances belonging to other classes, while at the same time low enough so as to take into account the variability found among the instances belonging to the same class. Some of the existing methods are more inclined towards the needs of specific research domains, while others assume a more generic and comprehensive facet. Moreover, there are issues that commonly arise in supervised classification, such as the low variability among classes in a data set, classification ambiguity issues, and time and space complexities associated with the practical implementation of these algorithms.

Various approaches have been developed in an attempt to abate the effects of these common issues. For instance, the authors of [Kim05] examined the effectiveness of four dimensional reduction techniques such as Centroid, Orthogonal Centroid, LDA (Linear Discriminant Analysis)/GSVD (Generalized Singular Value Decomposition), and LSI (Latent Semantic Indexing)/SVD (Singular Value Decomposition) in the reduction of data dimensionality, and introduced a novel threshold based classifier for centroid-based classification and support vector machines (SVMs) that capture the overlapping structure between closely related classes. Although their proposed classifier achieves relatively high classification accuracy and relatively low computational complexity, the core methods for both dimensionality reduction and classification are based on previous existing approaches.

A decision tree can also be exploited to formulate genetic algorithms to create rules that match to the set of anomalous connections. There are alternative classification approaches which can be effectively utilized for intrusion detection purposes [Qui93]. With intrusions, it is observed that over the time, the user establishes profile based on the number and types of commands they execute. Data mining classifier approaches like SOM (Self Organizing Maps) and LQM (Learning Vector Quantization) can be utilized for reducing dimensionality of these numbers. Furthermore, nearest neighbor classifier approaches based on SOM and LQM can be used to refine the collected network data in intrusion detection.

Thus, various classification approaches can be employed on network data for obtaining specific information and detecting intrusions. A number of such systems have been developed which utilized the fast and efficient computation and pattern matching strategies of data mining to complement the low cost and lightweight agent system architectures.

Supervised classification is an essential tool for the study of intrusion detection, providing various techniques that discover and extract previously unknown patterns from large quantities of data. Supervised classification techniques are widely used in intrusion detection systems. Specifically, the existing supervised intrusion detection methods can be categorized into two main types: misuse detection and anomaly detection [Bace01], [Lee99], [Lee00], [Markou03a], [Markou03b], [Noel02].

Misuse detection is based on the signature modeling of known network intrusions, which has the advantage of higher detection accuracy in detecting known network attacks [Barb01] and the shortcoming of an inability to detect previously unobserved attacks [Lazarevic03], [Paxson99]. Whereas, anomaly detection, which is based on the signature modeling of normal network traffic [Denning87], [Mahoney03], has the advantage of being able to detect new types of network intrusions [Anderson95], [Dokas02], [Lazarevic03] while suffering, as a disadvantage, from high false alarm rates [Fuller94], [Labib04].

Supervised classification can achieve a relatively high accuracy, but it usually requires tremendous efforts in terms of time, complexity, manpower, and cost, especially when large data sets are involved in the process. Currently, very few supervised classification approaches are able to address all these issues. In other words, most of the existing supervised classification schemes have difficulty in satisfying both classification accuracy and operational benefits as requested by the increasing real-time and/or practical applications.

2.7 Agent Technology and Data Mining

Various distributed intrusion detection architectures using the multiagent design methodology and the data mining techniques have been developed. These approaches widely range from being comprised entirely of mobile agents like the MANET system, [Jin05], [Pahlevanzadeh07], being merely a collection of static agents as in [Spafford00], or a combination of both as in DIDMA system, [Kannadiga05].

In [Kannadiga05], a system called "Distributed Intrusion Detection using Mobile Agents (DIDMA)" attempted to overcome the scalability issues inherent in the original DIDS architecture by employing mobile agents in the data analysis task. Thus, by decentralizing data analysis, DIDMA hoped to significantly neutralize the effects of the scalability issues.

Another well known system called *BODHI* [Kargupta00] was designed for heterogeneous data sources based on the techniques such as supervised inductive distributed function learning and regression. It focuses on the guarantee for correct local and global data models having least network communication. It was implemented in Java and offers message exchanges and runtime environments of the agent systems for the execution of mobile agents at each local site. A central facilitator agent takes care of initializing and coordinating the data mining tasks.

A Java-based multi-agent (*JAM*) [Stolfo97] system is designed to be used for meta-learning in distributed data mining environments. In this system, each site agent builds a classification model, where different agents built their classifiers using different techniques. JAM also provides a set of meta-learning. Once combined together, the classifiers are computed with the central JAM system coordinating the execution of these modules to classify data sets at all data sites simultaneously.

In [Vaidehi04], a distributed agent-based IDS analyzes anomalies to detect and identify the denial-of-services (DoS) and data theft attacks. It also attempts to respond to intrusions in real time by sending out alerts to the designated network administrator when network intrusions are detected. One of its main drawbacks is the design complexity of its comprising agents, since each agent must take on almost all work load of network traffic sniffing, data parsing, and intrusion detection. In addition, its data mining techniques are less powerful since they are capable of detecting only a limited number of attacks.

In [Xie06], also a distributed agent-based intrusion detection system is proposed which includes autonomous agents independent decision making with speed and accuracy of a classification scheme name Principal Component Classifier (PCC). It attempts to resolve the problem of intrusion detection by proposing a two layer agent architecture, where the problem is divided into two sub-problems and then tackled by two layers of agents. Its concluded here that a linear relationship between the response time and the number of agents introduced into the system, which implies that the response time performance of the system will degrade linearly with scalability.

In [Sainani09], above mentioned approach in [Xie06] was enhanced in capability and efficiency by adding another layer of intelligence. It also proposed a Java based communication protocol for the proposed set of agents. This protocol facilitates information transfer from one level of agents to another, which may or may not be located on same platform or machine. In other words this communication scheme allows agents to communicate with each other in order to achieve co-operation for identifying the ongoing attack. It is proved that the proposed protocol consumes negligible bandwidth and has low response time, thereby making the system low cost and fast.

In [Shyu09], a prototype of this work is proposed which handles the issue of intrusion detection by utilizing agent capabilities at three levels, each having its predefined set of capabilities. Each of this level is supported by speed and accuracy of classification strategies in order to mine the network data and identify the attack types. It also structures the in-depth analysis by the agents for the decisions they make, in the form of ‘policies’ and ‘rules’.

CHAPTER 3

The Proposed Architecture

In this study, a novel data mining assisted multiagent-based intrusion detection system architecture is presented [Shyu09]. It integrates a multi-class supervised classification algorithm and the agent technology for network intrusion detection. A hybrid layered multiagent architecture is designed that combines *layered* and *reactive* architectures in a hierarchy of three different layers having agents at each layer, namely the *Host*, *Classification*, and *Manager* Agents [Sainani09].

This is an attempt to depict the best of both types of *layered* architectures and its combination with the *reactive* architecture, which overcomes the disadvantages of the *horizontal* and *vertical* layered architectures, making the proposed architecture logically and semantically complete as well as stable. As mentioned earlier in chapter 2, one of the disadvantages of the *layered* architectures is the handling of the interactions between layers.

The cooperation between agents in a multiagent system is a must. To address this issue, we ornament the proposed architecture with a communication protocol that facilitates several desirable functionalities for a multiagent architecture, such as low response time, low cost, and reliability. The *Knowledge Query and Manipu-*

lation Language (KQML) agent communication language *(ACL)* is adopted for the developments of the proposed communication protocol [Froese03].

The proposed architecture utilizes the high accuracy and speed response of the *Principal Component Classifier (PCC)* [Xie06] at its first layer of proposed architecture. Once the results from PCC classification are obtained, the agents communicate them to the second layer. The second layer of the proposed architecture is integrated with the *Collateral Representative Subspace Projection Modeling (C-RSPM)* classifier [Quirino06] which includes collateral class modeling, class ambiguity solving, and classification components. Results from this stage of classification are further analyzed by the agents and 'policies' are derived which are communicated to the next layer using the proposed agent communication protocol.

This architecture is further elaborated in the following sections.

3.1 Agent Architecture Description

Figure 3.1 presents our Hierarchical Agent Network for Intrusion Detection System (HAN-IDS) hybrid layered multiagent-based architecture [Sainani09]. This testbed is located at the department of Electrical and Computer Engineering, University of Miami. It consists of three layers called Host, Classification, and Manager layers. Each of these layers comprises of deliberative agents which are well aware of each other's presence and are capable of communicating with each other using our developed communication scheme.

3.1.1 Host Layer

This layer marks the entry point of the proposed architecture. These end-user machines are workstations constituting the network, and they also act as the host agents inspecting each incoming network connection. Virtually, every machine in a network can be considered as a Host Agent. The Host Agents collect network connection information and classify these connections as 'normal' or 'abnormal'. Here, the *Principal Component Classifier (PCC)* is used [Xie06].

Each Host Agent belongs to one Classification Agent (in the second layer), to which it reports the connections that PCC classifies as 'abnormal'. The responsibilities of these agents are (i) capturing network traffic, (ii) detecting abnormal activities in these connections, (iii) passing the classification results to its Classification Agent, (iv) properly responding to these abnormal activities for intrusion detection, and (v) passing a subset of the normal connection instances and the abnormal connection instances to the Manager layer to be saved in a database for the purpose of re-training the classifier at a later time.

3.1.2 Classification Layer

The second layer of the architecture is the Classification Layer. The responsibilities of the Classification Agents are (i) responsible for a set of Host Agents, (ii) classifying the abnormal connection instances found in their host machines into known attack types, and (iii) passing the classification results to the Manager Agent. Each Classification Agent is facilitated with a misuse detection algorithm called the *Collateral Representative Subspace Projection Modeling (C-RSPM)* [Quirino06]. This is important as the attack type will determine how the IDS should respond to the attack to safeguard the data in the network.

Unlike the Host Agents, dedicated machines are needed to run the Classification Agents so that they have enough processing power to handle all classification requests of their Host Agents. Additionally, they have to generate 'policies' upon the instances which are identified as attacks. Once the policy is created, it is communicated to the next layer called the Manager Layer. All the Classification Agents present in the network add their policies to the policy repository in the Manager Layer.

3.1.3 Manager Layer

This layer, in terms of contemporary agent models, is the same as the planning layer. The Manager Agent is in charge of the entire system, performing several support tasks for the system. Its responsibilities include (i) assigning a Host Agent to the specific Classification Agent, (ii) assisting the Classification Agents in managing their host machines and the tasks related to them, and (iii) managing the routers and firewalls in the network.

The main task performed by the Manager Agent is to take the policies from the Classification Agents throughout the network. After receiving the policies, the Manager Agent broadcasts them to every agent present in the network. Once the policy is received by all Host Agents, the corresponding Host Agent who initiated the request implements the policy. This functionality is important as the Classification Agents can prevent or lessen the effects of a possible attack by managing resources in those nodes that they expect to be affected by the incoming attack, such as bandwidth, communication ports, and connection authorization.

3.2 Agent Communication

A simple and manageable communication scheme that utilizes a discrete number of KQML performatives is designed to accommodate the goals of all the agents. We used *TCP/IP Secure Socket Layer (SSL)* for the implementation of secure communication channel among the agents to provide total privacy and authentication capabilities.

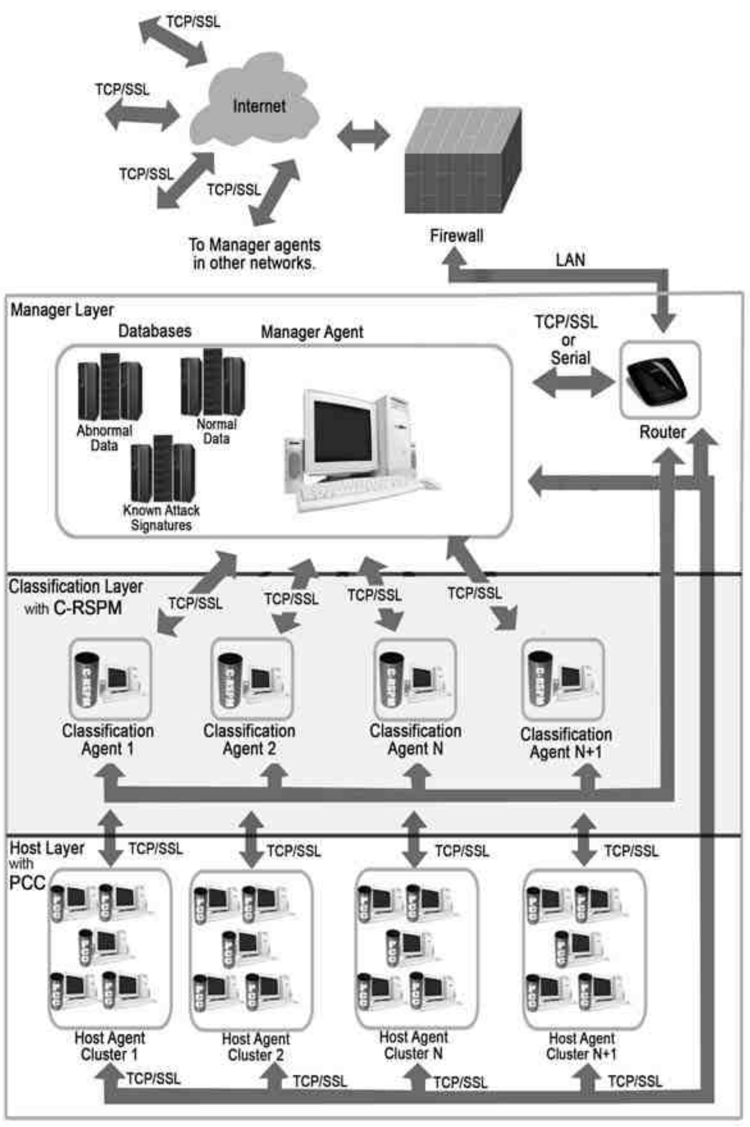


Figure 3.1: The HAN-IDS network testbed and the System Architecture of the Proposed Hybrid Layered Multiagent Intrusion Detection System

The adoption of cryptographic communication services is an important step toward making the distributed multiagent IDS architecture immune against attacks that can exploit the relatively simple agent communication scheme [Park94]. Once the main channel and important control functions are set for execution, there comes a point in the system where the logs are to be transferred to the Manager Agent so that they can be saved in the database for future reference and training of the *C-RSPM* classifier. This requires the Host Agents to transfer the logs to the Manager Agent as it holds 'abnormal' and 'normal' connection instances. If this data is transferred via the main channel, it could add delays to the ongoing threat detection which is definitely not desired. Therefore, a new channel is opened to transfer these logs and then closed as soon as the transfer is done [Koes04]. This concurrency helps avoid any interference or delays with the main channel functions, though the messages for the establishment of these channels are also written in KQML and are conversed through the main channel [Hayzelden00].

Let's discuss the proposed schema in brief first. Initially, as soon as the Classification Agent comes online, it registers itself with the Manager Agent using the "REGISTER" performative. The same registration process applies to the Host Agent when it has to register with the Classification Agent. However, since the Host Agent needs to know to which the Classification Agent it belongs as soon as it comes online, it uses the "RECOMMEND-ONE" performative to ask the Manager Agent. The Manager Agent then uses the "TELL" performative to reply to the Host Agent with the available Classification Agent's IP. As mentioned earlier, the Host Agent diagnoses each incoming connection using PCC and uses the "EVALUATE" performative to communicate this result to the Classification Agent. The Classification Agent further classifies any abnormal connection into a known attack type and derives a policy based on it. It uses the "REPORT" performative to communicate the corresponding policy to the Manager Agent. Finally, the Manager Agent broadcasts this policy to all other agents present in the network using the “BROADCAST” performative. In proposed architecture, the TCP/IP Secure Socket Layer (SSL) was adopted for the implementation of the secure communication channel among the agents to provide total privacy and authentication capabilities. The adoption of cryptographic communication services is important for making the distributed multiagent IDS architecture immune against attacks.

Here we describe how different communication messages can be deployed at different levels of proposed schema by the use of the KQML performatives.

RECOMMEND-ONE: As soon as a Host Agent comes online, it sends this KQML message to the Manager Agent as it is aware of the Manager Agent’s IP address, requesting for the Classification Agent’s IP address to which it should connect. The Manager Agent keeps the count of the number of Host Agents registered with each Classification Agent (called host agent count) to ensure that the load is symmetrically distributed for all agents. Following is an example of the RECOMMEND-ONE message, where the Classification Agent’s IP is requested in the ‘Content’ by asking ‘CA_IP’ which represents the Classification Agent’s IP.

(RECOMMEND-ONE

Sender IP: 10.0.0.3

Sender Port: 1000

Receiver IP: 10.0.0.6

Ontology: Agent Communication

Language: KQML

Content: CA_IP)

TELL: The Manager Agent uses this performative for replying to RECOMMEND-ONE from the Host Agent, informing it to which Classification Agent it should connect, in the content of the TELL message. For example, assuming that the Manager Agent is recommending the Classification Agent with IP '10.0.0.5' and port '2000'.

(TELL
Sender IP: 10.0.0.6
Sender Port: 3000
Receiver IP: 10.0.0.3
Ontology: Agent Communication
Language: KQML
Content: 10.0.0.5:2000)

REGISTER: The Classification Agents use this performative to register with the Manager Agent as soon as they come online. This message also carries the host agent count (i.e., the number of Host Agents registered with that Classification Agent) which may or may not be equal to zero initially. The Host Agents also use this performative to register with the Classification Agent upon receiving the IP address from the Manager Agent. In the following example, the 'Content' field of the message indicates that the Host Agent wants to register with Classification Agent by stating 'HA_reg'.

(REGISTER
Sender IP: 10.0.0.3
Sender Port: 1000
Receiver IP: 10.0.0.5
Ontology: Agent Communication

Language: KQML

Content: HA_reg)

UNREGISTER: As the name suggests, this performative does exactly the opposite of the 'REGISTER' performative. Whenever a Host Agent wants to quit from the network, it sends this UNREGISTER message to the Classification Agent and in turn the Classification Agent updates its host agent count by removing this Host Agent from its cluster. The same procedure follows when the Classification Agent desires to quit. The 'HA_unreg' value in the 'Content' field in the listed example indicates that the Host Agent wants to disconnect.

(UNREGISTER

Sender IP: 10.0.0.14

Sender Port: 1014

Receiver IP: 10.0.0.5

Ontology: Agent Communication

Language: KQML

Content: HA_unreg)

FORWARD: Every time a Host Agent registers or un-registers with the Classification Agent, the Classification Agent needs to update the host agent count and informs the Manager Agent about this change to ensure the consistency of the information. This performative is employed by the Classification Agent to inform the Manager Agent about the new host agent count. For example, the 'Content' field of the following message indicates that the new host agent count is '101' for this Classification Agent.

(FORWARD

Sender IP: 10.0.0.5

Sender Port: 2000

Receiver IP: 10.0.0.6

Ontology: Agent Communication

Language: KQML

Content: HA_count: 101)

EVALUATE: This performative is employed by the Host Agents for requesting the Classification Agent to evaluate an 'abnormal' instance into the corresponding attack type. It conveys the information of that particular instance which was classified as 'abnormal' to the Classification Agent in this message. The message shown below is a message from the Host Agent to the Classification Agent requesting for the evaluation of the abnormal instance 10.0.0.51 connecting at port 1345. In real case scenarios, it could be expected that the 'Content' field of the EVALUATE message will carry a number of attributes of the instance.

(EVALUATE

Sender IP: 10.0.0.3

Sender Port: 1000

Receiver IP: 10.0.0.5

Ontology: Agent Communication

Language: KQML

Content: Abnormal: 10.0.0.51: 1345)

REPORT: On receiving the EVALUATE message from the Host Agent, the Classification Agent classifies the instance into a known attack type. It also looks at other environmental situations such as different hosts complaining about the same instance which the Classification Agent diagnosed into different attack types, and then it tries to capture the nature of the attacker and derives a ‘policy’. The Classification Agent employs this performative to report this policy to the Manager Agent. The ‘Content’ field of the following message shows how a policy is represented by the Classification Agent, which it concluded 10.0.0.51 is to be blocked.

(REPORT
Sender IP: 10.0.0.5
Sender Port: 2000
Receiver IP: 10.0.0.6
Ontology: Agent Communication
Language: KQML
Content: Policy: Block: 10.0.0.51)

BROADCAST: This performative is utilized by the Manager Agent to broadcast rules to all the Classification Agents which it derived based upon the policies received from different Classification Agents. The Classification Agents also employ the same performative to broadcast the rule from the Manager agent to the Host Agents. The example below shows the rule ‘Disconnect’ for IP 10.0.0.51 as derived by the Manager Agent.

(BROADCAST
Sender IP: 10.0.0.6
Sender Port: 3000
Ontology: Agent Communication
Language: KQML
Content: Rule: Disconnect: 10.0.0.51)

REQUEST: This performative is used by the Host Agent to request the Manager Agent to open another channel which can be used to transfer the logs of both ‘normal’ and ‘abnormal’ instances from the Host Agent to the Manager Agent. The Manager Agent can then log these into the database for the re-training of the classifier later on. The ‘Content’ field of the REQUEST message requests for opening the log channel by sending ‘log-ch_open’.

(REQUEST

Sender IP: 10.0.0.3

Sender Port: 1000

Receiver IP: 10.0.0.6

Ontology: Agent Communication

Language: KQML

Content: log-ch_open)

REPLY: The Manager Agent uses this performative to reply to the Host Agent’s request for another channel, indicating whether it accepts the request or not. The Manager Agent indicates ‘log-ch_accept’ and ‘log-ch_reject’ respectively in the ‘Content’ field of this message.

(REPLY

Sender IP: 10.0.0.6

Sender Port: 3000

Receiver IP: 10.0.0.3

Ontology: Agent Communication

Language: KQML

Content: log-ch_accept)

READY: Upon receiving the REPLY message from the Manager Agent, the Host Agent checks if the Manager Agent accepts the request. If it accepts, then the Host Agent sends a READY message notifying that it is ready for transfer. This is indicated by 'log-ch_ready' as the 'Content' field value of the message as shown below.

```
(READY
Sender IP: 10.0.0.3
Sender Port: 1000
Receiver IP: 10.0.0.6
Ontology: Agent Communication
Language: KQML
Content: log-ch_ready)
```

RESPONSE: In reply to the READY message of the Host Agent, the Manager Agent sends the RESPONSE message to indicate that it is ready to start the transfer. As soon as the Host Agent receives a positive response in this message from the Manager Agent, it opens another channel and sends all the logs. As can be seen from the following message, the 'Content' field has the value 'log-ch_start'.

```
(RESPONSE
Sender IP: 10.0.0.6
Sender Port: 3000
Receiver IP: 10.0.0.3
Ontology: Agent Communication
Language: KQML
Content: log-ch_start)
```

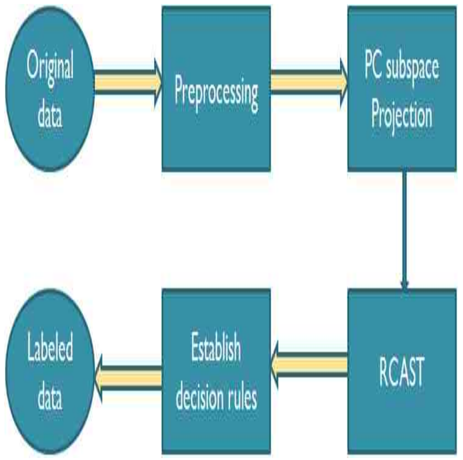


Figure 3.2: Principal Component Classifier

3.3 Supervised Classification

As mentioned above, the proposed agent based architecture was supported by two classification schemes as presented below:-

3.3.1 Principal Component Classifier (PCC)

In the classification module of C-RSPM, each classifier is called the Principal Component Classifier (PCC) [Xie06] (as shown in Figure 3.2). PCC will classify a connection as either normal (non-intrusion) or abnormal (possible intrusion). PCC basically goes through four basic steps of classification: (i) preprocessing, (ii) Principal Component Subspace Projection, (iii) Automatic Representative Component Selection, and (iv) Establishment of the Decision Rules.

In the preprocessing step, the average and standard deviation of the instances in the normal class are calculated, and these statistical characteristics are used to normalize the data. Let L be the set of normalized training data instances, $i = 1, 2, \ldots, p$, $j = 1, 2, \ldots, L$, $\bar{\mu}_i$ and s_{ii} be the sample mean and the variance of i^{th} row of the trimmed matrix $\mathbf{X}$ respectively, and $\mathbf{x}_{ij}$ (i=1,2,...,p, j=1,2,...,L) be the elements in matrix $\mathbf{X}$. Define the normalized untrimmed data set consists of p features as shown in Equation (3.1) and its corresponding column vectors as presented in Equation (3.2), where Equation (3.3) is used for normalization. In order to decide the percentage of the data instances that can be regarded as outliers, a Parzen window is used to decide which data instances are retained and which are removed as outliers according to a defined 'rtd' factor. The 'rtd' factor is chosen corresponding to the center of a Parzen window where the maximum accuracy is reached. If there is a tie, then the first one is chosen as the default one [Xie06].

$$\begin{aligned} \mathbf{Z} &= \{\mathbf{z}_{ij}\}, \ i = 1, 2, \ldots, p, \ j = 1, 2, \ldots, L. && (3.1) \\ \mathbf{Z}_j &= (\mathbf{z}_{1j}, \mathbf{z}_{2j}, \ldots, \mathbf{z}_{pj})', j = 1, 2, \ldots, L. && (3.2) \\ \mathbf{z}_{ij} &= \frac{\mathbf{x}_{ij} - \bar{\mu}_i}{\sqrt{s_{ii}}}. && (3.3) \end{aligned}$$

Next step is to automatically select the representative principal components (PCs). In the PC space, only those dimensions that have positive eigenvalues are selected. Before this, we need to perform the projection for the data instances from the original space to the PC space. That is, each retained training data instance is projected to a subspace by using those PCs derived from the normal class. Let $\mathbf{E}_i = (e_{i1}, e_{i2}, \ldots, e_{ip})'$ be the i^{th} eigenvector, and $(\lambda_1, \mathbf{E}_1)$, $(\lambda_2, \mathbf{E}_2)$, $\ldots$, $(\lambda_p, \mathbf{E}_p)$ be the p eigenvalue-eigenvector pairs of the robust correlation matrix $\mathbf{S}$. Also, let $\mathbf{Y}$ be the projection of $\mathbf{Z}$ onto the p-dimensional eigenspace (shown in Equation (3.4)) consisting of $\mathbf{Y}_j$ column vectors (given in Equation (3.5)). Then, a sample score value of the normalized training data instance vector can be computed using Equation (3.6).

$$\begin{aligned}
\mathbf{Y} &= \{\mathbf{y}_{ij}\},\ i = 1, 2, \ldots, p\ j = 1, 2, \ldots, L. && (3.4)\\
\mathbf{Y}_j &= (\mathbf{y}_{1j}, \mathbf{y}_{2j}, \ldots, \mathbf{y}_{pj})',\ j = 1, 2, \ldots, L. && (3.5)\\
\mathbf{y}_{ij} &= \mathbf{E}_i{}'\mathbf{Z}_j = e_{i1}\mathbf{z}_{1j} + e_{i2}\mathbf{z}_{2j} + \ldots + e_{ip}\mathbf{z}_{pj}. && (3.6)
\end{aligned}$$

We define the p score row vectors of $\mathbf{Y}$, representing the distribution of the p eigenspace features of all the untrimmed, normalized, and projected training instances, as $\mathbf{R}_i$=($\mathbf{y}_{i1}$, $\mathbf{y}_{i2}$, ..., $\mathbf{y}_{iL}$), i=1,2,...,p.
Following this, a lambda score value is computed for each PC.

In the attempt to generate a better predictive model, the set of available score row vectors is refined by eliminating those possessing extremely insignificant or null variability, i.e., extremely little standard deviation. Next, a distance measure is defined as shown in Equation (3.7).

$$\mathbf{c}_j = \sum_{m \in M} \frac{(\mathbf{y}_{mj})^2}{\lambda_m} \tag{3.7}$$

Finally, the classification decision rules can be established, where the decision rules to classify each of the data instances $\mathbf{X}'_j$, j=1,2,...,N', are based on the selected threshold value $\mathrm{C}_{\mathrm{thresh}}$. The details of the steps of establishing the decision rules and determination of $\mathrm{C}_{\mathrm{thresh}}$ can be found in [Quirino06]. In summary, we classify the j^{th} testing data instance as abnormal if $\mathbf{c}'_j > \mathrm{C}_{\mathrm{thresh}}$.

3.3.2 (C-RSPM)

Among various data mining techniques, *supervised classification* has become an essential tool that has been applied successfully in diverse research areas including *network intrusion detection systems.* Since it shows promising results with a number of data sets as compared to other available algorithms, it is utilized in our proposed framework. The system architecture C-RSPM classifier [Quirino06] is illustrated in Figure 3.3. As can be seen from this figure, it includes the *Classification* module and *Ambiguity Solver* module.

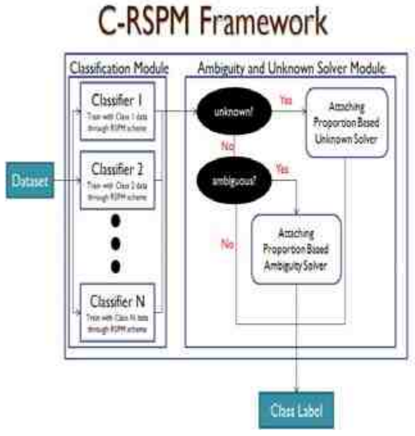

C-RSPM Framework
Classification Module
Ambiguity and Unknown Solver Module
Classifier 1
Train with Class 1 data through RSPM scheme
Classifier 2
Train with Class 2 data through RSPM scheme
Classifier N
Train with Class N data through RSPM scheme
Dataset
unknown!
Yes
No
Attaching Proportion Based Unknown Solver
ambiguous?
Yes
No
Attaching Proportion Based Ambiguity Solver
Class Label

Figure 3.3: Collateral Representative Subspace Projection Modeling (C-RSPM) for Supervised Classification

The Classification module is composed of an array of deviation classifiers (i.e., one PCC for each class) which are executed collaterally. That is, each of the classifiers receives and classifies the same testing instance simultaneously [Quirino06]. The basic idea of the C-RSPM classifier is that each classifier is trained with the data instances of a known class in the training data set. Thus, training the C-RSPM classifier consists basically of training each individual classifier to recognize the instances of each specific class. In the ideal case, a testing data instance will be classified as 'normal' to only one classifier's training data instances. However, in realistic situations, a testing data instance may be classified as 'normal' by multiple classifiers or none of the classifiers considers it as 'normal'. To address these two scenarios, the *Ambiguity Solver module* is introduced. Ambiguity Solver captures and coordinates classification conflicts and also provides an extra opportunity to improve the classification accuracy by estimating the true class of an ambiguous testing data instance.

There can be one more scenario for class ambiguity, which arises with a simple fact that no classifier can ensure the 100% classification accuracy, as in most cases, a data set would contain classes with very similar properties. Thus it makes it difficult for a classifier to identify the correlative differences between these classes with small feature differences.

Unlike many other algorithms which upon encountering such issues simply resort to solutions such as the random selection of a class label from among the ambiguous class labels, the C-RSPM classifier attempts to properly address the issues by defining a class-attaching measure called *Attaching Proportion* for each of the ambiguous classes. The goal of solving such an ambiguity issue via attaching proportion measure is to label an ambiguous testing data instance with the label of the classifier exhibiting the lowest *Attaching Proportion* value. Additionally, the Attaching Proportion can be viewed as a measure of the degree of normality of a data instance with respect to a class, indicating the percentile of normality the data instance under analysis is associated to the corresponding class.

3.4 Agent's Decision Making

3.4.1 Policy Derivation at Classification Agents

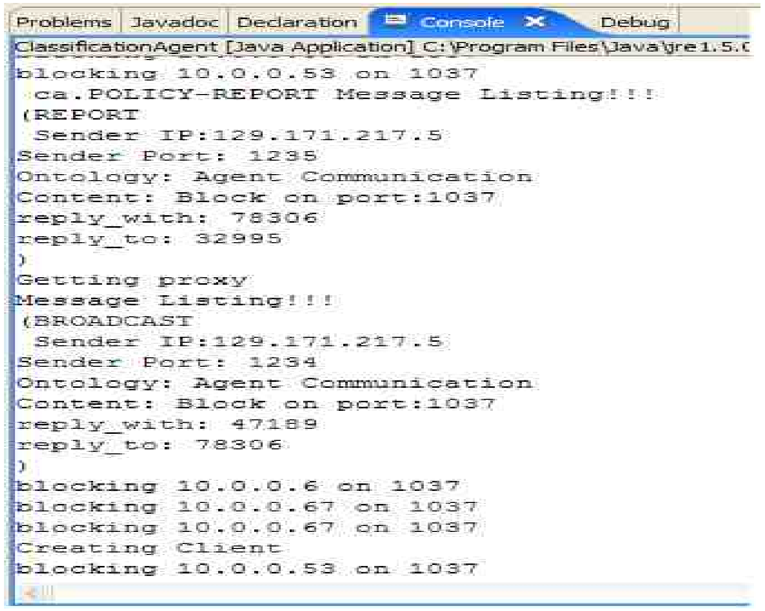
Ideally, an IDS should aim at not only detecting the intrusions but also stopping them as soon as they are detected. The proposed architecture takes this requirement of the IDSs into consideration as well. As mentioned above, the Classification Agent derives a policy after it concludes the type of the abnormal connection, and then reports this policy, rather than the attack type, to the Manager Agent.

Policy derivation plays a very important role in the scenarios when the same attacker generates different attacks to different hosts in the same cluster at the same time instant or at different time instants. If the Classification Agent just reports the attack types, it does not serve the purpose, as the Manager agent may recommend a common measure for all different attacks at the different hosts. In such situations, the policies prove their importance since they provide the solutions to the ongoing attacks. This could be better explained with the example as follows. If an attacker A makes one attack called 'attack1' at host HA1, the other attack called 'attack2' at host HA2, and both hosts belong to Classification Agent CA. Understanding the properties of 'attack1', it is enough to block the access of the attacker to a particular port; whereas for 'attack2', it may be required to block the access of the attacker to a particular server in the network. It is very much expected that in real case scenarios, we cannot just go blocking each doubtful machine. Hence, it is required to have case specific solutions, and the policies can well provide acceptable solutions for this. Policies add more dynamics and adaptability to an IDS by providing attack specific solutions.

To demonstrate how such types of policies can be established, in experiments, three types of policy options are defined. Of course, more policies can be defined to accommodate different network intrusions. Following gives the three pieces of policies that are defined in the experiments.

1. Block;
2. Block on a port; and
3. Block on a server.

As soon as the Classification agent concludes the attack type, it looks at the features of the attacks, chooses one of the three policy options, and then reports this



```
ClassificationAgent [Java Application] C:\Program Files\Java\jre1.5.
blocking 10.0.0.53 on 1037
 ca.POLICY-REPORT Message Listing!!!
(REPORT
 Sender IP:129.171.217.5
Sender Port: 1235
Ontology: Agent Communication
Content: Block on port:1037
reply_with: 78306
reply_to: 32995
)
Getting proxy
Message Listing!!!
(BROADCAST
 Sender IP:129.171.217.5
Sender Port: 1234
Ontology: Agent Communication
Content: Block on port:1037
reply_with: 47189
reply_to: 78306
)
blocking 10.0.0.6 on 1037
blocking 10.0.0.67 on 1037
blocking 10.0.0.67 on 1037
Creating Client
blocking 10.0.0.53 on 1037
```

Figure 3.4: Bandwidth Consumption of Proposed Intrusion Detection System

to the Manager Agent. Figure 3.4 shows a screen dump where the incoming instances are being blocked to the port which has been diagnosed as a penetration point in the network.

CHAPTER 4

Experiments

4.1 Assessment Plan

In order to assess the overall performance of the proposed MAS architecture together with the communication protocol in a realistic scenario, a prototype of the proposed architecture was implemented using Java RMI package, in particular the lipe RMI [lipeRMI09], which allows a number of machines to communicate with each other at the TCP level. Evaluations are conducted in terms of scalability-related criteria such as network bandwidth and system response time for the analysis of implemented protocol and for its performance. Specifically, the experiments focus on (i) the response time characteristics of the proposed architecture in terms of agent communication with supervised classification scheme, and (ii) the accuracy of classification and policy derivation of the agents.

Figure 3.1 shows the network testbed, where the different types of agents were placed in mutually exclusive machines within the testbed in a manner such that any communication among the agents could only be realized through the generation of network traffic rather than local traffic within the same machine. Both the training and testing data sets were acquired from network traffic data generated in the testbed. To evaluate the performance of the proposed framework, the classifiers were trained off-line. Here, ten types of network traffic were generated, which include 5,000 normal connections and 100 connections for each type of abnormal traffic classes. The generated 'normal' connections provide a proper quantity of data and transfer them during a 5-second interval; whereas typical abnormal connections generate a large amount of data in a short span of times (varying for all types of attacks) continuously. For example, connections sending extremely huge packets are used to simulate the 'ping of death' attacks; connections with a lot of packets in a short time duration simulate the 'mail bombing' attacks; connections which try to access the reserved ports are simulated as the 'Trojan infections'; connections transmitting a number of large packets in a short time are used to simulate the 'buffer-overflow' cases, etc.

For processing these data sets, a number of tools were employed. As mentioned above, JAVA's JPCAP [Jpcap09], JAMA [Jama09] and JADE [Jade09] packages were used in the program to capture the packets from the network interface card, and TCP trace [Tcptrace] was used to transform these packets into data instances. This traffic was generated using a traffic generator, which is capable of generating a number of myriad attacks by simply varying its input parameters. The TCP trace extracts 88 attributes from each TCP connection like elapsed time, number of bytes, and segments transferred on both ways, etc. Another feature extraction [Shyu05] technique to extract 46 features from the output of TCP trace, which are useful for the proposed architecture. These features include some basic, time based, connection-based, and ratio-based network features. Out of these 46 features, 14 features were utilized for experiments.

The focus of the testbed based experiments is on network attacks based on the TCP network protocol, since a great majority of the attacks are either executed via or rely on a certain degree on the TCP protocol. This is due mostly to the TCP's frangibility and instability. Please note, however, that the proposed architecture is not limited to the detection of simply TCP based network attacks. The network protocol is simply one of many categorical features employed to describe a network connection.

4.2 Experimental Set-up

Let's first focus on the experimental set up of our testbed based hybrid layered distributed multiagent intrusion detection system together with the communication protocol in a realistic scenario. Evaluations were conducted in terms of scalability-related criteria such as network bandwidth and system response time for the analysis of protocol performance. Figure 3.1 shows our *HAN-IDS* network testbed, where the different types of agents were placed in mutually exclusive machines within the testbed, in a manner such that any communication among the agents could only be realized through the generation of network traffic rather than local traffic within the same machine. Here its ensured that every conversation between the agents would generate measurable network traffic and yield realistic scalability results. These machines were connected to each other via a router. The Host and Classification agents were simulated on Linux OS and the Manager Agent was executed on Windows OS. The Host Agents were all assigned to different interfaces with different IPs on Linux OS by the use of the 'ifconfig' command and specifying virtual interfaces with virtual IPs. This allows us to simulate up to 100 host agents on the same machine.

```
ifconfig eth0:1 10.0.0.1
ifconfig eth0:2 10.0.0.2
ifconfig eth0:3 10.0.0.3 etc.
```

The same procedure was employed for the Classification Agents. Five Classification Agents were simulated on another machine running Linux OS. Having decided on the location of the agents within the testbed, a realistic experimental scenario was devised to capture the effects that the communication between an increasing number of Classification and Host Agents would have on the traffic requirements, and consequently on the scalability performance of the proposed architecture. In summary, several experiments consisting of instantiating an increasing number of the classification and host agents to simulate an increasing IDS network were conducted, in a manner that would reflect the performance of the architecture as it was expanded from a small scale to a large scale. That is, the experiments included a total of 5 * 10 =50 cases with one Manager Agent and 1 to 5 classification agents, each with 10, 20, . . . , 100 host agents. Therefore, the maximum number of agents is 506, simulating a realistic network of 500 Hosts, 5 Classification Agents, and one Manager Agent connecting to the a network. For all these experiments, two performance metrics were observed, namely the average bandwidth (in Mbps) and the response time (i.e., the time when an abnormal instance is detected by a Host Agent to the time when the last BROADCAST message is received by the Host Agent). Both of these features were analyzed by using the packet capture tool called *Wireshark Network Protocol Analyzer* [Wireshark09]. As most of the network traffic flows are between the Host and Classification Agents, Wireshark was executed in all of the Host and Classification Agents. Also, not many changes were identified in the figures on the machine running the Manager Agent as the communication between the Manager Agent and the Host Agent or the Manager and Classification Agents is almost constant. The results which were captured from the network simulation were visualized using MATLAB [matlab09].

For these experiments, data from the testbed was used including, 10 types of network traffic data were generated, which consists of 5000 normal connections and 100 abnormal connections for each type of abnormal traffic. Typical normal connections generate a proper quantity of data and transfer them during a moderate time period in a suitable rate, frequency, and pace. Typical abnormal connections generate a large amount of data and transfer them during a quite short time period continually. For example , connections transport extremely large amount of data are used to simulate the attacks like ping of death, numerous connections in a short time period can be mail-bombing, the connections which try to access reserved ports can be Trojans, and similarly, the connections with a lot of large packages in a short time period are used to simulate buffer-overflow, etc.. For processing these network data, a number of tools were utilized such as Windump [Windump] and TCP dump [Tcpdump].

Windump captures the data directly from the network card. Tcpdump follows by analyzing the windump output and extracts 88 features form each TCP connection, where the extracted features include elapsed time and, the number of bytes and segments transferred. Next, another feature extraction tool is used to further extract 46 features which can be utilized for network intrusion analysis. These features include some basic, time based and ratio based features. Out of these 46 features 43, features are numerical and are used by C-RSPM [Quirino06].

Having all tools installed in all of the machines in the testbed, the traffic generator was executed on one of the machines serving as the sender and another machine as the receiver where the Host Agent is running. This Host Agent captures the incoming packets by reading them from the network card in the machine and caches them in the memory at the interval of every 5 seconds. Next, it calls the TCP trace software which reads the dump files and converts them to the data instances. Having the data instance from the TCP trace, the Host Agent calls the feature extraction tool to extract the required features from the data instance (i.e., a network connection), the Host Agent classifies it using PCC.

All the classification parameters for each of the classes are saved into text files in the training phase. This helps us classify data instances in real time. PCC [Xie06] classifies the data instances only in 'normal' and 'abnormal' classes and returns this label to the Host Agents. If the data instance is classified as 'normal', then it is saved with the Host Agent. However if it is classified as 'abnormal', then the Host Agent further sends it to its Classification Agent on another machine using the 'EVALUATE' message. Upon receiving the 'EVALUATE' message with the abnormal data instance information, the Classification Agent calls C-RSPM [Quirino06] to further analyze this data instance to classify it to a known attack type. This label is used by the Classification Agent to derive its policy by looking at the features of that attack. The Classification Agent then places this policy into the 'REPORT' message and connects it to the Manager Agent which is located on another different machine. Upon receiving the message from the Classification Agent, the Manager Agent broadcasts it to all the Classification Agents in the network which further send it out to its Host Agents using the 'BROADCAST' message.

CHAPTER 5

Results and Analysis

There are several reasons to believe that the proposed architecture and the communication protocol improve intrusion detection performance. It is known that two of the most important elements of network performance are bandwidth and latency. Hence bandwidth and latency are used as the metrics for evaluating the performance of the system. Bandwidth is the transmission capacity of the network, usually measured in bits per second. Frequency is the number of bytes transferred between agents and the maximum rate at which information can be exchanged in a network. Network latency is the amount of time it takes for a packet to travel from the source to the destination. Latency is used to describe the amount of time it takes from the moment an attack takes place till it gets resolved. That is, the round trip time of the proposed communication protocol is considered, including the transfer and processing times of the messages. Latency is of interest in systems with real-time constraints.

Since the layered architecture is adopted, hierarchical heuristics reduce the time complexity in logarithmic manner. In a layered structure, it allows means-ends heuristic which reduces the complexity dramatically. As explained in [Weiss01], if it is assumed that a hierarchy divides the problem of size 'n' into problems each of size 'k', yielding n/k sub-problems, each of which requires $f(k)$ time to solve. These solutions are fed to the next level up in the hierarchy such that 'k' is given to each of agents in this level. Each of these n/k^2 agents has to synthesize 'k' results, again requiring $f(k)$ time. This aggregation process continues up the hierarchy, such that at the next to the topmost level, n/k^{l-1} agents combine 'k' results from below in the hierarchy with 'l' levels. The topmost agent then combines these n/k^{l-1} results together, requiring $f(n/k^{l-1})$ time. The total expenditure is as follows.

$f(n/k^{l-1}) + (n/k^{l-1} \times f(k)) + (n/k^{l-2} \times f(k)) + \ldots + (n/k \times f(k))$

Since 'k' is a constant, and we can choose $l = log_k n$, the equation can be reduced to $O([(k^l - 1)/(k-1)]f(k))$ which can be simplified to $O(n)$. More importantly, if each level of the hierarchy has the agents that solve their sub-problems in parallel, then the time needed below the top of the hierarchy is simply $f(k)$ for each level, so $(l-1)f(k)$. This is added to the top agent's calculation $f(n/k^{l-1})$.

Again since k is constant and $l = log_k n$, this reduces to $O(log_k n)$. This means that through decomposition and parallel problem solving, the exponential problem can be reduced to logarithmic time complexity [Weiss01]. This is one of the main advantages of the proposed architecture. In a multiagent system, latency is particularly important in real-time domains where small changes in latency may mean the difference between success and failure.

5.1 Performance of Communication Protocol

In this subsection the performance of proposed communication protocol is evaluated. In ideal case scenarios, when the network related delays and other network constraints are ignored, its performance such as awareness time and scalability should be linear and dependent only on the number of Classification Agents in the network and the number of Host Agents in each of their clusters. The time taken for the Manager Agent to send rules to the Classification Agents is usually less than what it will take for the Classification Agents to warn the Host Agents in their cluster since there are a large number of Host Agents per Classification Agent.

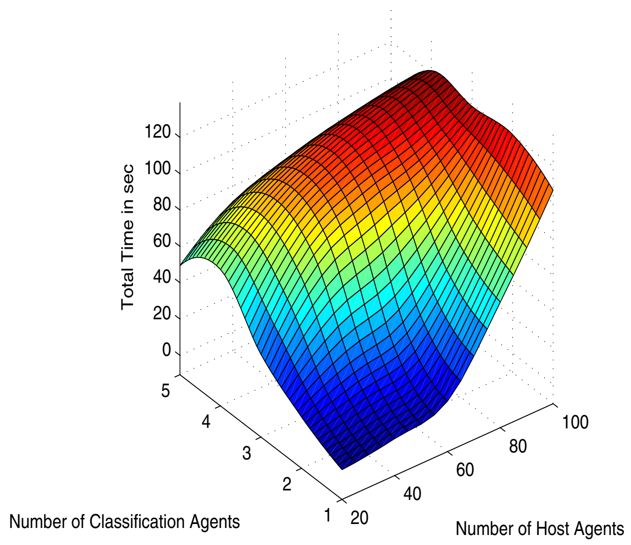


Figure 5.1: Plot of the ***Response Time*** for the proposed communication protocol

Figure 5.1 shows the time plot that can be used to resemble the plane with the number of agents. Linearity of the plot indicates that the system scales linearly in terms of the attack response time, which is a desirable feature. From this figure, it can be seen that the highest response time is about 95 seconds corresponding to the use of 100 host agents with 5 classification agents. These are favorable and substantial results as it demonstrates that the proposed architecture together with the proposed communication protocol produces promising results.

Next, the bandwidth performance of the proposed communication protocol is considered. A regular average sized LAN is first evaluated, where the required bandwidth is at least 10 Mbits/sec. Figure 5.2 shows the experimental results for bandwidth con-

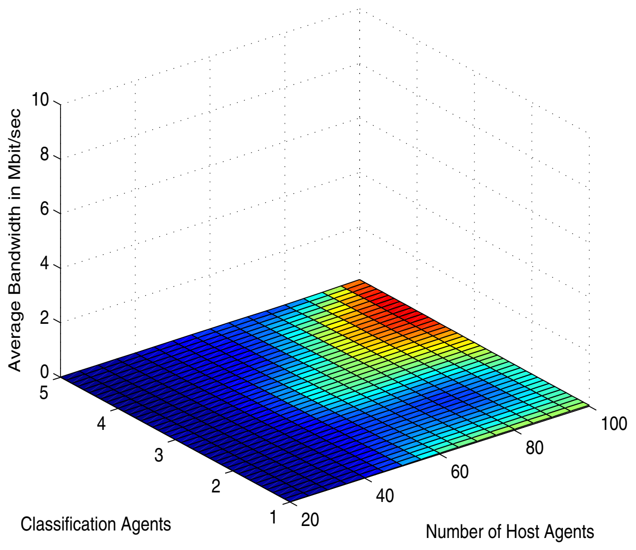


Figure 5.2: Plot of ***Communication Bandwidth*** used by the communication protocol vs a regular network with 10 Mbits/sec

sumption of the communication protocol, where the values for bandwidth consumed for almost all the experiments were less than 1 Mbit/sec, which is almost negligible. As can be seen from this figure, the required bandwidth touches the ground as it consumes negligible bandwidth.

Finally, we look at the zoomed view of the bandwidth performance in Figure 5.3. Ideally, the bandwidth usage of the communication protocol should linearly increase with the number of agents connecting to the network and this is well depicted by the performance of the protocol. Here, the maximum bandwidth consumed is 0.69 Mbits/sec which is absolutely small. Even in the order of magnitude of 506 agents, the system is still efficient in terms of its bandwidth performance. This demonstrates that the proposed communication protocol enables information exchange with low overhead and system scalability. Also, the steady curve in the bandwidth consumption is probably the result of retransmission of lost packets, CPU delays, noise, etc. This, in other words, can be considered as the variability in the network conditions, which

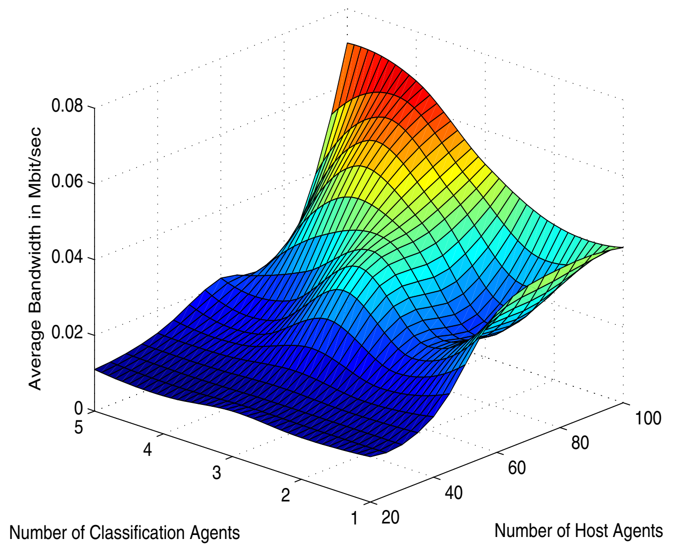


Figure 5.3: Plot of a zoomed view of Figure 5.2

is always a case in real network scenarios. However, since the communication protocol consumes very low bandwidth, even those network problems like noise, losses, and retransmissions do not extremely derogate its performance. As we know, nowadays the major concern of networked systems is the limitation on the bandwidth, and thus maintaining a low bandwidth consumption of bandwidth is a desirable feature. Furthermore, for most of the existing IDS architectures, when the number of attacks increases, the system becomes slow and finally freezes. In the proposed architecture, the performance of the system will not deteriorate too much with the increase in the number of attacks, which is justified by its low bandwidth consumption behavior.

5.2 Performance of C-RSPM

The performance of C-RSPM is also studied. As mentioned earlier, C-RSPM is capable of performing high accuracy supervised classification and outperforms many other classification algorithms [Quirino06]. C-RSPM has shown excellent performance with the testbed data, that it achieved 100% accuracy for some of the attack data generated from the testbed, and 99.97% on average for other data with the standard deviation varying only up to 0.09. It also achieved high classification accuracy with the KDD CUP 1999 data set [Quirino06].

Table 5.1 shows the accuracy of C-RSPM for both data sets and their corresponding standard deviations (as indicated in parentheses) for the 10-fold cross-validation experiments. As can be seen from this table, C-RSPM achieves quite promising classification accuracy for each class. This was motivating enough to integrate it with

Table 5.1: Classification accuracy of C-RSPM (where the standard deviations are given in parentheses)

KDD99	Accuracy	Testbed	Accuracy
normal	99.41(0.16)	normal	99.82(0.14)
back	100.00(0.00)	attack1	99.91(0.14)
neptune	99.99(0.01)	attack2	99.97(0.03)
pod	99.94(0.01)	attack3	99.96(0.09)
smurf	99.58(0.08)	attack4	99.97(0.03)
teardrop	100.00(0.00)	attack5	99.95(0.05)
guess-passwd	99.98(0.01)	attack6	100.00(0.00)
warezclient	99.65(0.06)	attack7	99.96(0.04)
ipsweep	99.11(0.05)	attack8	99.98(0.03)
nmap	99.11(0.06)	attack9	99.97(0.03)
portsweep	99.94(0.01)		
satan	99.88(0.00)		

this agent system and build the proposed architecture.

5.3 Overall Performance of proposed Architecture

5over this setup were conducted in order to evaluate its performance in

The previous subsections have proven the promising performance of the communication protocol and C-RSPM. Here, the performance of the entire architecture together will be evaluated. As described above, the attack data for 5900 instances

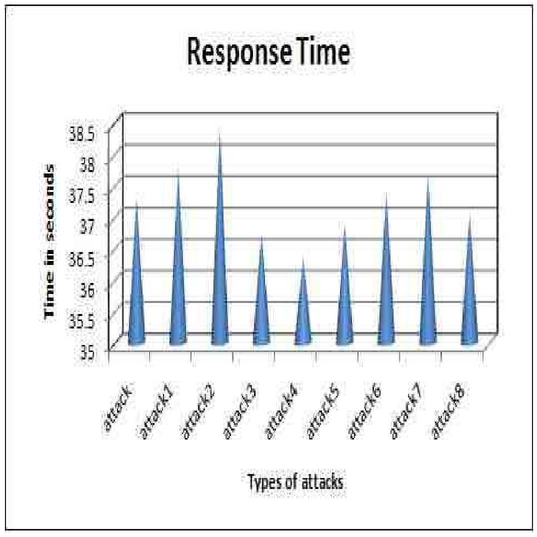


Figure 5.4: Response Time of Proposed Intrusion Detection System

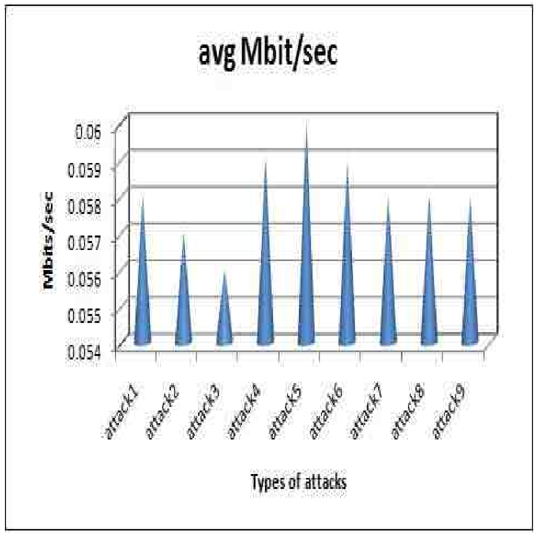


Figure 5.5: Bandwidth Consumption of Proposed Intrusion Detection System

was generated and each of these instances was classified in real time, being captured from the network data card of the machine and then going through each agent layer and respective classifiers. Figure 5.4 shows the response times required by each of the 100 instances continually going through the whole cycle of classification and communication, with respect to by the attack types. As can be seen from this figure, the data instances from all types, on average, result in very similar and low response time irrespective of the attack types. This is definitely a desirable feature. Also, the longest time taken is 40 seconds in the experiments, which shows that the proposed architecture is promising in propagating the information of an ongoing attack within 40 seconds after it has been detected as an intrusion. The time is measured from when an attack is detected and until the last 'BROADCAST' message has reached the Host Agent. As depicted in Figure 5.5, the maximum bandwidth consumed by

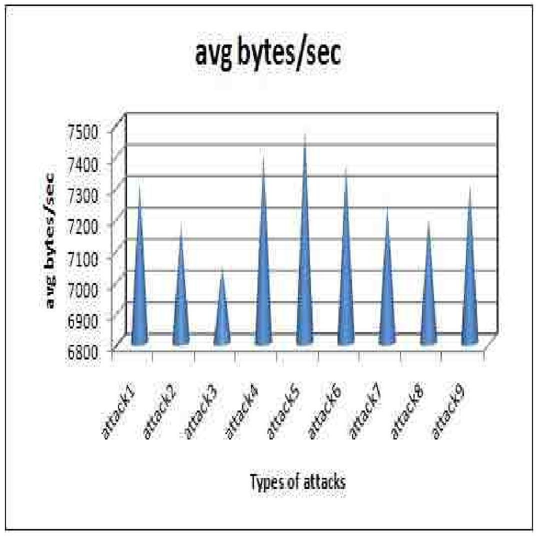


Figure 5.6: Number of bytes per second transmitted from the Host Agent to the Classification Agent

the system is 0.06 Mbits/sec, which is very low as well. This proves that with the increase in number of attacks, the system still results in considerably fair performance. Also, if more machines are connected to the network, the proposed architecture can still withstand the load and deliver the results.

CHAPTER 6

Applications

At this stage, we will see how the proposed architecture also serves another important purpose of controlling automated runs in the domain of hurricane research and analysis.

6.1 Domain Introduction

A tropical cyclone is a storm system characterized by a large low pressure center and numerous thunderstorms that produce strong winds and flooding rain. Tropical cyclones feed on heat released when moist air rises, resulting in condensation of water vapor contained in the moist air. They are fueled by a different heat mechanism from other cyclonic windstorms such as European windstorms, and polar lows, leading to their classification as "warm core" storm systems.

The term "tropical" refers to both the geographic origin of these systems, which form almost exclusively in tropical regions of the globe, and their formation in Maritime Tropical air masses. The term "cyclone" refers to such storms' cyclonic nature, with counterclockwise rotation in the Northern Hemisphere and clockwise rotation in the Southern Hemisphere. Depending on its location and strength, a tropical cyclone is referred to by many other names, such as *hurricane*, typhoon, tropical storm, cyclonic storm, tropical depression, and simply cyclone.

A hurricane or typhoon (sometimes simply referred to as a tropical cyclone, as opposed to a depression or storm) is a system with sustained winds of at least 33 meters per second (64 knots) or 74 miles per hour (119 km/h). A cyclone of this intensity tends to develop an eye, an area of relative calm (and lowest atmospheric pressure) at the center of circulation. The eye is often visible in satellite images as a small, circular, cloud-free spot. Surrounding the eye is the eyewall, an area about 16 kilometers (9.9 mi) to 80 kilometers (50 mi) wide in which the strongest thunderstorms and winds circulate around the storm's center. Maximum sustained winds in the strongest tropical cyclones have been estimated at about 85 meters per second (165 knots) or 195 miles per hour (314 km/h).

In its most common design, hurricane researchers usually first download data from the central servers where all the data resides after being collected from satellites, radars, and other sources. The data is usually atmospheric data which is collected periodically from different sources. Further, the research organizations take the data and do research and analysis, trying to find patterns in the data and then reasons behind them. Such research and analysis also facilitate forecasting capabilities at later stage. Various models have been developed and are being still developed by the hurricane scientists who process and study these data sets. They try to identify the patterns and nature of the hurricanes and other tropical storms, which thereby makes it easy for them to forecast the future hurricane activities in a better way.

6.2 Literature Review

A number of hurricane models and systems are in place today and are being used at National Hurricane Center (NHC) around US and world. The best existing hurricane forecasting models are "global" models that solve the mathematical equations governing the behavior of the atmosphere at every point on the globe. Models that solve these equations are called "dynamical" models. The four best hurricane forecast models (GFDL, GFS, UKMET, and NOGAPS) are all global dynamical models. These models take several hours to run on the world's most advanced supercomputers. There are also dynamical models that cover just a portion of the globe. These are less useful, unless the hurricane happens to start out inside the domain the model covers and stays there. Hurricanes moving from outside the model domain into the model domain are not well handled with such models. An example of this kind of model is the NAM model covering North America and the surrounding waters, run by the National Weather Service (NWS) [NWS]. National Hurricane Center NHC [NHC] uses three types of mathematical models: Statistical, Dynamical or a combination (Statistical-Dynamical).

Statistical Models forecast the future by using current information about a tropical cyclone and comparing it to the historical behavior of similar storms. The historical record for storms in the north Atlantic begins in 1871, while the record for storms for the east Pacific extends back to 1945.

Dynamical Models use the results of global atmospheric data to forecast tropical cyclone motion and intensity. Global models take current wind, temperature, pressure, and humidity observations and make forecasts of the actual atmosphere in which the cyclone exists.

Statistical Models are defined by their simplicity; while Dynamical Models are hampered by incomplete data sets and lack of computation. Predictions from both computer models are only approximate. Statistical Models do not directly include current atmospheric conditions, and Dynamical Models omit the historical behavior of storms.

Because of their simplicity, Statistical Models were designed first in the late 1960s for tropical cyclone forecasting. In the early 1970s, Combination Models were developed as global models and began making forecasts in the tropical regions. As computers became more powerful, pure Dynamic Models began dominating the accuracy race. This is particularly true when the tropical cyclones approach the data-rich regions close to the continents, where the state of the atmospheric environment is adequately observed and well known. Furthermore, mathematical models typically have varying initial assumptions, and thus different models produce different final results. They do not necessarily reflect the "official" hurricane track issued by the National Hurricane Center. Forecasters review all of the model data but use their own experience and scientific expertise to arrive at a final forecast. Following subsection provide a summary of the top six models developed in *Hurricane Research Division* of *National Oceanographic and Atmospheric Administration (NOAA)* [NOAA]:

6.2.1 GFDL

The Geophysical Fluid Dynamics Laboratory (GFDL) [GFDL] is engaged in comprehensive long lead-time research fundamental to NOAA's mission. The goal of this research is to expand the scientific understanding of the physical processes that govern the behavior of the atmosphere and the oceans as complex fluid systems.

These systems can then be modeled mathematically and their phenomenology can be studied by computer simulation methods. In particular, GFDL research concerns the following:

1. Predictability of weather on large and small scales;

2. Structure, variability, predictability, stability, and sensitivity of global and regional climate;

3. Structure, variability, and dynamics of the ocean over its many space and time scales;

4. Interaction of the atmosphere and oceans, and how the atmosphere and oceans influence and are influenced by various trace constituents; and

5. Earth's atmospheric general circulation within the context of the family of planetary atmospheric circulations.

The scientific work of GFDL encompasses a variety of disciplines including meteorology, oceanography, hydrology, classical physics, fluid dynamics, chemistry, applied mathematics, and numerical analysis. Research is also facilitated by the Atmospheric and Oceanic Sciences Program (AOSP) which is a collaborative program at GFDL with Princeton University. Under this program, regular Princeton faculty, research scientists, and graduate students participate in theoretical studies, both analytical and numerical, and in observational experiments in the laboratory and in the field. The program is supported in part by NOAA funds. AOSP scientists may also be involved in GFDL research through institutional or international agreements [NHCNOAA].

6.2.2 GFS

The Global Forecast System (GFS) [GFS] is a global spectral and numerical computer model run by NOAA four times a day. It is an excellent model in the one to five day range. The accuracy drops significantly after the fifth day, and significant long-range forecast changes are noted from run to run.

GFS is used for NCEP's [NCEP] global data assimilation system and for the aviation and medium-range forecasts (MRF). As one might guess from its name, the "aviation model" was not specifically developed to predict hurricane motion or intensity. Rather, one of the primary uses of GFS is to produce forecasts for aviation guidance worldwide. The GFS model is run four times each day at the primary and intermediate synoptic times (0000, 0600, 1200 and 1800 UTC) with a wait of 2.75 hr for data arrival. Forecasts are made out to Day 16.

Like the GFDL Hurricane Model (GHM), the GFS model is a primitive equation model which predicts winds, temperature, surface pressure, humidity, and precipitation. The prediction equations include the divergence and vorticity equations, the hydrostatic equation, the thermodynamic equation, a mass continuity equation, and a conservation equation for water vapor.

The GFS model differs from the GHM model in that it has a global domain, and the fields within the model are represented by a set of mathematical (sine and cosine) functions rather than the values at discreet grid points. The forecast equations are solved for the coefficients of the mathematical functions.

Currently, the GFS model is configured to handle 382 triangular waves across the globe (comparable resolution to a grid point model with a grid spacing of 37 km) and has 64 vertical levels. For integrations between 7 1/2 and 16 days, the horizontal resolution is reduced to 190 triangular waves. The loop time steps are 6 hours from analysis time to 240 hours (10 days), and then change to 12-hour time steps out to 384 hours (16 days) [NHCNOAA].

6.2.3 UKMET

The United Kingdom Meteorological Office model (UKMET) [UKMET] is restricted from being redistributed according to the international agreement, and hence graphics from the UKMET model are difficult to find on the web. Only the paying subscribers are supposed to have access to the data from UKMET. UKMET model was completely formulated by U.K, Met Office in year 2002. The new formulation consisted of the model's dynamical core, the fundamental equations, and physical parameterizations. In 2005, physics package of UKMET was again revised. The UKMET typically provides useful tropical cyclone track forecasts. The limitation of this model is to produce valuable intensity forecasts [NHCNOAA].

6.2.4 NOGAPS

The NOGAPS [NOGAPS] model was not designed specifically to predict the motion of tropical cyclones. Rather it is the Navy's operational global atmospheric prediction system. The NOGAPS model is run four times daily every day of the year, producing forecasts out to 144 hours.

The NOGAPS model initializes the tropical cyclone using synthetic soundings based on the National Hurricane Center's estimates of the storm location and intensity. These soundings are automatically inserted into the model in the vicinity of the tropical storm. The artificial atmospheric soundings are constructed at the storm center and at radii of two, four and six degrees from the storm The winds are derived from a specified vortex which has a radius of maximum winds of 50 km. Due to the horizontal spatial resolution of the NOGAPS model, the maximum wind speeds inserted into the model are 60-80 percents of those observed [NHCNOAA].

6.2.5 HWRF

The NWS/Hurricane Weather Research and Forecast Model (HWRF) [HWRF] was developed in 2007, and is scheduled to replace the GFDL by 2009. HWRF is a non-hydrostatic coupled ocean-atmosphere model which will utilize highly advanced physics of the atmosphere, ocean and waves in one prediction system, providing unparalleled understanding of the science of tropical cyclone evolution. Its output gives meteorologists an analysis of the hurricane in three-dimensions from real-time airborne Doppler radar. It will make use of a wide variety of observations from satellites, data buoys, and hurricane hunter aircraft. No other hurricane model accesses such a wide range of meteorological information. The GFDL and HWRF models are the only models that provide specific intensity forecasts of the hurricanes [NHCNOAA].

6.3 Hurricane Research System (HRS)

In this section, we will look at how the proposed architecture can be used for the Automated Hurricane System.

6.3.1 Host Layer

This layer being at the initial level takes care of the data downloading task. Data on the remote server exists in directories and files. Each Host Agent logs into the server at a predefined periods and starts downloading the data sets by creating respective directories locally. The Host Agent checks at every moment which files have downloaded, and continues downloading the ones which are still not locally available. Initially, the Host Agent checks if the same folder exists locally, which it sees on the remote server. If it finds the same folder available locally, it further looks for the files within it. A number of host agents are deployed in the similar manner, which download different data sets as soon as they are available on the remote server.

6.3.2 Classification Layer

As soon as the Host Agent is done with data downloading task, it informs the Classification Agent. Each Classification Agent is now responsible for running the model scripts sequentially. The Classification Agent starts by calling the model's first script 'wps_nmm.sh', and then continually checks the script's progress by logging the timely information into the log files. This way it makes sure that the process went through successfully and has written out the output files necessary for the second process. Once this is finished, it initiates the second script of the HRS model called 'real_nmm.sh'. It repeats the similar steps as for the previous process, i.e., monitors the process and writes logs. Each consecutive script takes previous scripts's output as its input. This follows by the third script called 'wrf_nmm.sh' as well. Once this script is completed, post-processing scripts for two different domains are to be run namely 'postd01_nmm.sh' and followed by the script for the inner domain called 'postd02_nmm.sh'. The Classification Agent is responsible for maintaining synchronization between all these processes. It is also responsible for making sure that each process writes out the desired output files.

6.3.3 Manager Layer

The Manager Agent just like the one for Intrusion Detection System is responsible for the overall management of the entire architecture. It also uploads the outputs to the database in a fixed structure so that they can be easily analyzed later. The Manager Agent is also responsible for maintaining synchronization among all the uploads.

6.4 The Communication

As mentioned earlier, every multiagent system requires a good communication scheme in order to attain its pre-defined set of goals. It holds the same for this case. As soon as one agent finishes its set of tasks, it informs the upper layer agent to start its functions. This is done by using the similar communication scheme as proposed for Intrusion Detection System. KQML is the perfect choice again for this system because it requires quick system functioning and response. As it is proved in the experiments, KQML facilitates faster response times, and thus it is clearly the technique required.

As soon as the host agent finishes downloading the data sets, it sends a KQML message to its designated classification agent informing it about the availability of the data. Upon receiving this notification from the host agent, the classification agent starts its first process and takes care of all the following functions. Once the classification agent finishes its tasks, it informs the manager agent by means of a message that it has finished all the tasks it is responsible for. As soon as the Manager Agent gets this message, it goes ahead with its own responsibilities of storing the data to the correct destination and in the correct format after data processing.

CHAPTER 7

Conclusions and Future Work

In chapters 3, 4 and 5, the proposed framework was presented in detail, and a discussion of its performance under different conditions was provided. In this con- cluding chapter, a summary of the contributions and the performance of the proposed approach is given, followed by a detailed discussion of the future work.

7.1 Conclusions

In this thesis, a novel distributed multiagent IDS architecture is presented, which incorporates the desirable features of the multiagent design methodology with *PCC* and *C-RSPM* classification schemes. This integration of agent's intelligence with speed and accuracy of above mentioned classification schemes, results into a smart and fast Intrusion Detection System (IDS). This solution was preceded by initially laying out the multiagent architecture and facilitating it with a communication protocol. With this system implemented, experiments were conducted utilizing simulated attack data generated from the testbed located at Department of Electrical and Computer Engineering of University of Miami. These experiments proved that the agent architecture and the protocol were light weighted, consumed very less bandwidth and had low response times. Also as proved by experiments, even when a larger number of agents are introduced into the network, the system is still able to communicate using extremely low bandwidth in small amount of time.

Next challenge was to utilize the capabilities of the above developed system in real time. In order to have real time data processing capabilities, some of the fast and accurate classification schemes were required. Therefore, in order to enable HAN-IDS multiagent architecture do the real time processing of network data, C-RSPM [Xie06a] and PCC [Xie06] classification schemes were employed. However, for this integration to be realized several other tools were required. So, a new implementation was done which was able to bring together all these multi-platform tools to a common platform and facilitate the functioning of agent based system together with classification schemes. Having, achieved this integration experiments were conducted to check if both the systems were able to complement each others capabilities and overcome each others limitations.

A key concept in the design of the proposed architecture was to show the integration of the agent technology with the data mining strategies, and to prove how both complement each other. Therefore, from the above discussion it can be seen that the system succeeded in proving that agent technology and data mining integration has long way to go.

7.2 Future Work

As the proposed HAN-IDS multiagent based architecture demonstrates its promising per- formance for real-time applications, there is still much work ahead to be considered. In this section, research enhancement to the proposed approach is presented.

As for now no rule derivation is done by the Manager Agent. This is an integral part of the proposed architecture and must be included as this depicts the agent's decision making criterion thereby dominating its intelligence.

Next, some fault tolerance capabilities are needed for this architecture as it is a layered architecture so malfunctioning or failure of any one of these layers can bring down the entire architecture. Therefore, inclusion of some fault tolerance is an important issue that needs to be worked in future. Adding of one more layer to this architecture can resolve this problem. This layer called *Critic Layer* should be directly linked to each of these existing three layers in concurrent. Critic layer will constantly check for status of each layer and agents in these layers. Its most important responsibility will be to make sure the fault free functioning of the entire system.

Another important consideration for future work should be to take into consideration the other linked networks. At present no outer networks are taken into consideration, only the intranet is considered for checking of intrusions.

Also, for now only one ongoing attack in the network is considered for experiments. Multiple attack scenarios were not considered for checking the system's reliance. In future, multiple attack situations should be considered and system should be improved regardingly.

Bibliography

[Agre87] P.Agre and D.Chapman. Pengi: An implementation of a theory of activity.

[Agre96] P. E Agre and S.J Rosenschein. Computational theories of interaction and agency. In *THe MIT Press*, Cambridge, MA, 1996.

[Anderson95] D. Anderson, T. Frivold, and A. Valdes. Next-generation intrusion detection expert system (NIDES): A summary. *SRI International Technical Report*, 95(7):28–42, May 1995.

[Angiulli05] F. Angiulli and C. Pizzuti. Outlier mining in large high-dimensional data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):203–215, Feb. 2005.

[Assal04] A. Assal and V. Groza. Agent based resource management for smart robotic sensors. In *CCECE-2004*, volume 4, pp. 2239–2242, Niagara Falls, 2004.

[Bace01] R. Bace and P. Mell. *NIST Special Publication on Intrusion Detection Systems.* National Institute of Standards and Technology, 2001.

[Barb01] D. Barbara, N. Wu, and S. Jajodia. Detecting novel network intrusions using Bayes estimator. In *First SIAM Conference on Data Mining*, Chicago, IL, April 2001.

[Barringer89] H. Barringer,M.Fisher, D. Gabbay, G.Gough, and R. Owens. Metatem: A framework for programming in temporal logic. In *Workshop on Stepwise Refinement of Distributed systems*, pp. 94–129, 1989.

[Bayardo97] R.J. Bayardo. Infosleuth: Agent-based semantic integration of information in open and dynamic systems. In *Proc. ACM Sigmod Intl Conf. Management of Data,ACM Press*, New York, 1997.

[Bradshaw97] J.M. Bradshaw et al. Kaos: Toward an industrial- strength open agent architecture,. In *Software Agents, J.M. Bradshaw, ed.,AAAI Press, 1997. 21. M.R. Genesereth, A.M. Keller, and O.M. Duschka, Infomaster: An Information Integration System, Proc. ACM Sigmod Intl Conf. Management of Data,ACM Press*, 1997.

[Briggs95] W.Briggs and D.J. Cook. Flexible social laws. In *In Proceedings of the Fourteenth International Joint Conference on artificial Intelligence(IJCAI-95)*, 1995.

[Brooks86] R.A. Brooks. A robust layered control system for a mobile robot. In *IEEE journal of Robotics and Automation*, pp. 4:349–355, 1986.

[Brooks90] R.A. Brooks. Elephants don't play chess. in p.maes, editor, designing autonomous agents. In *The MIT press.*, pp. 3–15, Cambridge, MA, 1990.

[Brooks91] R.A. Brooks. Intelligence without reason. in proceedings of the twelfth international joint conference on artificial intelligence (ijcai-91). pp. 569–595, Sydney, Australia, 1991.

[Brooks91a] R.A. Brooks. Intelligence without representation. In *Artificial Intelligence*, pp. 47:139–159, 1991.

[Cao04] J. Cao and L. Zhang and J. Yang and S.K. Das. A reliable mobile agent communication protocol. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pp. 468–475, Tokyo, Japan, 2004.

[Chalupsky02] H. Chalupsky and Y. Gil and C.A. Knoblock and K. Lerman and J. Oh and D.V. Pynadath and T.A. Russ and M. Tambe. Electric Elves: Agent technology for supporting human organizations,. *AI Magazine*, 23(2):11–24, 2002.

[Chauhan97] D. Chauhan. Jafmas: A java-based agent framework for multiagent systems development and implementation. In *masters thesis., Electrical Engineering and Computer Science Dept., Univ. of Cincinnati*, Cincinnati, 1997.

[Chauhan98] D. Chauhan and A. Baker. Afmas: A multiagent application development system, proc. second acm conf. autonomous agents,. In *M. Wooldridge and T. Finin, eds.,ACM Press*, 1998.

[Conry91] Susan E. Conry, Kazuhiro Kuwabara, Victor R. Lesser, and Robert A. Meyer. Multistage negotiation for distributed constraint satisfaction. In *IEEE Transactions on systems, Man, and Cybernetics*, pp. SMC–21(6):1462–1477, 1991.

[Corkill82] Daniel D.Corkill. A framework for organizational self-design in distributed problem solving networks. In *Ph D thesis, University of Massachusetts*, 1982.

[Cost98] R.S. Cost et al. Jackal: A java-based tool for agent development. In *Working Papers of the AAAI-98,Workshop on Software Tools for Developing Agents, AAAI Press*, 1998.

[Denning87] D. E. Denning. An intrusion detection model. *IEEE Transactions on Software Engineering*, SE13(2):222–232, Feb 1987.

[Dokas02] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan. Data mining for network intrusion detection. *Proceedings of National Science Foundation Workshop on Next Generation Data Mining*, pp. 21–30, November 2002.

[Edmund87] Edmund H. Durfee, Victor R. Lesser, and Daniel D.Corkill. Coherent cooperation among communication problem solvers. In *IEEE transaction on Computers*, pp. C–36(11):1275–1291, 1987.

[Edmund88] Edmund H. Durfee. Cordination of distributed problem solvers. In *Kluwer*, 1988.

[Edmund90] Edmund H. Durfee and Thomas A. Montgomery. A heirarchical protocol for coordinating multiagent behaviors. In *In Proc. AAAI-90*, 1990.

[Froese03] T. Froese. Steps toward the evolution of communication in a multi-agent system. In *Symposium for Cybernetics Annual Reserch Projects, SCARP'03*, University of Reading, UK, 2003.

[Fuller94] R.M. Fuller, G.B. Groom, and A.R. Jones. Photogramm. engg. *Remote Sensing*, 60:553–562, 1994.

[Garuba08] M. Garuba and C. Liu and D. Fraites. Intrusion techniques:comparative study of network intrusion detection systems. In *Fifth International Conference on Information Technology.* New Generations, 2008.

[Genesereth97] M.R. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: An information integration system. In *Proc. ACM Sigmod Intl Conf. Management of Data,ACM Press*, 1997.

[Georgeff87] M.P. Georgeff and A.L. Lansky. Reactive reasoning and planning. In *In Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pp. 667–682, Seattle, WA, 1987.

[GFDL] GFDL. *Available at http://www.gfdl.noaa.gov/*, 2009.

[GFS] GFS. *Available at http://www.nco.ncep.noaa.gov/pmb/nwprod/analysis/namer/gfs*, 2009.

[Ghosh99] A.K. Ghosh, A. Schwartzbard, and M. Schatz. Learning program behavior profiles for intrusion detection. In *The 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, pp. 51–62, Santa Clara, USA, 1999.

[Hayzelden00] A.L.G. Hayzelden and J. Bigham and S.J. Poslad and P. Buckle and E.H. Mamdani. Communication systems driven by software agent technology. *Journal of Network and Systems Management*, 8(3):321–347, 2000.

[Heady90] R. Heady, G. Luger, A. Maccabe, and M. Servilla. The architecture of a network level intrusion detection system. *Technical report, Computer Science Department, University of New Mexico*, August 1990.

[HWRF] HWRF. *Available at http://www.emc.ncep.noaa.gov/HWRF/index.html*, 2009.

[Ilgun95] K. Ilgun and R. A. Kemmerer and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. In *IEEE Transaction of Software Engineering*, pp. 181–199, 1995.

[Jade09] Java Agent Development Framework. *Available at http://jade.tilab.com/*, 2009.

[Jama09] JAMA. *Available at:http://math.nist.gov/javanumerics/jama/*, 2009.

[Jin05] X.Jin and Y.Zhang and Y.Zhou and Wei Y. A novel ids agent distributing protocol for manets, v.s. sunderan et al. (eds.). In *ICCS 2005*, pp. 502–509. LNCS 3515, 2005.

[Jpcap09] JPCAP. *Available at:jpcap.sourceforge.net/javadoc/index.html*, 2009.

[Kaelbling86] L.P.Kaelbling. An archtecture for intelligent reactive systems. In *Reasoning about actions and plans-Proceedings of the 1986 workshop*, pp. 395–410, San Mateo, CA, 1986.

[Kannadiga05] P.Kannadiga and M.Zulkernine. Didma: A distributed intrusion detection system using mobile agents. In *Proceedings of Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks*, pp. 238–245, 2005.

[Kargupta00] H.Kargupta and B.Park and D.Hershberger and Johnson E. Advances in distributed and parallel knowledge discovery, chapter 5, collective data mining: A new perspective toward distributed data mining. AAAI/MIT Press, 2000.

[Khasteh06] S.H. Khasteh and S.B. Shouraki and R. Halavati and E. Khameneh. Evolution of a communication protocol between a group of intelligent agents. In *World Automation Congress (WAC)*, pp. 1–6, Budapest, Hungary, 2006.

[Kim05] H. Kim, P. Howl, and H. Park. Dimension reduction in text classification with support vector machines. *The Journal of Machine Learning Research*, 6:37–53, September 2005.

[Klusch03] M.Klusch and S.Lodi and G.Moro. The role of agents in distributed data mining: Issues and benefits. In *Proceedings of the IEEE/WIC International Conference on Intelligent Agent Technology (IAT'03)*, 2003.

[Koes04] M.B. Koes and I. Nourbaksh and K. Sycara. Communication efficiency in multi-agent systems. In *Proceedings of ICRA 2004*, pp. 2129–2134, Barcelona, Spain, 2004.

[Kumar95] S.Kumar and E. H.Spafford. A software architecture to support misuse intrusion detection. In *In Proceedings of the 18th National Conference on Information Security.*, pp. 194–204, 1995.

[Labib04] K. Labib and V.R. Vemuri. Detecting and visualizing denial-of-service and network probe attacks using principal component analysis. In *Third Conference on Security and Network Architectures, SAR04*, La Londe, France, June 21-25, 2004.

[Lark95a] R.M. Lark. A reappraisal of unsupervised classification (I): Correspondence between spectral and conceptual classes. *International Journal of Remote Sensing*, 16:1425–1443, 1995.

[Lark95b] R.M. Lark. A reappraisal of unsupervised classification (II): Optimal adjustment of the map legend and a neighbourhood approach for mapping legend units. *International Journal of Remote Sensing*, 16:1445–1460, 1995.

[Laskov05] P. Laskov, P. Dussel, C. Schafer, and K. Rieck. Learning intrusion detection: supervised or unsupervised? In *ICIAP*, Cagliari, ITALY, October 2005.

[Lazarevic03] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Third SIAM Conference on Data Mining*, San Francisco, California, May 2003.

[Lee00] W. Lee and S. J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and Systems Security*, 3(4):227–261, November 2000.

[Lee99] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *IEEE Symposium on Security and Privacy*, pp. 120–132, Santa Clara, USA, 1999.

[lipeRMI09] lipeRMI. *Available at http://lipermi.sourceforge.net/*, 2009.

[Maes89] P. Maes. The dynamics of action selection. in proceedings of the eleventh international joint conference on artificial intelligence (ijcai-89). pp. 991–997, Detroit, MI, 1990.

[Maes90] P. Maes. Designing autonomous agents. In *The MIT press.*, Cambridge, MA, 1990.

[Maes90a] P. Maes. Situated agents can have goals. in p.maes, editor, designing autonomous agents. In *The MIT press*, pp. 49–70, Cambridge, MA, 1990.

[Maes91] P. Maes. the agent network architecture (ana). In *SIGART Bulletin*, pp. 2(4):115–120, 1991.

[Mahoney03] M.V. Mahoney and P.K. Chan. Learning rules for anomaly detection of hostile network traffic. *The Third IEEE International Conference on Data Mining*, pp. 601–604, November 2003.

[Markou03a] M. Markou and S. Singh. Novelty detection: A review part1: Statistical approaches. *Signal Processing*, 83(12):2481–2497, Dec 2003.

[Markou03b] M. Markou and S. Singh. Novelty detection: A review part2: Neural network-based approaches. *Signal Processing*, 83(12):2499–2521, Dec 2003.

[matlab09] Mathworks.2009.MATLAB. *Available at http://www.mathworks.com/matlabcentral/*, 2009.

[McCarthy69] J. McCarthy and P.J.Hayes. Some philosophical problems from the standpoint of the artificial intelligence. In *Machine Intelligence 4. Edinburgh University Press*, 1969.

[Mukkamala02] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *IEEE Internation Joint Conference on Neural Networks*, pp. 1702–1707, Santa Clara, USA, 2002.

[Muller95] J.P. Muller, M. Pischel, and M. Thiel. Modelling reactive behaviour in vertically layered agent architectures. In *Intelligent agents: Theories, Architectures, and Languages*, pp. 261–276, Berlin, Germany, 1995.

[NCEP] NCEP. *Available at http://www.ncep.noaa.gov/*, 2009.

[NHC] NHC. *Available at http://www.nhc.noaa.gov/*, 2009.

[NHCNOAA] NHCNOAA. *Available at http://www.nhc.noaa.gov/modelsummary.shtml*, 2009.

[NOAA] NOAA. *Available at http://www.noaa.gov/*, 2009.

[Nodine97] M. Nodine and A. Unruh. Facilitating open communication in agent systems: The infosleuth infrastructure,. In *Proc. Fourth Intl Workshop on Agent Theories, Architectures, and Languages, M. Singh, A. Rao, and M. Woolridge, eds., Springer-Verlag*, 1997.

[Noel02] S. Noel, D. Wijesekera, and C. Youman. *Applications of Data Mining in Computer Security.* Kluwer Academic Publishers, 2002.

[NOGAPS] NOAGAPS. *Available at http://www.nwmangum.com/NOGAPS.phtml*, 2009.

[NWS] NWS. *Available at http://www.nws.noaa.gov/*, 2009.

[Pahlevanzadeh07] B.Pahlevanzadeh and A.Samsudin. In *Distributed hierarchical IDS for MANET over AODV+*, pp. 220–225. IEEE International Conference on Telecommunications and Malaysia International Conference on Communications, 2007.

[Park94] J.W. Park and H.G. Park and W.H. Kwon. A real time communication protocol with contention-resolving algorithm for programmable controllers. In *20th International Conference on Industrial Electronics, Control and Instrumentation (IECON'94)*, pp. 1159–1164, Brazil, 1994.

[Paxson99] V. Paxson. BRO: A system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[Qui93] Quinlan J.R. C4.5: Programs for machine learning. San Francisco, CA, USA, 1993. Morgan Kaufmann.

[Quirino06] T. Quirino, Z. Xie, M.-L. Shyu, S.-C. Chen, and L. Chang. Collateral representative subspace projection modeling for supervised classification. In *Proc. of the 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI06)*, pp. 98–105, Washington D.C., USA.

[Rao91] A.S. Rao and M.P. Georgeff. Assymetry thesis and side-effects problem in linear time and branching time intention logics. In *In proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pp. 498–504, Sydney, Australia, 1991.

[Rao91a] A.S. Rao and M.P. Georgeff. Modelling rational agents within a bdi-architecture. In *Proceedings of Knowledge Representation and Reasoning*, pp. 473–484, San Mateo,CA, 1991.

[Rao92] A.S. Rao and M.P. Georgeff. An abstract architecture for rational agents. In *Proceedings of Knowledge Representation and Reasoning*, pp. 439–449, 1992.

[Rao92a] A.S. Ra0, M.P. Georgeff and E.A. Sonenberg. Social plans: A preliminary report. In *Proceedings of the Third European Workshop on Modelling Autonomous Agents in a Multi-agent world (MAAMAW-91)*, pp. 57–76, Amsterdam,The Netherlands, 1992.

[Rao93] A.S. Rao and M.P. Georgeff. A model-theoretic approach to the verification of situated reasoning systems. In *In proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pp. 318–324, France, 1993.

[Rao96] A.S. Rao. Agentspeak(l):agents breaking away. In *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multiagent World*, pp. 42–55, Berlin, Germany, 1996.

[Rao96a] A.S. Rao. Decision procedures for propositional linera-timebeleif desire intention logics. In *In M, Wooldridge, J.P. Muller, and M.Tambe, editors, Intelligent Agents*, pp. 33–48, Berlin, Germany, 1996.

[Sainani09] Varsha S. and M.-L Shyu. A hybrid layered multiagent architecture with low cost and low response time communication protocol for network intrusion detection systems. In *The IEEE 23rd International Conference on Advanced Information Networking and Applications, Accepted for publication*, Bradford, UK, 2009.

[Shyu05] M.-L. Shyu and K. Sarinnapakorn and I. Kuruppu-Appuhamilage and S.-C. Chen and L. Chang and T. Goldring. Handling nominal features in anomaly intrusion detection problems. In *the 15th International Workshop on Research Issues on Data Engineering (RIDE-SDMA'2005), in conjunction with ICDE 2005*, pp. 55–62, National Center of Sciences, Tokyo, Japan, April 2005.

[Shyu09] Mei-Ling Shyu and Varsha Sainani. A multiagent-based intrusion detection system with the support of multi-class supervised classification. In *accepted for publication, Edited by Longbing Cao, Data Mining and Multiagent Integration, Springer*, Bradford, UK, 2009.

[Spafford00] E.Spafford and D.Zamboni. Intrusion detection using autonomous agents. computer networks 34, 4. pp. 547–570, 2000.

[Stolfo97] S.Stolfo and A.Prodromidis and S.Tselepis and W.Lee and D.Fan and Chan P. Jam: Java agents for meta-learning over distributed databases. In *Proceedings of KDD-97*, pp. 74–81, Newport Beach, California, USA, 1997.

[Sycara03] K. Sycara and M. Paolucci and M. Van Velsen and J. Giampapa. The RETSINA MAS infrastructure. *Autonomous Agents and Mulit-Agent Systems*, 7(1-2):29–48, 2003.

[Tcpdump] V. Jacobson, C. Leres, and S. McCanne. Tcpdump. *Available at anonoymous@ftp.ee.lbl.gov*, May 2009.

[Tcptrace] Tcptrace software tool. Tcptrace. *Available at http://www.tcptrace.org*, May 2009.

[Tseng05] V.S. Tseng, C.-J. Lee, and J.-H. Su. Classify by representative or associations (CBROA): A hybrid approach for image classification. In *The 6th International Workshop on Multimedia Data Mining: Mining Integrated Media and Complex Data*, pp. 37–53, Chicago, Illinois, USA, August 2005.

[UKMET] UKMET. *Available at http://www.ukmet.net/*, 2009.

[Vaidehi04] K. Vaidehi and B. Ramamurthy. Distributed hybrid agent based intrusion detection and real time response system. In *In Proceedings of the First International Conference on Broadband Networks*, pp. 739–741, 2004.

[Weiss01] G. Weiss. *Multiagent Systems.* The MIT Press, 2nd edition, 2001.

[Windump] Windump. *Available at http://www.windump.com*, 2009.

[Wireshark09] Wireshark. *Available at http://www.wireshark.org/download.html/*, 2009.

[Xie06] Z.Xie and T. Quirino and M.-L.Shyu. A distributed agent-based approach to intrusion detection using the lightweight pcc anomaly detection classifier. In *Proceedings of the IEEE International Conference on Sensor Networks, Ubiqquitous, and Trustworthy Computing (SUTC'06)*, pp. 446–453, 2006.

[Xie06a] Z. Xie, T. Quirino, M.-L. Shyu, S.-C. Chen, and L. Chang. A distributed agent-based approach to intrusion detection using the lightweight PCC anomaly detection classier. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2006)*, pp. 446–453, Taichung, Taiwan, R.O.C., June 5-7 2006.

[Yin06] X. Yin, J. Han, J. Yang, and P.S. Yu. Efficient classification across multiple database relations: A CrossMine approach. *Transactions on Knowledge and Data Engineering*, 18(6):770–783, June 2006.