Network Intrusion Detection through Adaptive Sub-Eigenspace Modeling in Multiagent Systems

MEI-LING SHYU, THIAGO QUIRINO, and ZONGXING XIE
University of Miami
SHU-CHING CHEN
Florida International University
and
LIWU CHANG
Naval Research Laboratory

9

Recently, network security has become an extremely vital issue that beckons the development of accurate and efficient solutions capable of effectively defending our network systems and the valuable information journeying through them. In this article, a distributed multiagent intrusion detection system (IDS) architecture is proposed, which attempts to provide an accurate and lightweight solution to network intrusion detection by tackling issues associated with the design of a distributed multiagent system, such as poor system scalability and the requirements of excessive processing power and memory storage. The proposed IDS architecture consists of (i) the Host layer with lightweight host agents that perform anomaly detection in network connections to their respective hosts, and (ii) the Classification layer whose main functions are to perform misuse

This article is based on the paper "A Distributed Agent-Based Approach to Intrusion Detection Using the Lightweight PCC Anomaly Detection Classifier" by Zongxing Xie, Thiago Quirino, Mei-Ling Shyu, Shu-Ching Chen, and Liwu Chang, which appears in the Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 446–453. © 2006 IEEE.
For Mei-Ling Shyu, this research was supported in part by NSF ITR (Medium) IIS-0325260. For Shu-Ching Chen, this research was supported in part by NSF EIA-0220562 and NSF HRD-0317692.
Authors' addresses: M.-L. Shyu, T. Quirino, and Z. Xie, Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33124; S.-C. Chen, Distributed Multimedia Information System Laboratory, School of Computing and Information Sciences, Florida International University, Miami, FL 33199; email: chens@cs.fiu.edu; L. Chang, Naval Research Laboratory, Washington, DC 20375.

© 2007 ACM 1556-4665/2007/09-ART9 $5.00 DOI 10.1145/1278460.1278463 http://doi.acm.org/10.1145/1278460.1278463

detection for the host agents, detect distributed attacks, and disseminate network security status information to the whole network. The intrusion detection task is achieved through the employment of the lightweight Adaptive Sub-Eigenspace Modeling (ASEM)-based anomaly and misuse detection schemes. Promising experimental results indicate that ASEM-based schemes outperform the KNN and LOF algorithms, with high detection rates and low false alarm rates in the anomaly detection task, and outperform several well-known supervised classification methods such as C4.5 Decision Tree, SVM, NN, KNN, Logistic, and Decision Table (DT) in the misuse detection task. To assess the performance in a real-world scenario, the Relative Assumption Model, feature extraction techniques, and common network attack generation tools are employed to generate normal and anomalous traffic in a private LAN testbed. Furthermore, the scalability performance of the proposed IDS architecture is investigated through the simulation of the proposed agent communication scheme, and satisfactory linear relationships for both degradation of system response time and agent communication generated network traffic overhead are achieved.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*

General Terms: Experimentation, Security

Additional Key Words and Phrases: Agent communications, agent-based distributed system, adaptive sub-eigenspace modeling (ASEM), intrusion detection, network security

ACM Reference Format:

Shyu, M.-L., Quirino, T., Xie, Z., Chen, S.-C., and Chang, L. 2007. Network intrusion detection through adaptive sub-eigenspace modeling in multiagent systems. ACM Trans. Autonom. Adapt. Syst. 2, 3, Article 9 (September 2007), 37 pages. DOI = 10.1145/1278460.1278463 http://doi.acm.org/10.1145/1278460.1278463

1. INTRODUCTION

The increases of speed and capacity in computational and communication resources as well as the advances in computing and information technologies have enabled network systems to play an increasingly critical role in modern society. Particularly, the popularity of Web-based applications has led to the interconnection of almost all the computers in the world in a global network that facilitates communications among people. Securing such a large-scale networked system becomes a great challenge, since network intruders have found the perfect environment to develop innumerous algorithms that make effective use of the current simplistic, end-user empowering, networking model of the Internet [Clark 2001]. The fact that more sensitive data have been stored and manipulated through the Internet where numerous intrusions have brought serious damages to people, corporations, and the whole society has made network security an extremely vital issue. To address this issue, accurate and efficient intrusion detection systems (IDSs) have been developed to safeguard the network systems and crucial information.

There are two main categories for the existing intrusion detection methods that are employed in IDSs, namely misuse detection and anomaly detection [Anderson et al. 1995]. The methods in the misuse detection category are mainly based on signature modeling of known attacks [Lazarevic et al. 2003], and have the advantage of higher accuracy in detecting known attacks. However, the most obvious shortcoming of the misuse detection methods is their

incapacity to detect previously unobserved attacks. Different from the misuse detection methods, anomaly detection methods are based on signature modeling of normal traffic [Labib and Vemuri 2004]. Anomaly detection methods have the advantage of detecting new types of attacks [Hochberg et al. 1993], but their false alarm rates are high. IDSs have undergone rapid developments in both power and scope in the last few years. Various types of IDS architectures have been developed in the literature, such as monolithic, hierarchic, agent-based, and distributed (GrIDS) systems [Verwored and Hunt 2002]. Improvements made to these methods and architectures would better enhance their ability to take into account the evolving nature of network attacks.

Recently, the agent concept has emerged as a powerful paradigm in distributed computing environments due to its favorable characteristics including higher level abstractions, scalability, adaptability, graceful degradation of service and so on, over the non-agent based IDSs [Lee and Stolfo 2000]. It is a fact that many complex systems at work today in the real world are composed of components that are either geographically spread out hardware systems (traffic control systems and energy distribution systems), software subsystems spread out across many different servers (large scale database systems such as the ones maintained by government agencies, hospitals, and banks), or a combination of both (such as the Internet infrastructure itself and modern mobile phone networks). According to the distributed agent (or multiagent) design methodology, the characteristics that a system must present in order to be modeled as a distributed organization of agents are the following [Moreno 2005]:

(1) Environmental constraint: Knowledge is distributed in different locations of the system (decentralized information);
(2) System constraint: Several entities, possessing different abilities, work together to solve a complex problem;
(3) Domain constraint: The problems in the application domain can be decomposed into smaller subproblems, even if some type of interdependency exists among them.

Communication networks are large, complex, and modularized distributed systems, which present the aforementioned characteristics, and thus the distributed agent (or multiagent) design methodology can and have been successfully employed in the design of intrusion detection architectures suitable for such a distributed environment. Among some of the issues encountered in the design of a distributed agent-based IDS are (i) the amount of network traffic overhead introduced by the detection system into the residing network, (ii) the appropriateness of the expressive provisions of the agent communication language and negotiation protocols associating the various entities in the system, (iii) the hardware processing power required to execute the agents' software, and (iv) the performance of the data mining detection algorithm employed by the agents. Most of the designed agent-based intrusion detection systems require comparatively high processing power in local, end-user machines to execute the agents' software and other supportive software, and introduce considerably more traffic into their residing network [Dasgupta and Brian 2001;

Spafford and Zamboni 2000]. All of these design issues are indications that a lightweight agent-based system, with low network traffic overhead generation and low processing power requirements, is needed to overcome the issues encountered in the existing architectures.

In this article, a distributed IDS with multiagent design technology is proposed, where a set of classification agents communicate with each other and with lower level agents to acquire a global scope of the security state of the network. The proposed architecture integrates anomaly and misuse detection schemes, and focuses on the application of the lightweight agent concept in an attempt to solve the issue of executing agent software in end-user machines that have low processing power. It intends to detect heterogeneous intrusions in the network by seeking multiple information sources to extract features suitable for an effective intrusion detection process. The Adaptive Sub-Eigenspace Modeling (ASEM) based anomaly and misuse detection schemes are proposed, incorporating our proposed Weighted Multiple Correspondence Analysis (WMCA) to handle nominal features. Our experimental results demonstrate that ASEM outperforms the K-nearest neighbor (KNN) method [Liao and Vemuri 2002] and LOF algorithm [Breuning et al. 2000] with high detection rates and low false alarm rates in anomaly detection. ASEM also outperforms several well-known supervised classification methods such as C4.5 Decision Tree [Quinlan 1993], Support Vector Machine (SVM) [Han 2003], K-Nearest Neighbor (KNN) [Tou and Gonzalez 1974], Logistic [Hooper 1999], Nearest Neighbor (NN) [Tou and Gonzalez 1974], and Decision Table (DT) (all implemented in the Weka package [Weka 2007]) in the misuse detection task. Moreover, in order to validate the feasibility of the deployment of our proposed distributed agent-based IDS architecture in a real-world scenario, in terms of scalability related criteria such as generated network traffic overhead and nonlinear degradation of system response time, the implementation for the proposed architecture was developed using Matlab [Mathworks 2007] and Java software. The promising experimental results indicate both a satisfactory linear degradation of system response time and low overhead in agent communication related network traffic.

The remainder of this article is organized as follows. Section 2 discusses existing work related to the design of agent-based IDSs. Section 3 presents our proposed distributed multiagent IDS architecture. In Section 4, WMCA and ASEM-based anomaly and misuse detection schemes are introduced. Experimental setup is presented in Section 5. Experiments and results are given in Section 6. Finally, in Section 7, conclusions are discussed.

2. EXISTING WORK

Various distributed intrusion detection architectures using the multiagent design methodology and/or the data mining techniques have been developed, ranging from those comprised entirely of mobile agents [Helmer et al. 2003], simulating the human body immune system model [Foukia et al. 2001], entirely comprised of static agents [Spafford and Zamboni 2000; Ertoz et al. 2004; Snapp et al. 1991], or as a combination of both [Kannadiga and Zulkernine 2005].

One well-known example of applying distributed agent design methodology in the intrusion detection domain is the Distributed Intrusion Detection System (DIDS) [Snapp et al. 1991]. DIDS was an attempt 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. DIDS architecture presents both advantages and disadvantages. On one hand, the system utilizes the real-time traffic information from various sources, namely, data from various host monitors, to assess the security status of its residing network. However, as a drawback, the system's scalability is poor for large networks, as an increasing number of host monitors also significantly increase 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. In a recent study by [Kannadiga and Zulkernine 2005], the Distributed Intrusion Detection using Mobile Agents (DIDMA) system 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.

In contrast to DIDS, a more recent development in the domain of distributed IDS architectures is MINDS [Ertoz et al. 2004]. The MINDS system analyzes data collected directly by sensors distributed throughout the network, tapping information directly from the routers. It combines an unsupervised anomaly detection data mining algorithm, which assigns to each of the collected network connections a score reflecting how anomalous it is, and an association pattern analysis-based module, which generates a summarization report of those network connections that are ranked highly anomalous. Although MINDS seems to solve both anomaly and misuse detection problems, it requires human efforts to assist in its data mining techniques for their proper functioning. That is, the summarized anomalous data information needs to be supplied to a human analyst who is then responsible for manually performing the unsupervised anomalous data labeling process.

Another distributed agent-based IDS called Distributed Hybrid Agent Based Intrusion Detection and Real Time Response System [Vaidehi and Ramamurthy 2004] analyzes anomalies to detect and identify the Denial of Service (DoS) and data theft attacks, in addition to analyzing intrusion signatures capable of detecting wardriving-based hacks. 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, in that each agent must take on almost all of the work load of network traffic sniffing, data parsing, and intrusion detection. This makes the architecture inherently less lightweighted. In addition, its data mining techniques are less powerful since they are capable of detecting only a limited number of network attacks.

In Helmer et al. [2003], an IDS prototype entirely comprised of mobile agents was developed. In this architecture, the mobile agents travel among monitored systems in a network of distributed systems, obtain information from

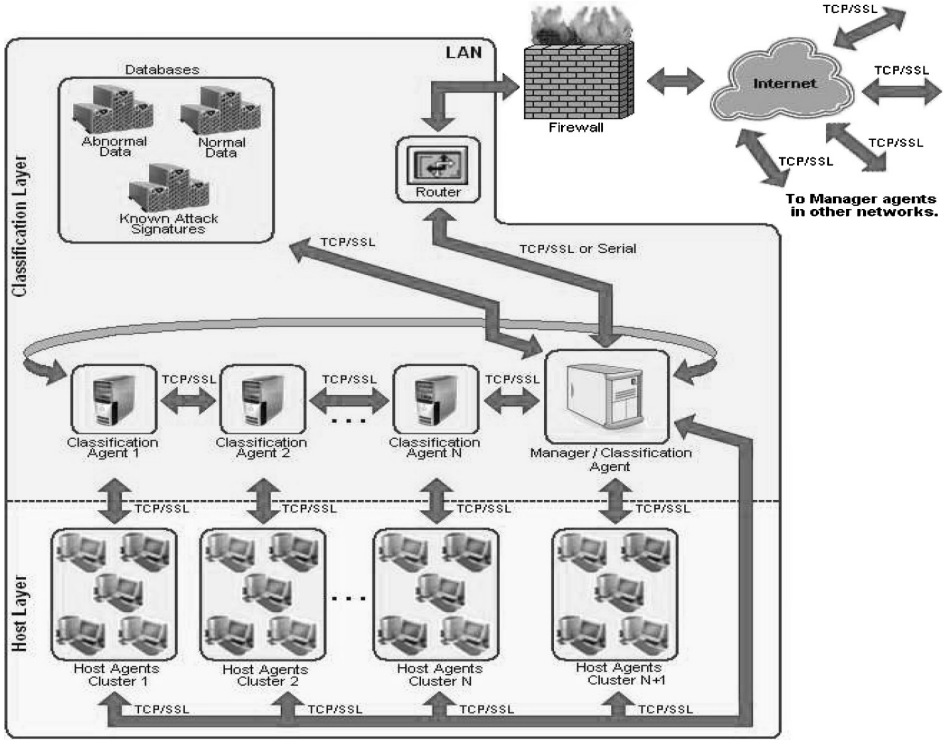

Fig. 1. The proposed IDS architecture.

designated data-cleaning agents that reside at each host, classify and correlate the supplied information, and finally report the analysis results to a designated administrator through a user interface and several databases. One of its main advantages is its support for the runtime addition of new capabilities into the mobile agents. However, one of its main disadvantages is the overhead in time required to transmit the mobile agents' code and required data among the monitored hosts in the residing network, which reduces the system's ability to respond to network intrusions in real time.

All of these architectures, having both their advantages and disadvantages, attempt to achieve the common goal of effective intrusion detection, while at the same time minimizing the adverse side effects of realistic constraints, such as the limited availability of processing power at hosts, and the scalability issues inherent in distributed system design.

3. DISTRIBUTED MULTIAGENT INTRUSION DETECTION SYSTEM

Figure 1 presents our proposed distributed multiagent IDS architecture, which consists of a host layer and a classification layer.

3.1 Host Layer

A set of lightweight host agents constitutes the host layer. Virtually, every machine in a network can be equipped with a host agent. Host agents run as

background processes in end user machines. Host agents are mainly concerned with detecting abnormal network traffic activities occurring in their hosts and properly responding to them. These agents collect information about the network connections in their hosts and classify these connections into two categories, normal or abnormal, using the ASEM-based (Adaptive Sub-Eigenspace Modeling) anomaly detection scheme. Each host agent, upon coming online, connects to an upper-layer agent called a classification agent, to whom the host agent reports abnormal connections. To reduce network traffic requirements for the proposed IDS, no communication occurs among individual host agents. Furthermore, host agents make use of the fact that every connection to their respective hosts, whether normal or abnormal, must be analyzed for abnormality using the ASEM-based anomaly detection scheme. Thus, host agents retain the information of a small percentage of normal connection instances and pass it along to the upper layers, which save this information into databases. This saved normal connection information can then be used to retrain the intrusion detection classifiers at a later time. This process introduces a desirable feature, where the burden of collecting updated training data for the classifiers is shared among every host agent connected to the proposed IDS network. The collected information is only sent to the upper layers during times of low network activity, thus preventing the IDS from overloading its residing network system during demanding periods. This delayed feedback scheme is facilitated by the fact that the intrusion detection classifiers do not have to be retrained so often as to require host agents to continuously feed back updated classifier training data to the upper layers.

3.2 Classification Layer

This layer is the classification agent layer, which is composed of classification agents. Classification agents are more specialized agents that attend to the concern of host agents on their suspicions of a possible ongoing attack. Multiple host agents connect to a single classification agent, a concept depicted in Figure 1 as the TCP/SSL connection between clustered host agents and their respective classification agents. Host agents rely on their respective classification agents to classify into known attack types, the abnormal connection instances found in their host machines through anomaly detection, a task performed by the classification agent through our proposed ASEM-based misuse detection scheme. This is an important step, as the uncovering of the attack type will determine the proper response from an IDS to an intrusion. Within the classification layer, lies a more distinctive classification agent named the manager agent, which carries a briefly higher degree of authority and responsibility within its layer. The goal of the further specialization of the manager agent is to introduce a significant degree of dynamism into the distributed system by having the manager agent serve as a central point to which all agents can register their presence within the IDS system and report, in real-time, interesting events taking place in the network.

Agents in the classification layer are assumed to be running in dedicated machines capable of delivering the processing power required to handle all the

misuse detection classification requests of their respective host agents, and communication with their fellow manager agent. These agents observe a strict sequence of communication steps upon uncovering an attack type from an abnormal connection instance received from their respective host agents. This process was devised to provide a more effective, yet less costly manner of achieving global awareness of an ongoing threat:

(1) Upon uncovering of an attack, a classification agent will first quickly warn the host agent who is the source of the abnormal connection instance with information about the attack type. This is due to the fact that this host agent is a primary victim of the uncovered attack.
(2) Next, the classification agent will warn its fellow manager agent of the ongoing attack situation taking place in one of its nodes, by providing information regarding the attack type, source IP, and source port of the intruder.
(3) Then the classification agent will initiate the process of warning all of its remaining connected host agents about the ongoing attack by providing them with the same information, so that they can prevent the attack from harming their respective hosts.
(4) The manager agent, upon receipt of a warning message from a fellow classification agent, will in turn ask all the other fellow classification agents to broadcast to all of their respective host agents the same information it received about the ongoing threat.
(5) Finally, the remaining classification agents, upon receiving a broadcast request from the manager agent, will warn all of their respective host agents of the ongoing network threat so that the host agents can place barriers to prevent, or at least lessen, the effects of the ongoing threat.

This communication among agents in the classification layer is another important feature of the proposed distributed multiagent IDS, as other classification agents can prevent or lessen the effects of a possible attack, by managing resources in their nodes that they expect to be affected by the incoming attack, such as bandwidth, communication ports, and connection authorizations. Finally, classification agents collect the normal connection information from host agents, in addition to those abnormal connection instances classified by the ASEM-based misuse detection scheme, and pass them along to the manager agent, who then saves it in the corresponding databases for the purpose of future classifier retraining.

As was previously mentioned, the manager agent is a classification agent, which is slightly more specialized within its class and performs extra managing duties within the classification layer. It is the only agent in the proposed distributed multiagent IDS architecture that is required to reside in a machine with a static network address. One of the purposes of the employment of the manager agent is to enhance the dynamism of the proposed IDS architecture by introducing into the system desirable features such as:

(1) It allows host and classification agents to dynamically join the IDS network at any time and register their services and presence at a common point of access. This feature is depicted in Figure 1 at the classification layer as the

TCP/SSL connection link between the classification agents and the manager agent, and at the host layer as the TCP/SSL connection link between the host agent clusters and the manager agent. Classification agents can register their dynamic network addresses with the manager agent, thus preventing the need for hard-coding their addresses into the agents' software or requiring human supervision.

(2) It prevents the issue of single point of failure. For instance, upon coming online and joining the IDS network, classification agents are required to register their services with the manager agent. If a classification agent requires maintenance or has a software issue that requires it to go offline for a period of time, host agents connected to that specific classification agent can be rerouted to other available classification agents by requesting from the manager agent the network address of others that are available.

(3) It allows the misuse detection classification work load at the classification layer to be fairly divided among the classification agents, thus allowing the distributed system to make full use of the available resources. As was previously mentioned, classification agents attend to the concern of their host agents by classifying abnormal connection data instances into specific attack types. With the introduction of the manager agent, host agents coming online and joining the IDS network are required to request from the manager agent the network address of an available classification agent. Furthermore, classification agents are required to inform the manager agent of the number of host agents they are serving, as more and more host agents connect to them, in addition to providing information about their incoming communication message rates. Through this feedback scheme, the manager agent is able to control and fairly divide the workload among the classification agents by appropriately routing host agents.

(4) It provides the capability of detecting and blocking distributed network attacks such as DoS and network scanning. The manager agent, upon receiving warning messages from various classification agents, can deduce from the received information about the attack, including the attack types, source and destination ports, intruders' IP addresses, and so on, whether a distributed attack is taking place in hosts throughout the network. The manager can then utilize common protocols, such as SNMP and CISCO, or direct cable connections, to make routers and switches block attacks at specific points in the network.

(5) The task of retraining the ASEM-based anomaly and misuse detection schemes and updating the required databases is assigned to the manager agent. The manager agent has access to databases where information about normal and abnormal connection instances, and known attacks' signature information are kept. This is depicted in Figure 1 as the link between the manager agent and the set of databases. The manager agent uses updated normal connection data found in the database for normal data and the abnormal connection data classified into different attack types by the misuse detection scheme—found in the database for abnormal data—to retrain the classifiers. The retraining process yields attack signatures that are placed

in the known intrusion signature database. Upon retraining the classifiers, the manager agent sends updated classifier parameter information down to the lower layer.

(6) Finally, the manager agent can stay in contact with manager agents residing in other networks, while exchanging relevant information such as new intrusion signatures. This feature makes use of the proactive behavior of agents and continually strengthens the immunity of the system, while allowing it to maintain a global view of its network environment.

3.3 Communication between Two Layers

At the very core of the efficient performance of the coordination aspect of a distributed multiagent system is an appropriate agent communication language (or ACL). An ACL must be capable of providing clear and declarative (what?) rather than mostly descriptive (how?) meaningful semantics [Singh 1999], in addition to an ontology with broad coverage in its domain, possibly extensible to other domains so as to allow for the interoperability of various classes of agents. The ACL must also provide a negotiation protocol that reflects a large domain of possible interaction among different agents (goals and operations) [Kone et al. 2000]. Due to all these requirements, the decision on the choice of which communication language to use in the design of a distributed multiagent system is not to be taken lightly. The KQML language is one of the most commonly used languages due to its versatility and generality of purposes. KQML supports multiagent communication through an extensible set of reserved primitives called performatives, which represent communicative acts [Kone et al. 2000]. Having this knowledge in mind, the KQML language was adopted as the default ACL for our proposed architecture, and a simple and manageable communication scheme that utilizes a discrete number of KQML performatives was designed to accommodate the goals of all the agents. To provide total privacy and authentication capabilities, TCP/IP Secure Socket Layer (SSL) is adopted for the implementation of secure communication among agents. 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. Every possible inter/intra-layer communication event that can arise was taken into consideration in the design of our proposed communication scheme. We propose a standard communication scheme that utilizes the following KQML performatives at the different architectural layers:

(1) RECOMMEND-ONE: Upon coming online, host agents connect to the manager agent. Since they are aware of the manager agent's unique network address, they request from it the address of an available classification agent using the RECOMMEND-ONE performative. A unique reply-id is placed within the *:reply-with* field of the message.

(2) REGISTER: Upon receiving from the manager agent the address of an available classification agent, host agents connect to the respective classification agent and utilize the REGISTER performative to join that classification

agent's cluster. Classification agents also utilize the REGISTER performative upon coming online to inform the manager agent of their availability in the IDS network.

(3) UNREGISTER: Employed by both the host and classification agents to disconnect from the IDS network.

(4) EVALUATE: Employed by the host agents to request misuse detection services from their respective classification agents, and consequently and indirectly, inform them of possible attacks. Host agents place the feature values of the abnormal connections within the *:content* field and a unique reply-id for the message within the *:reply-with* field of the KQML message.

(5) TELL: Employed by classification agents for the purposes of both replying to host agents' classification request and informing the manager agent, and consequently and indirectly all other classification agents, of an ongoing threat. Classification agents place information about an attack, including the extracted connection feature values, classification attack type, the intruder's IP address and source port and so on, within the *:content* field of the KQML message. The same message sent as a classification reply to a host agent can also be sent to the manager agent as the semantics of the message involve describing an ongoing threat in both scenarios. The only difference is that reply messages sent to host agents contain a unique reply-id within the *:reply-to* field of the message, while warning messages sent to the manager agent do not contain the *:reply-to* field at all. The TELL performative is also employed by the manager agent to reply to RECOMMEND-ONE performative messages sent by host agents requesting the address of available classification agents, upon coming online.

(6) BROADCAST: The BROADCAST performative is used to disseminate information about an ongoing threat to the whole IDS network. It is employed by the manager agent to inform all other classification agents of an ongoing threat, while also requesting them to forward that information to all of their respective connected host agents.

(7) FORWARD: Employed by host agents, in periods of low network traffic conditions, to forward to the manager agent through their respective classification agents the normal connection instances that are temporarily stored in files within a host agent's residing machine. The *:content* field of the message carries an embedded KQML database INSERT performative message that is employed to request the manager agent to save the connection instance into the database.

(8) INSERT: As mentioned above, this KQML database performative is employed by host agents to send normal connection instances to the manager agent.

Additionally, an agent's name within the IDS network, which is placed within the *:sender* and *:receiver* fields of all KQML messages, is defined in a manner that carries significant semantic information and uniquely identifies every agent within the IDS network. An agent's name is defined as a string formed by the concatenation of the keywords "HA" for host agents, "CL" for classification

agents, or "MA" for the manager agent, followed by the "@" character, the agent's host network IP address, the ":" character, and the agent's desired local communication port. For instance, an agent with the name "HA@10.0.1.4:3345" is a host agent embedded in a machine with IP address 10.0.1.4, and utilizing port 3345 for all of its network communication.

Please note that EVALUATE, TELL, and BROADCAST performatives are the three most commonly employed in this communication scheme and are the core of the information and knowledge exchange process required for effective intrusion detection. All the other performatives supplement the communication scheme in spite of being employed less often: when agents join or leave the IDS network and when data is sent to the manager agent during periods of low network traffic. One desirable feature that results from this communication scheme is that the employed KQML messages are equivalent to relatively small amounts of data exchanges among the agents. This contributes to making the proposed communication scheme very lightweight, as will be demonstrated in Section 6.

Finally, regarding the choice of languages used to encode the information within the *:content* keyword of the KQML messages, many standard choices of languages, such as Prolog and KIF, exist and have been found suitable for our communication purposes. The Knowledge Interchange Format (KIF) language is chosen, however, due to its expressive power and extensive online reference resources.

4. INTRUSION DETECTION SCHEMES

In response to the challenge of automated analysis of increasing network data and intrusion detection, the ASEM (Adaptive Sub-Eigenspace Modeling) scheme was developed to provide a lightweight solution to both the anomaly and misuse detection tasks, which can be incorporated into our proposed distributed multiagent IDS architecture to facilitate an effective and efficient data mining process while taking into account the lightweight requirements of our proposed architecture.

4.1 Nominal Feature Handling

Features extracted from network connections can be either numerical or have some form of categorical (or nominal) significance [KDD 1999]. Examples of numerical features are the number of bytes transferred between two hosts, the duration of a connection, the number of successful or failed login attempts, the source and destination ports of a connection, among numerous another features. Examples of nominal features that can be extracted from network connections are the source and destination IP addresses of two communicating hosts, whether a connection successfully connected to a host (successful two-way TCP handshake), the username associated with a specific TCP connection to a host, the protocol of the connection, the service (HTTP, SSH, TELNET, FTP, etc.), whether the connection has reached root shell (full administrator privileges), among numerous others. Notice that some of these nominal features contain many categories (the service feature), while some are binary in nature

(whether a host successfully connected or not can either be "*Yes*" or "*No*"). Both numerical and nominal features contain valuable information that can make a significant difference in the effective discrimination between completely normal or malicious network connections. These motives beckon the exploration of the possibility of enhancing the capabilities of our proposed ASEM-based anomaly and misuse detection schemes (which inherently are capable of dealing only with numeric features) to also handle nominal variables. Two basic distinctive approaches that are commonly employed in the task of nominal to numerical feature conversion are:

(1) Using indicator variables in place of a nominal variable: This approach converts nominal values of each nominal feature into binary valued, zero-one dummy variables that each can be treated as if they were numerical attributes. The presence or absence of a specific nominal value in a data instance is signified by either 0 or 1 in its respective dummy variable.
(2) Employing the Multiple Correspondence Analysis (MCA) approach to compute numerical scaling values to each of the values of a categorical variable.

Both of these solutions have their advantages and disadvantages. For instance, indicator variables are easy to generate from a data set. However, their utilization is analogous to assigning the same weight to each nominal value in a data set (either 0 or 1), which may not be desirable or fully correct in certain scenarios. For instance, consider the nominal feature we discussed, which described whether or not a host successfully connected to another host. Its nominal value "*YES*" combined with the information provided by other features may indicate that an intruder has connected to a machine or is performing some sort of port scanning attack on a host, and thus may be more significant in the intrusion detection domain than its nominal value "*NO*." This is exactly what the MCA approach [Greenacre and Blasius 2006; Greenacre 1984] attempts to consider. In summary, MCA attempts to generate numerical scaling values for the nominal feature values in a data set that maximizes the overall correlation among the features. By analyzing a data set's nominal feature space, MCA captures the degree of significance and similarities among each of the nominal feature values and generates respective numerical scaling values which, in contrast to the indicator variable approach, do not necessarily have to be within the same numerical range. Moreover, numerical features can be incorporated into the MCA process as long as they are discretized into categories and then treated as nominal features. Based on MCA, we proposed the Weighted Multiple Correspondence Analysis (WMCA) algorithm to consider more effectively all information from its statistical analysis on a data set. The WMCA algorithm is described as follows.

Let the $Q \times N$-dimensional matrix $\mathbf{X} = \{\mathbf{x}_{ij}, i = 1, 2, \ldots, Q \text{ and } j = 1, 2, \ldots, N\}$, be a data set comprised of N Q-dimensional column vectors $\mathbf{X}_j = (\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{Qj})'$, $j = 1, 2, \ldots, N$, which represent the N data instances in the data set. Each of the Q rows of matrix $\mathbf{X}$ correspond to a nominal attribute or feature. Let us denote the number of categories for the qth nominal attribute by T_q, that is, the total number of distinct values that appear in the qth row of matrix $\mathbf{X}$.

Then, the total number of nominal attribute values in the entire data set is given by

$$T = \sum_{q=1}^{Q} T_q. \tag{1}$$

Next, an indicator matrix $\mathbf{Z}$ is generated to represent the data matrix $\mathbf{X}$ numerically, where $\mathbf{Z} = \{\mathbf{z}_{ij},\ i = 1, 2, \ldots, T\ and\ j = 1, 2, \ldots, N\}$. The N T-dimensional column vectors of $\mathbf{Z}$, given by $\mathbf{Z}_j = (\mathbf{z}_{1j}, \mathbf{z}_{2j}, \ldots, \mathbf{z}_{Tj})'$, $j = 1, 2, \ldots, N$, represent the N data instances in the data set. The tth row of Z corresponds to the zero-one dummy variable associated with one of the T nominal values of the data set. Furthermore, matrix $\mathbf{Z}$ can be considered as the concatenation of Q indicator matrices corresponding to the Q individual nominal attributes of the data set, that is, $\mathbf{Z} = [\mathbf{Z}_1^\top \mathbf{Z}_2^\top \ldots \mathbf{Z}_Q^\top]^\top$. Next, the $1 \times T$-dimensional row vector $\mathbf{R}$ is computed as the horizontal concatenation of the Q row mass vectors of the indicator matrix $\mathbf{Z}$ as $\mathbf{R} = [\mathbf{R}_1\ \mathbf{R}_2\ \ldots \mathbf{R}_Q]$, where each row vector $\mathbf{R}_j \in \mathbf{R}$, for $j = 1, 2, \ldots, Q$, is given by

$$\mathbf{R}_j = \left(\frac{1}{N}\right) \mathbf{1}^\top \mathbf{Z}_j^\top. \tag{2}$$

Now, the $T \times T$-dimensional diagonal matrix $\mathbf{D}$, corresponding to the row mass matrix of $\mathbf{Z}$, is computed from the diagonalization of the row vector $\mathbf{R}$ as

$$\mathbf{D} = \left(\frac{1}{Q}\right) \times diag(\mathbf{R}),\ \text{where} \tag{3}$$

$diag(\mathbf{R})$ is the matrix whose diagonal elements correspond to the elements of row vector $\mathbf{R}$. Next, the $T \times T$-dimensional matrix $\mathbf{C}$, also known as the *Burt* matrix, is computed as the inner-product of the indicator matrix $\mathbf{Z}$ using $\mathbf{C} = \mathbf{Z}\mathbf{Z}^\top$. Now, the $T \times T$-dimensional square matrix $\mathbf{M}$ can be defined as

$$\mathbf{M} = \mathbf{D}^{-\frac{1}{2}} \left(\frac{\mathbf{C}}{Q^2 N} - \mathbf{D}\mathbf{1}\mathbf{1}^\top\mathbf{D}\right) \mathbf{D}^{-\frac{1}{2}},\ \text{where} \tag{4}$$

—$\mathbf{D}^{-\frac{1}{2}}$ is the inverse of the matrix composed of the square root of the elements in matrix $\mathbf{D}$;

—is a $1 \times T$ column vector of 1's.

Singular Value Decomposition (SVD) is now applied to matrix $\mathbf{M}$ in order to extract its T eigenvalue-eigenvector pairs, namely, $(\lambda_1, \mathbf{E}_1)$, $(\lambda_2, \mathbf{E}_2)$, $\ldots$, $(\lambda_T, \mathbf{E}_T)$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_T \geq 0$ and $\mathbf{E} = [\mathbf{E}_1\ \mathbf{E}_2 \ldots \mathbf{E}_T]$ is the matrix composed of the T column eigenvectors $\mathbf{E}_j = (\mathbf{e}_{1j}, \mathbf{e}_{2j}, \ldots, \mathbf{e}_{Tj})'$, $j = 1, 2, \ldots, T$. It is at this point where our proposed WMCA approach detaches, at least in a statistical sense, from the traditional procedure followed in the execution of the original MCA approach. In essence, traditional MCA utilizes the T elements of the first eigenvector $\mathbf{E}_1$, namely, the first major principal component of matrix $\mathbf{M}$, to compute T scaling values for the T nominal attribute values found in the data set. As a result of the utilization of a single eigenvector, all the information provided by the remaining eigenvectors and eigenvalues is simply discarded. In order to

more effectively utilize the valuable scaling value information provided by all of the eigenvalues and eigenvectors of matrix $\mathbf{M}$, we propose a procedure that automatically selects significant eigenvectors and compute a weighted average of these eigenvectors, while taking into account the significance of each eigenvector based on the magnitude of their eigenvalues. This automated procedure is described as follows.

First, the set of eigenvectors is refined by eliminating those possessing extremely insignificant or null eigenvalues: carrying very little information. Thus, all those eigenvectors not satisfying the refinement equation (5) are discarded.

$$\lambda_\upsilon > \phi, \text{ where} \tag{5}$$

—ϕ is an adjustable coefficient whose value is set by default to $0.01 \times \lambda_1$, based on our empirical studies;

—$\lambda_\upsilon > \phi$ is the eigenvalue of the (υ)th eigenvector $\mathbf{E}_\upsilon$ satisfying Equation (5);

—$\upsilon \in \mathbf{V}$ is defined as the refined eigenvector space.

Next, a function based on the standard deviation values of the refined eigenvector space, and which captures the similarity degree among the nominal feature values in the data set, is defined. Thus, all those eigenvectors satisfying Equation (6) are retained while all others are discarded.

$$STD(\mathbf{E}_\psi) \leq \text{Mean}_{\text{STD}}(\mathbf{E}_\upsilon). \tag{6}$$

—$Mean_{STD}(\mathbf{E}_\upsilon)$ is the average value of all the standard deviation values of the eigenvectors in the refined eigenvector space $\mathbf{V}$;

—$STD(\mathbf{E}_\psi)$ is the standard deviation of the eigenvector satisfying Equation (6) and corresponding to the (ψ)th eigenvector $\mathbf{E}_\psi$;

—$\psi \in \mathbf{W}$ is defined as the final refined eigenvector space containing all eigenvectors satisfying both Equations (5) and (6).

Now, those eigenvectors satisfying both Equations (5) and (6) are combined through a weighted average into a single $T \times 1$-dimensional column vector $\mathbf{H} = \sum_{\psi \epsilon W} \frac{\lambda_\psi}{\lambda_t} \mathbf{E}_\psi$.

—λ_ψ is the eigenvalue corresponding to the (ψ)th eigenvector $\mathbf{E}_\psi$ in the final refined eigenvector space $\psi \in \mathbf{W}$;

—$\lambda_t = \sum_{\psi \in \mathbf{W}} \lambda_\psi$ is the total sum of the eigenvalues corresponding to those eigenvectors present in the final refined eigenvector space $\psi \in \mathbf{W}$.

The T scaling values for the T nominal attribute values found in the data set, and which are ordered by the order of appearance of their corresponding dummy variables in the indicator matrix $\mathbf{Z}$, are given by the $T \times 1$-dimensional column vector $\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}\mathbf{H}$. Here, $\mathbf{S}$ can be further divided into Q column vectors corresponding to the Q nominal attributes of the data set, and which are represented by the Q indicator matrices composing matrix $\mathbf{Z}$, that is, $\mathbf{S} = [\mathbf{S}_1^\mathsf{T}\ \mathbf{S}_2^\mathsf{T}\ \ldots\ \mathbf{S}_Q^\mathsf{T}]^\mathsf{T}$. Finally, the numeric data set matrix $\mathbf{X}'$, corresponding to the numeric version

of the nominal data set matrix $\mathbf{X}$, is computed as:

$$\mathbf{X}' = \left[\mathbf{Z}_1^{\mathsf{T}}\mathbf{S}_1 \;\; \mathbf{Z}_2^{\mathsf{T}}\mathbf{S}_2 \;\; \ldots \;\; \mathbf{Z}_Q^{\mathsf{T}}\mathbf{S}_Q\right]^{\mathsf{T}}. \tag{7}$$

In summary, through a modified numerical scaling value generation process, our proposed WMCA approach attempts to make effective use of the similarity information found in all the eigenvectors and eigenvalues attained from a data set, rather than limiting our statistical analysis to a single major principal component.

4.2 Adaptive Sub-Eigenspace Modeling (ASEM)

Periodically, normal data connections collected from the host agents are stored in the database of normal data maintained in the manager agent. Using the normal data, the proposed Adaptive Sub-Eigenspace Modeling (ASEM) algorithm is employed in the ASEM-based anomaly detection and the misuse detection schemes, which are executed in the host agents and classification agents, respectively.

Let $p \times N$-dimensional matrix $\mathbf{X} = \{\mathbf{x}_{ij},\ i = 1, 2, \ldots, p \ and \ j = 1, 2, \ldots, N\}$, represent the training normal data set, where the N p-dimensional column vectors $\mathbf{X}_j = (\mathbf{x}_{1j}, \mathbf{x}_{2j}, \ldots, \mathbf{x}_{pj})'$, $j = 1, 2, \ldots, N$, represent the N training normal data instances. Let $\mathbf{Z} = \{\mathbf{z}_{ij},\ i = 1, 2, \ldots, p \ and \ j = 1, 2, \ldots, N\}$ represent the normalized training normal data with the corresponding column vectors representing normalized training normal data instances $\mathbf{Z}_j = (\mathbf{z}_{1j}, \mathbf{z}_{2j}, \ldots, \mathbf{z}_{pj})'$, and let $\bar{\mu}_i$ and s_{ii} be the mean and variance of the ith row of $\mathbf{X}$. Equation (8) is used to normalize the training normal data instances.

$$\mathbf{z}_{ij} = \frac{\mathbf{x}_{ij} - \bar{\mu}_i}{\sqrt{s_{ii}}}. \tag{8}$$

Let $(\lambda_1, \mathbf{E}_1), (\lambda_2, \mathbf{E}_2), \ldots, (\lambda_p, \mathbf{E}_p)$ be the p eigenvalue-eigenvector pairs of the correlation matrix $\mathbf{S} = \frac{1}{N-1}\sum_{j=1}^{N}(\mathbf{Z}_j - \bar{\mathbf{Z}})(\mathbf{Z}_j - \bar{\mathbf{Z}})'$ for the normalized training normal data set $\mathbf{Z}$, $\bar{\mathbf{Z}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{Z}_j$, and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$. The p eigenvalue-eigenvector pairs make up the p-dimensional eigenspace, which is the platform for both the anomaly detection scheme and the misuse detection scheme.

Next, the transformation or projection of the normalized training normal data set can be obtained from the original space to the eigenspace. Let the matrix $\mathbf{Y} = \{\mathbf{y}_{ij},\ i = 1, 2, \ldots, p \ and \ j = 1, 2, \ldots, N\}$ be the projection or score matrix of $\mathbf{Z}$ onto the p-dimensional eigenspace, where the score column vectors $\mathbf{Y}_j = (\mathbf{y}_{1j}, \mathbf{y}_{2j}, \ldots, \mathbf{y}_{pj})'$, $j = 1, 2, \ldots, N$, correspond to the projection of the normalized training normal data instances. Let $\mathbf{E}_i = (e_{i1}, e_{i2}, \ldots, e_{ip})'$ be the i^{th} eigenvector for the normalized training normal data set. Equation (9) defines the ith score value of the jth normalized vector $\mathbf{Z}_j$.

$$\mathbf{y}_{ij} = \mathbf{E}_i{}'\mathbf{Z}_j = \ e_{i1}\mathbf{z}_{1j} + e_{i2}\mathbf{z}_{2j} + \cdots + e_{ip}\mathbf{z}_{pj}. \tag{9}$$

After defining the score row vectors $\mathbf{R}_i = (\mathbf{y}_{i1}, \mathbf{y}_{i2}, \ldots, \mathbf{y}_{iN})$, $i = 1, 2, \ldots, p$, an adaptive sub-eigenspace selection function is proposed in Equation (10) to model the essential characteristics of the training normal data set.

$$a < STD(\mathbf{R}_m) < b, \tag{10}$$

where

—a is the preset coefficient to exclude extremely small principal components, with the default value 0.0001;

—b is an adjustable coefficient, with the default value $MEAN_{STD}(\mathbf{R}_\upsilon)$ (the average value of all $STD(\mathbf{R}_\upsilon)$), and $\mathbf{R}_\upsilon$ satisfies the left-hand side of Equation (10);

—$STD(\mathbf{R}_m)$ is the standard deviation of the selected score row vector satisfying Equation (10);

—$m \in \mathbf{M}$, $\mathbf{M}$ is defined as the adaptive sub-eigenspace.

The selection function in Equation (10) uses the extracted inherent statistical information from the training data to model the training data set. Standard deviation, a widely used statistical measure for differentiating the degree of variance or similarity [Pentland et al. 1994], is used based on our empirical study with comparison to other statistical measures like mean, slope, and so on. In addition, the default values of both a and b are also determined based on empirical studies.

Here, the original training data set (normal data) is reconstructed and modeled in the adaptive subeigenspace with dimensionality reduction. The selected principal component contributing to the sub-eigenspace represent the most meaningful information of the training data set with respect to the hidden similarity and are possibly nonconsecutive, as demonstrated by our experiments.

4.3 ASEM-based Anomaly Detection Scheme

For anomaly detection, a suitable discriminate measure to differentiate normal and abnormal data instances is required. For this purpose, a distance threshold function for each training normal data instance is defined by using Equation (11).

$$\mathbf{c}_j = \sum_{m \in M} \frac{(\mathbf{y}_{mj})^2}{\lambda_m}, \tag{11}$$

where

—$m \in \mathbf{M}$ is the index of the mth principal component selected from Equation (10);

—λ_m is the eigenvalue of the corresponding mth principal component;

—$\mathbf{y}_{mj}$ is the score value of the mth feature in the adaptive subeigenspace;

—$\mathbf{c}_j$ is the threshold value for each data instance in the training data set; all of them contribute to the array $\mathbf{C} = \{\mathbf{c}_j,\ j = 1, 2, \ldots, N\}$.

Accordingly, let $\mathbf{SC}$ be the sorted $\mathbf{C}$ in an ascending order, $h = [(1-\alpha)\times N]$ be the nearest integer to $(1-\alpha)\times N$, and α be the preset false alarm rate of the classifier. A class-deviation threshold $\mathrm{C}_\theta = \mathbf{SC}[h]$ can be defined to differentiate between the normal and abnormal data instances.

Let $\mathbf{X}' = \{\mathbf{x}'_{ij},\ i = 1, 2, \ldots, p \ and \ j = 1, 2, \ldots, N'\}$ be a $p \times N'$-dimensional matrix containing N' p-dimensional column vectors $\mathbf{X}'_j = (\mathbf{x}'_{1j}, \mathbf{x}'_{2j}, \ldots, \mathbf{x}'_{pj})'$,

$j = 1, 2, \ldots, N'$, representing N' testing instances, including both normal and abnormal data instances. The testing data instances are then normalized with $\mathbf{z}'_{ij} = \frac{\mathbf{x}'_{ij} - \bar{\mu}_i}{\sqrt{s_{ii}}}$, where $\mathbf{Z}' = \{\mathbf{z}'_{ij}\}$ represents the normalized testing data instances with corresponding column vectors $\mathbf{Z}'_j = (\mathbf{z}'_{1j}, \mathbf{z}'_{2j}, \ldots, \mathbf{z}'_{pj})'$, and $\bar{u}_i$ and s_{ii} are the same parameters used in Equation (8).

Let the testing data score matrix $\mathbf{Y}' = \{\mathbf{y}'_{ij},\ i = 1, 2, \ldots, p \text{ and } j = 1, 2, \ldots, N'\}$ be the projection of $\mathbf{Z}'$ onto the p-dimensional adaptive sub-eigenspace of the training data set $\mathbf{X}$, where the score column vectors $\mathbf{Y}'_j = (\mathbf{y}'_{1j}, \mathbf{y}'_{2j}, \ldots, \mathbf{y}'_{pj})'$, $j = 1, 2, \ldots, N'$, correspond to the projection of each of the N' testing data instances. Then the ith score value of the jth normalized testing instance vector $\mathbf{Z}'_j$ is given by:

$$\mathbf{y}'_{ij} = \mathbf{E}_i{}' \mathbf{Z}'_j = e_{i1}\mathbf{z}'_{1j} + e_{i2}\mathbf{z}'_{2j} + \cdots + e_{ip}\mathbf{z}'_{pj}, \tag{12}$$

where $\mathbf{E}_i = (e_{i1}, e_{i2}, \ldots, e_{ip})'$ is the same eigenvector used in Equation (9).

Compute the threshold vector $\mathbf{C}' = \{\mathbf{c}'_j,\ j = 1, 2, \ldots, N'\}$ using the class-deviation function in Equation (13).

$$\mathbf{c}'_j = \sum_{m \in M} \frac{(\mathbf{y}'_{mj})^2}{\lambda_m}, \tag{13}$$

where

—λ_m and $\mathbf{M}$ are the same parameters used in Equation (10);

—$\mathbf{y}'_{mj}$ is the the score value of the mth eigenspace feature of the projected and normalized original testing data instance $\mathbf{X}'_j$.

Thus, a testing observation $\mathbf{X}'_j$ is classified as abnormal to the class of the training normal data when $\mathbf{c}'_j > \mathrm{C}_\theta$, or as normal when $\mathbf{c}'_j \leq \mathrm{C}_\theta$.

4.4 ASEM-Based Misuse Detection Scheme

For misuse detection, it is required to define a suitable discriminate measure to differentiate each attack. Let $\mathbf{X}^k = \{\mathbf{x}_{ij},\ i = 1, 2, \ldots, p \text{ and } j = 1, 2, \ldots, N^k\}$ represent each attack in the training data set. The corresponding projection or score matrix onto the obtained subeigenspace from the training normal data set (described in Section (4.2)) can be obtained using Equation (9), and the corresponding array $\mathbf{C}^k = \{\mathbf{c}_j,\ j = 1, 2, \ldots, N^k\}$ can be obtained using Equation (11). Then, Equation (14) defines an attack reference threshold value $\mathrm{C}^{\mathrm{k}}_\theta$ to differentiate each attack. For each testing observation $\mathbf{X}'_j$, the distance with respect to each attack $\mathbf{D}^k_j$ is obtained with Equation (15). Accordingly, the attack label for $\mathbf{X}'_j$ is determined by the smallest $\mathbf{D}^k_j$ value.

$$\mathrm{C}^{\mathrm{k}}_\theta = \frac{1}{N^k} \sum_{j=1}^{N^k} \mathbf{c}_j. \tag{14}$$

$$\mathbf{D}^k_j = |\mathbf{c}'_j - \mathrm{C}^{\mathrm{k}}_\theta|. \tag{15}$$

The ASEM-based misuse detection scheme is inspired by the fact that each network attack has different characteristics and thus results in a self-cluster for the corresponding score values projected onto the adaptive subeigenspace, obtained from the training normal data set in different subsections, with an obvious gap. In other words, compared with the training normal network data, each network attack possesses distinct characteristics of anomaly, which is represented as a separate subsection of the score values. Therefore, the gaps among different attacks can be captured for misuse detection.

5. EXPERIMENTAL SETUP

In order to test our framework, various experiments are organized to assess the performance of (i) the scalability characteristics of the proposed architecture in terms of the required agent communication, and (ii) the accuracy and time performance of the proposed ASEM-based intrusion detection schemes, which are the cores of agents' reasoning module. Testing data sets were acquired from three resources: the KDD CUP 1999 Data [KDD 1999], network traffic data generated in our testbed through the application of our proposed Relative Assumption Modeling algorithm, and network attacks traffic data also generated in our testbed through the application of specialized attack generation tools such as NMAP and Nessus vulnerability scanners, the THC-Hydra and Brutus multiprotocol brute-force scanners, the IRS port scanner, and finally the D-ITG network traffic generator [D-ITG 2006; InsecureOrg 2006; Oxid 2006]. Additionally, all the generated network traffic data is preprocessed by our proposed feature extraction (FE) technique in order to acquire the necessary features for our proposed ASEM-based schemes. Experimental results include not only the agent communication and system scalability performance of the proposed architecture, but also the intrusion detection accuracy rates in comparison to several well-known data mining approaches. A 10-fold cross-validation process was performed for every intrusion detection comparison experiment for a fair and better evaluation, and the standard deviation of the classification accuracy for each approach is also included. A smaller standard deviation value indicates that the classification approach performs in a consistently stable manner, whereas a larger standard deviation value indicates inconsistent or unstable performance. The term cross-validation indicates the iterative process by which a given data set is randomly split randomly into training and testing sets composed of $\frac{2}{3}$ and $\frac{1}{3}$ of all data instances, respectively.

5.1 Private Network Testbed Setup

Figure 2 illustrates our private LAN network testbed setup, where the generation of realistic network traffic and the performance assessment experiments of our proposed architecture take place.

The focus of our testbed-based experiments is on network attacks based on the TCP network protocol, since a great majority of attacks are either executed via, or rely to a certain degree, on the TCP protocol. This is due mostly to TCP's frangibility and instability. For example, a survey has shown that 90%

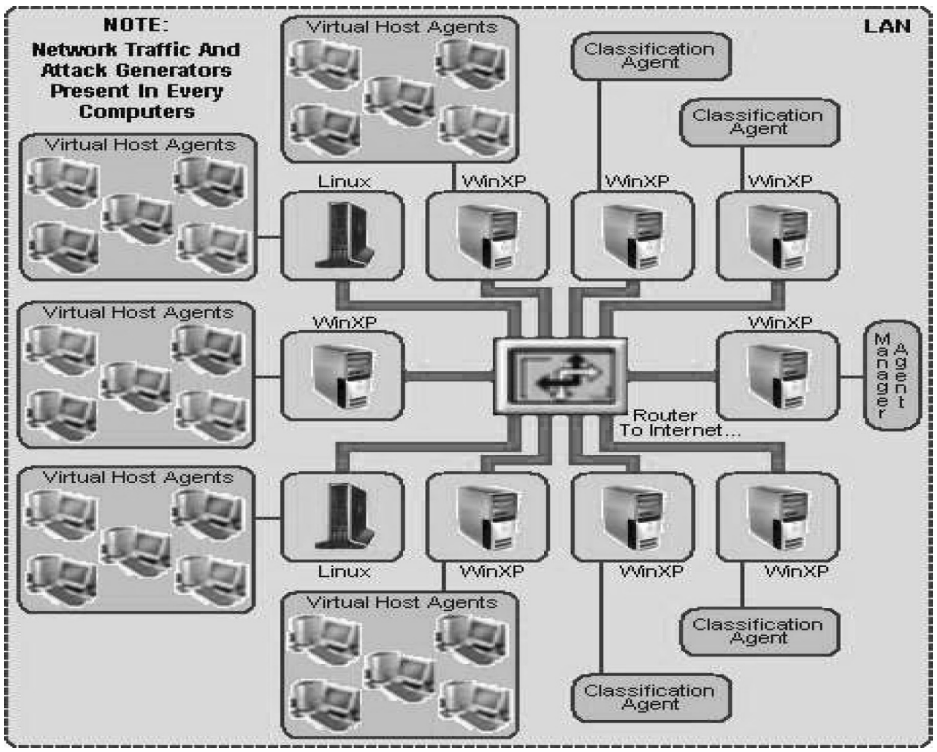

Fig. 2. The private LAN testbed setup employed to generate realistic network traffic and test the proposed distributed IDS system.

to 94% of Denial of Service (DoS) attacks, which are among the major threats to the whole Internet community, are employed via the TCP protocol [Moore et al. 2001]. Note, however, that our 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 (TCP, UDP, ICMP, and ARP) and is seamlessly incorporated into our intrusion detection schemes through the employment of WMCA.

5.2 Relative Assumption Modeling

The Relative Assumption Modeling approach is proposed due to the fact that the so-called normal and abnormal data sets are relative concepts determined by factors such as the available bandwidth of a network, server processing capability, average network load, network services provided, among innumerous others. The same type of network connections may be labeled distinctly among different kinds of networks. Furthermore, it is highly possible that the current increasing abnormal data sets would be considered as normal in the near future due to the rapid development of networks, computers, and their relative techniques.

In an attempt to produce both the normal and relatively complete abnormal data sets, we utilize a relative reversing method on core phases to propose a

pair of abstract definitions to express the following two opposite concepts:

—*typical normal connections:* generate fewer data transfers during a moderate time period in suitable rate, frequency, and pace.

—*typical abnormal connections:* generate increasing data transfers during a very short or very long time period continually and quickly.

Next, five pairs of opposite core phases are combined, such as ("suitable rate," "increasing"), ("not much," "much"), ("during a moderate time period," "during a very short or very long time period"), ("suitable frequent," "continually"), and ("suitable pace," "fleetly"), to simulate the differences between typical normal and abnormal connections in a real-world network environment. In general, abnormal connections should possess at least one or more typical features that are in the form of core phases opposite to the normal ones. That is, for one group of typical normal connections, there would be 31 ($C_5^1 + C_5^2 + C_5^3 + C_5^4 + C_5^5$) groups of corresponding abnormal connections. Additionally, different groups with different degrees of anomaly or different distribution parameters can be naturally considered as different types of attacks. Finally, these abstract phases with different numeric values in different networks should yield good relative and adjustable generated network traffic patterns.

5.3 Feature Extraction (FE)

The proposed feature extraction (FE) technique is executed in order to transform the raw generated data into data applicable for data mining and classification. Furthermore, for any supervised classification method, feature extraction (FE) is an important module, as it can drastically affect an algorithm's performance [Liu et al. 2003; Liu and Motoda 1998]. The generated network data contains all the required analysis information, which can be extracted from the data packets flowing across the network. Before the features are extracted from the network data, the network traffic needs to be collected and stored. The following three steps are proposed to extract features that are critical for effective network intrusion detection:

—Windump: the windows version of Tcpdump [Jacobson et al. 2007], which uses the Libcap [2007] library to extract low-level traffic from the network, is used to collect and store all the raw data directly from the network card.

—Tcptrace [2007]: a tool used to produce several different types of outputs containing information such as elapsed time, number of bytes and segments transferred both ways, number of retransmissions, round trip times, window advertisements, network throughput, and many more features, through the analysis of the Windump generated files, is used to extract basic information on each TCP connection from the Windump collected data.

—Our own FE techniques are used to extract/generate basic, time-based, connection-based, and ratio-based network features from the Tcptrace output file.

(1) *Basic Features*: basic information related to the connection. Seventeen basic features are extracted, which include duration, IP, port, total packets, acknowledge packets, throughput, and so on.

Table I. Relation between Core Phases and Features

Core Word Pair	Feature Type
("suitable rate," "increasing rate")	Basic and Ratio-based
("not much," "much")	Basic
("during a moderate time period," "during a very short or very long time period")	Basic
("suitable frequent," "continually")	Basic and Connection-based
("suitable pace," "fleetly")	Basic and Time-based

(2) *Time-Based Features*: the number of connections having the same IP or/and port in a 3-second sliding window. Four time-based features are extracted to provide information on both the source and destination sides of a connection.
(3) *Connection-Based Features*: the number of connections having the same IP or/and port in the last 100 connections. Four connection-based features are also extracted to provide information on both the source and destination sides of a connection.
(4) *Ratio-Based Features*: the ratio of transferred packets between two connections that have the same IP and port. Sixteen ratio-based features are extracted to provide information on both the source and destination sides of a connection. Ratios are found between the current and neighbor connections (neighbor ratio) and between the current and the first connection having the same IP and port.

In fact, all of these features correspond to one or more of the core phases previously described. For example, the first feature "duration" directly represents the core phase ("during a moderate time period," "during a very short or very long time period"). Yet another example is the ratio features, which reflect the core phase ("suitable rate," "increasing rate"). Detailed information about these relationships is shown in Table I.

5.4 Network Attack Generation Tools

In this article, a total of six software tools, ranging from vulnerability and network scanner to traffic generators, were selected in order to generate the network attack data needed for our studies.

(1) Nessus: It is perhaps the best free network vulnerability scanner today. It provides plugins for the generation of over 11,000 types of network related vulnerability tests including DoS, port scanning, and OS-specific vulnerabilities.
(2) NMAP: Similarly to Nessus, this tool has been commonly employed for the assessment of vulnerabilities present in network system setups and also as a tool for the exploration of services available across hosts of a network. It effectively tests IDS and firewall policies by generating various types of network and port scanning attacks including SYN, FIN, ACK, Window, X-MAS Tree, among others.

(3) THC-Hydra: Developed by *The Hacker's Choice Group* (THC), THC-Hydra is a brute force password-guessing tool that supports over 30 protocols and services, including cryptographic-based services such as SSH2. THC-Hydra has the ability to break into not only hosts, but also routers and firewall, making this tool a real threat for network administrators.
(4) Brutus: It is another brute force network cracker, which offers the same functionalities as THC-Hydra with an even easier installation process.
(5) IRS Scanner: It is a powerful tool that scans for IP-based restrictions rules for a particular network service. It combines ARP Poisoning and Half-Scan techniques to generate totally spoofed TCP connections. IRS is the perfect software for complementing tools such as the ones we have described.
(6) D-ITG: It provides a platform capable of producing various types of traffic at the packet level and for the network, transport, and application layers, while accurately implementing appropriate stochastic processes for traffic flow features such as the packet interdeparture time and packet size. It plays an important role in the field of network traffic analysis and can be employed to test IDS, switches, ACLs, and firewalls, as well as to provide the traffic needed by analytical network applications.

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Agent Performance Evaluation

In order to assess the scalability performance of our proposed distributed multi-agent IDS architecture in a realistic scenario, a prototype of the proposed architecture was implemented using Matlab [Mathworks 2007] and Java software. We conducted evaluations in terms of scalability-related criteria such as generated network traffic overhead and the degree of linearity of the degradation of system response time for the employment of our proposed agent communication scheme. Figure 2 shows our network testbed, where the different types of agents were placed in mutually exclusive hosts 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 any host. This means that host and classification agents were not executed together within the same machine, while the manager agent was placed in a host all by itself and without the presence of any other classification agents. This measure ensures that every conversation between the agents would generate measurable network traffic and yield realistic scalability results. With these constraints in mind, the agents were placed in the hosts having the following IP addresses within the testbed:

—Host agents: host agents were placed in the hosts having local IP addresses 10.0.1.1, 10.0.1.2, 10.0.1.3, and 10.0.1.5.
—Classification agents: classification agents' hosts occupied the local addresses 10.0.1.4, 10.0.1.7, and 10.0.1.8.
—Manager agent: the manager agent occupied by itself the host having the local IP address 10.0.1.6.

Each classification agent, including the manager agent was executed within its own Matlab session. An equal number of multiple Matlab sessions was executed in the hosts occupied by the host agents in order to support the large number of host agents being simulated during the experiments. Each Matlab session was allowed to support a maximum of 50 host agents before a new session was initiated to support the execution of more host agents. Next, 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 distributed architecture. In summary, our experiments consisted of instantiating an ever increasing number of classification and host agents to simulate an ever increasing IDS network, in a manner that would reflect the performance of the proposed architecture as it was expanded from a small to a large scale. Throughout the experiments, the number of classification agents connected to the IDS ranged from a minimum of 1 (the manager agent running alone) to a maximum of 6 (the manager agent plus 5 more classification agents sharing an equal classification load). For each of these different numbers of connected classification agents, the total number of host agents varies from a minimum of 10 host agents to a maximum of 100 host agents per classification agent. There are a total of 60 (6×10) experiments, 1 to 6 classification agents, each with 10, $20, \ldots,$ 100 host agents. Therefore, the maximum number of agents is 600, to simulate a realistic network of 600 hosts connecting to the IDS network. For each experiment, the following simulation takes place:

(1) The desired number of, and types of, agents are instantiated. The IDS network lies quietly waiting for an abnormal event.
(2) A host agent is chosen and is ordered to emulate the detection of an abnormal connection to its residing host.
(3) The host agent requests the classification agent to which it is connected, to perform supervised classification of the abnormal connection instance and to reply with the attack type. This is achieved using the KQML EVALUATE performative.
(4) Upon receiving the abnormal instance from the host agent, the classification agent performs the misuse detection task, and then sends the classification results back to the host agent. This is achieved using the KQML TELL performative.
(5) The classification agent then informs the manager agent about the intrusion recently detected by one of its host agents, by also sending the classification results to the manager agent. This is achieved using the KQML TELL performative.
(6) The classification agent then starts the process of informing, one by one, all of its remaining connected host agents of the ongoing threat. This is also achieved through the use of the KQML TELL performative.
(7) The manager agent, upon receiving the ongoing threat warning from the classification agent, informs the other remaining classification agents, one

by one, of the ongoing threat and also requests them to forward the warning to all their respective host agents. This is achieved through the KQML BROADCAST performative.

(8) As a last step, the classification agents, upon receiving the warning message or broadcast request from the manager agent, forward the received warning to their respective host agents, one by one.

(9) Now, the IDS network lies once again quiet, all agents having been informed of the ongoing threat, waiting for the next abnormal event.

(10) At any time, the size of the IDS network may increase—more classification or host agents are added—and the process is repeated from Step 2.

Furthermore, for each of these 60 scenarios, two features are measured, starting from the time when step 1 is executed to the time when step 9 ends, as an indication of the scalability performance. These features are the average bandwidth (in Mbps) consumed by the communication among the agents throughout the simulation, and the system response time—the time period between the detection of an abnormal instance by a host agent (Step 1) and the last host agent being informed of the ongoing threat (Step 9). The system response time can also be interpreted as the system's awareness time, that is, the time required for all the agents in the IDS network to acquire the most updated information about the security status of their residing network. Both the required average bandwidth and system response time features are measured using the Ethereal [2007] packet capture and analysis software. Since the classification agent is the midpoint between every host/classification/manager agent's conversations, Ethereal is executed in all of the hosts where the classification agents reside.

The experiment makes a few assumptions, such as the fact that the effectiveness of the agent communication scheme will be assessed in the absence of network traffic or network attacks, and anomaly and misuse detection are ignored at the host and classification layers respectively. Additionally, as illustrated by Step 1 of the simulation scenario, it is assumed at the beginning of the simulation that all the required number and types of agents are already connected to the network. Thus, some provisions of the proposed communication scheme, such as the agent registration part and host $\rightarrow$ classification $\rightarrow$ manager agents normal data forwarding scheme, which utilize the KQML RECOMMEND-ONE, REGISTER, UNREGISTER, FORWARD, and INSERT performatives, are not required for the successful implementation of the experiment and are thus left out of the experiment for convenience. This is due to the fact that the focus of the experiment is primarily on the evaluation of those KQML performatives that are involved with the intrusion detection and information dissemination processes. Furthermore, by varying both the numbers of classification and host agents that have to be informed of a possible attack from low to significantly high values, the experimental results should illustrate the pattern in the increase of the system response time and required network bandwidth resources as more and more agents are introduced to reveal the practicality of the employment of our proposed communication scheme, and consequently, the scalability performance of our proposed IDS.

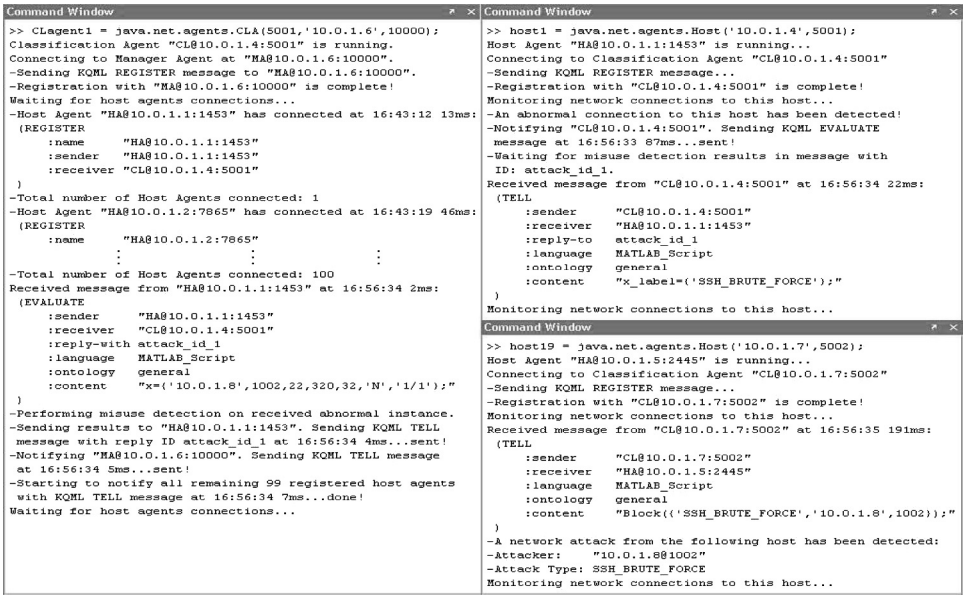

Fig. 3. KQML messages employed by the classification and host agents for communication.

6.1.1 *Performance of the Communication Scheme.* The execution of the 60 different simulation scenarios yielded the generation of numerous KQML messages, which follow the communication scheme described in Section 3.3. Figures 3 and 4, for instance, illustrate a sample of the messages generated during the simulation of the last simulation scenario, namely when 6 classification agents and 100 host agents per classification agent were connected to the IDS network.

Figure 3 is divided into three parts. The left side shows the execution of the classification agent software, while the top-right part shows the execution of the host agent responsible for initiating the simulation process by emulating the detection of an abnormal connection (simulation Step (2)), and the bottom-right shows the execution of the host agent that is the last agent in that particular simulation to have received the BROADCAST warning from the manager agent. Figure 4, on the other hand, corresponds entirely to the execution output of the manager agent. In particular, the left-hand side of Figure 4 corresponds to the manager agent's execution output during the simulation process. The interpretation of the right-hand side will be elaborated in Section 6.1.4. Notice from the top-right side of Figure 3 that the host agent initiates the simulation process at 16:56:33 87 *ms* by sending a KQML EVALUATE message to the classification agent. The execution of the proposed communication scheme can be followed from the figures. The simulation ends at 16:56:35 191 *ms*, when the last host agent receives a warning message forwarded by its classification agent from the manager agent, as shown in the bottom-right of Figure 3. The total required system awareness time for this simulation is then found to be (16:56:35 191 *ms* - 16:56:33 87 *ms* = 00:00:02 104 *ms*). This result is more clearly illustrated in Figure 5, the plot of the required system awareness time vs. the

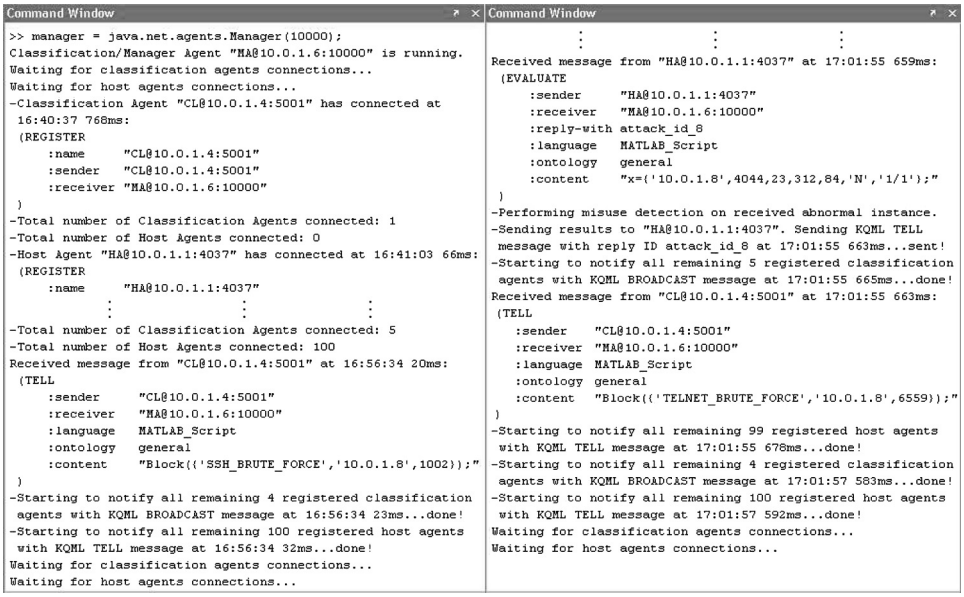

Fig. 4. KQML messages employed by the manager agent for communication.

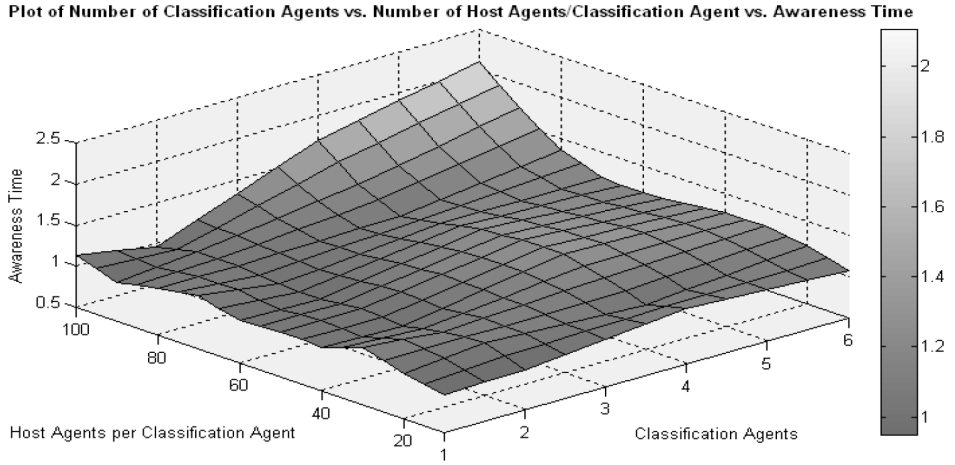

Fig. 5. Plot of the communication time required by the system for full intrusion awareness, as a function of the number of agents in the IDS.

number of classification agents and vs. the number of host agents per classification agent, where it corresponds to the peak awareness time required by the architecture, for all the simulation scenarios.

6.1.2 *System Awareness Time Scalability.* Figure 5 depicts an interesting and rather promising result. Note that the ideal case scenario for the awareness time scalability performance of the system, when network related delays are ignored, is linear and dependent only on the number of hosts per classification

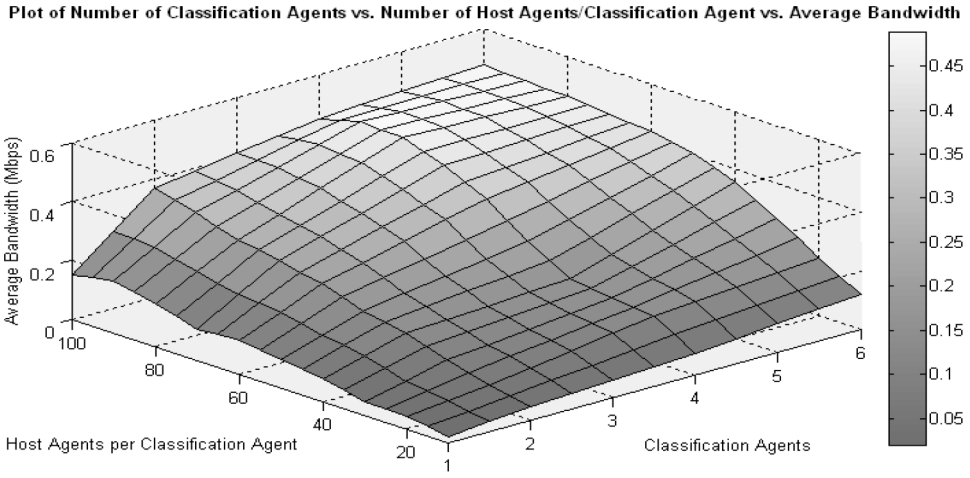

Fig. 6. Plot of the average bandwidth (in Mbps) required by the IDS system during the communication process, as a function of the number of agents in the IDS.

agent. For instance, we can assume that the time taken for the manager agent to warn the classification agents is negligible compared to the time taken by the classification agents to warn the host agents, since the number of host agents in the IDS network is much larger. At the peak of the communication process, when all classification agents are broadcasting in parallel exactly the same warning TELL message to their respective host agents, the time taken by each classification agent should be ideally equal since they work individually and in parallel. Thus, the system awareness time is independent of the number of classification agents and linearly dependent solely on the number of host agents per classification agent under the assumptions.

Figure 5 resembles for the most part a plane, as expected, with a very low slope. This indicates that the increase in the awareness time of the IDS is linear with respect to the number of agents connected to the system, while the low slope indicates a good stability of the system to a significant increase in the number of agents connected to it. Also, it is not until the number of hosts per classification agent reaches between 80 and 100, and the number of classification agents varies between 3 and 6 (between 240 to 600 total agents) that the network-related delays become noticeable and the awareness time starts to increase with a higher linear slope. Mature programming techniques can lead to significant improvement of the communication scheme performance, better than a Matlab and Java based simulation, thus decreasing the slope of the line shown in Figure 5, and consequently, the degrading effect that scalability has on the time response performance of the system. Finally, the presented linearity of the awareness time scalability is a promising result, indicating the feasibility of a realistic employment of the proposed architecture.

6.1.3 *Required Average Bandwidth Scalability.* Figure 6 also depicts an interesting and rather promising result. The ideal case scenario for the average bandwidth scalability performance of the system is also linear. However,

in contrast to the awareness time scalability case, it is dependent only on the number of classification agents in the system. Making the same assumptions as before, if a snapshot is taken at any point in time during the peak of the communication process, when all classification agents are broadcasting in parallel exactly the same warning TELL message to their respective host agents, the average total number of messages traveling from the classification layer to the host layer is ideally equal to the number of classification agents, since each classification agent sends a maximum of 1 message at a time. Thus, the system average bandwidth scalability is independent of the number of host agents per classification agent, and linearly dependent solely on the number of classification agents under the assumptions.

It can be observed that Figure 6 also resembles a plane, as expected, having a relatively linear slope. This linear slope indicates that (i) the increase in the average bandwidth of the IDS system is linear with respect to the number of agents connected to the system, and (ii) it presents relatively good stability due to the absence of sharp changes in the average bandwidth as the number of agents in the system increases. Approximate linear scalability, whether time or bandwidth related, is always a desirable feature in any distributed system. Systems with a nonlinear response to scalability may collapse a network upon the execution of communication intensive jobs or fail to function as well as desired for the users of the system.

6.1.4 *Distributed Attack Detection Capabilities.* As was previously mentioned, let us return once again to Figure 4, which shows the execution output of the manager agent. One important feature of the proposed architecture is illustrated in the right-hand side of the figure. Recall that the manager agent is also a classification agent, and thus it is also responsible for its own cluster of host agents. The figure illustrates the scenario in which the manager agent receives a misuse detection classification request from one of its host agents, namely, "HA:10.0.1.1:4037" at 17:01:55 659 *ms* and initiates the process of classifying the abnormal instance, replying to the host agent, warning all the other 5 connected classification agents, and finally, warning all the remaining 99 host agents in its own cluster. Notice, however, that in the middle of this process, the manager agent receives a warning message from one of its classification agents "CL@10.0.1.4:5001" at 17:01:55 663 *ms*. After handling the host agent triggered event sequence, the manager agent then handles the new incoming message by notifying the other remaining 4 classification agents of the new attack, and all of the 100 host agents in its own cluster. Notice also the content of the message received from agent "HA:10.0.1.1:4037." The first, second, and third features describe the source IP (10.0.1.8), source port (4044), and destination port (23) of the detected abnormal connection. Upon classifying this feature vector, the manager agent will have knowledge that a TELNET based brute force attack is being generated from host 10.0.1.8 at source port 4044. At the same time, the message from "CL@10.0.1.4:5001" describes that a TELNET based brute force attack is also being generated from a malicious host 10.0.1.8 from source port 6559. At this point, the manager agent is capable of intelligently entailing that a distributed TELNET based brute force attack is being generated from

host 10.0.1.8, and it can further entail that, since the source ports are different for both attacks, the intruder may be generating them in parallel and from different local socket addresses.

In essence, these results show that the proposed IDS architecture provides a natural way of detecting distributed network attacks, which may include different types of attacks, such as DoS and network scanning attacks, and entailing possible characteristics of the attack such as the parties involved in it and its frequency. This is a desirable feature of the proposed IDS, as distributed forms of attacks commonly occur in our networks today.

6.2 Anomaly Detection Performance Evaluation

Various experiments comparing the K-nearest neighbor (KNN) [Liao and Vemuri 2002] and LOF [Breuning et al. 2000] methods, are conducted with the following three pairs of data sets with preset false alarm rates from 0.01% to 2%. It is noted that the KNN method employed here is a threshold-based algorithm designed for anomaly detection [Liao and Vemuri 2002], which is not the same as the one commonly used in the regular classification domain, since here only normal, rather than both normal and abnormal, data instances are used in the anomaly detection process. The average anomaly detection rates of KNN with the pre-set parameter $k = 1, 2, 3, 4$, and 5 are used for better evaluation. In LOF [Breuning et al. 2000], each instance is assigned an *outlierness* degree called the local outlier factor (LOF) of an instance, which depends on how isolated the instance is with respect to the surrounding neighboring instances.

—Pair 1: The 12,932 normal instances and 11,618 abnormal instances used in Xie et al. [2006]. It is a mix of generated normal and abnormal instances based on our testbed and DoS connections from the MIT Tcpdump data set (LLDOS2.0.2.) [DARPA 2007]. The data has a total of 39 numeric features. As in a typical anomaly detection scenario, from among the normal data instances, 3,000 are randomly chosen for training.

—Pair 2: The 2,233 normal and 3,005 abnormal instances generated in our testbed through our proposed Relative Assumption Modeling approach and Feature Extraction techniques. The data has a total of 47 features, where 4 of them are nominal. A total of 2/3 of the normal data instances are randomly chosen for training.

—Pair 3: The 5,000 normal instances and 4,444 abnormal instances obtained from KDD CUP 1999 Data [KDD 1999]. The data has a total of 41 features, where 3 of them are considered as nominal. Among them, 2/3 of the normal data instances are randomly chosen for training.

For the Pair 1 data set, where the abnormal instances are mixed with several kinds of attacks, the true detection rates of the ASEM-based scheme, LOF, and KNN are presented in Table II. Additionally, all the observed false alarm rates for all of the methods are approximately equal to the preset values, and thus we move our focus to the anomaly detection rates for evaluating the performance of ASEM. From the table, it can be seen clearly that ASEM maintains a high detection rate ($> 94\%$) under a low range of preset false alarm rates (from

Table II. Anomaly Detection Rates Comparison among ASEM, LOF, and KNN with the Pair 1 Data Set. Standard Deviations of Classification Accuracy are Shown in Parentheses

False Alarm	ASEM	LOF	KNN
0.01%	94.27%(±2.13)	49.83%(±10.53)	47.41%(±15.90)
0.05%	94.34%(±2.53)	51.04%(±9.88)	49.33%(±15.47)
0.10%	94.56%(±2.67)	67.22%(±9.72)	65.10%(±16.08)
0.50%	95.98%(±2.85)	71.68%(±13.33)	70.09%(±13.55)
1%	97.23%(±1.46)	77.26%(±17.97)	76.95%(±14.75)
2%	98.05%(±1.32)	80.32%(±19.21)	77.33%(±14.95)

Table III. Anomaly Detection Rates Comparison among ASEM, LOF, and KNN with the Pair 2 Data Set. Standard Deviations of Classification Accuracy are Shown in Parentheses

False Alarm	ASEM	LOF	KNN
0.01%	88.88%(±2.04)	0.00%(±0.00)	0.00%(±0.00)
0.05%	95.34%(±1.96)	0.07%(±0.04)	0.00%(±0.00)
0.10%	95.44%(±1.88)	1.09%(±0.58)	0.00%(±0.00)
0.50%	100%(±0.00)	99.92%(±0.06)	99.90%(±0.08)
1%	100%(±0.00)	100%(±0.00)	15.67%(±4.36)
2%	100%(±0.00)	100%(±0.00)	0.00%(±0.00)

0.01% to 2%), outperforming LOF and KNN, especially in the false alarm range of 0.01% to 0.05%, where ASEM always possesses a higher detection rate (40% higher) than those of LOF and KNN. Furthermore, the ability of the ASEM-based scheme to maintain high detection rates with increasing false alarm rates indicates that the stability of the ASEM-based scheme is superior to that of LOF and KNN. This is achieved through the selection of the representative components, in contrast to assigning the same weight to all the attributes, as is done in LOF and KNN, which yields a more robust predictive model. These results reveal that ASEM possesses the favorable feature of maintaining a preponderant and stable anomaly detection rate.

Table III lists the detection rates for the Pair 2 data set. From this table, it can be clearly seen that ASEM maintains a high detection rate ($\geq$88.88%) and outperforms LOF and KNN. In general, ASEM seems to be more stable than both LOF and KNN since the anomaly detection accuracy of the latter two methods varies greatly from 0% to even 100%, under different false alarm rates, or the use of different training data sets. KNN performs the worst among the three methods, which indicates that when few training instances or even few unsuitable instances are present in the training data set, it severely impacts KNN's classification accuracy. LOF also exhibits failure points when the false alarm rate is set to less than or equal to 0.1%. This indicates its inefficiency in relatively low false alarm rates.

For the Pair 3 data set, Table IV exhibits the detection rates of ASEM, LOF, and KNN methods. ASEM's ability to maintain almost 100% accuracy again demonstrates its robustness and stability. ASEM outperforms LOF and KNN in all false alarm rates in the experiments. Similarly, KNN performs the worst, as its anomaly detection rate is always below 31%. This is due to the fact that

Table IV. Anomaly Detection Rates Comparison among ASEM, LOF, and KNN with the Pair 3 Data Set. Standard Deviations of Classification Accuracy are Shown in Parentheses

False Alarm	ASEM	LOF	KNN
0.01%	99.96%(±0.04)	0.00%(±0.00)	0.00%(±0.00)
0.05%	99.98%(±0.02)	70.59%(±9.03)	30.78%(±10.53)
0.10%	100%(±0.00)	73.58%(±8.84)	25.87%(±9.86)
0.50%	100%(±0.00)	74.39%(±8.32)	25.67%(±11.77)
1%	100%(±0.00)	79.79%(±6.58)	19.89%(±8.21)
2%	100%(±0.00)	80.27%(±5.79)	19.87%(±9.02)

the difference in Euclidean distance between the two attacks in space is so small that KNN cannot distinguish well between the different classes.

In summary, it is noted that in all three pairs of data sets, the observed false alarm rates of the ASEM-based scheme are approximately equal to the preset values, indicating that ASEM maintains the misclassification rate for normal data instances under control.

Another important performance measure acquired from all of these experiments is the different methods' training and classification speeds. ASEM performs much faster than LOF and KNN, especially in comparison to the LOF method. For example, for the Pair 1 data set, ASEM only requires about $\frac{1}{20}$ of the time associated with training and classification by KNN, and about $\frac{1}{60}$ of that of LOF, in the same execution environment of the empirical studies. It is well known that training and classification speeds are crucial factors that must be considered in the implementation of systems capable of practical real-time responses. ASEM's promising speed results make it a good choice for use in anomaly detection applications. Therefore, along with its inherent lightweight feature from the representative components based approach, where the extracted representative principal component and related information are used to replace all the information provided by hundreds of data instances, ASEM also shows good operational benefits.

6.3 Misuse Detection Performance Evaluation

Three groups of data sets are used to evaluate ASEM's misuse detection performance.

—Group 1: Seven types of network attacks, namely 884 Back, 908 Satan, 1,215 Smurf, 138 Apache, 929 Neptune, 108 Portsweep, and 262 Warezmaster instances from the KDD CUP 1999 Data [KDD 1999]. The data has a total of 41 features, where 3 of them are considered as nominal features.

—Group 2: Three types of network attacks, namely $attack1$, $attack2$, and $attack3$, generated in our LAN network testbed [Xie et al. 2006] through the proposed Relative Assumption Modeling and Feature Extraction techniques. The data has a total of 47 features, where 4 of them are nominal. The data in this group include (i) 881 $attack1$ abnormal network connection instances, out of which 500 are randomly selected for training and the remaining ones are used for testing, (ii) 1,047 $attack2$ abnormal network

Table V. Misuse Detection Rates Comparison among ASEM, C4.5, SVM, KNN, Logistic, NN, and DT. Standard Deviations of Classification Accuracy are Shown in Parentheses

Accuracy %	Group 1	Group 2	Group 3
ASEM	99.85%(±0.08)	97.75%(±1.02)	98.41%(±0.36)
C4.5	99.79%(±0.17)	97.80%(±1.24)	97.74%(±1.18)
SVM	98.94%(±0.24)	93.94%(±2.48)	96.44%(±2.86)
KNN	97.76%(±1.75)	96.47%(±2.07)	96.62%(±2.96)
Logistic	98.03%(±0.93)	96.86%(±2.04)	96.79%(±2.44)
NN	97.77%(±1.31)	96.89%(±2.54)	96.02%(±2.81)
DT	99.65%(±0.18)	95.30%(±2.75)	96.65%(±2.29)

connection instances, where 500 are randomly selected for training and the remaining ones are used for testing, and (iii) 1,077 $attack3$ network connection instances, out of which 500 are randomly selected for training and the remaining ones are used for testing.

—Group 3: Data generated by network attack generation tools. It is composed of 6 different attacks including 2,295 ACK, 4,661 Connection, 1,576 HTTP, 4,373 SSH, 2,607 TELNET, and 603 Window. The data has a total of 47 features, where 4 of them are nominal. From them, $\frac{2}{3}$ of each attack is randomly selected for training.

In order to ensure the full manifestation of the two aspects of high anomaly detection and low observed false alarm rates in the experiments, each group of data is comprised of data instances belonging to multiple classes and the α value is set to 0.1% (a typical low false alarm rate being used in many research areas [Branden and Hubert 2004, 2005]).

Table V shows the classification accuracy of the ASEM, C4.5 decision trees, SVM, K-Nearest Neighbor (KNN) (k = 5), Logistic, Nearest Neighbor (NN), and Decision Table (DT) for each group of data sets. As can be seen from this table, the misuse detection rates for all methods are comparable, as the known attacks always possess obviously different characteristics. In fact, ASEM outperforms all other compared methods (except for the C4.5 method in Group 2), and more importantly, it has the smallest standard deviation values for all three groups of data, indicating it is more stable than the other methods in the experiments.

Another important comparison is the response time. For many systems that employ misuse detection, real-time response is a critical issue. Many existing data mining algorithms present algorithmic responses that, without suitable modifications, cannot be employed in real-time demanding applications such as intrusion detection. Our experiments have, however, further indicated that ASEM presents the potential to be employed in a real-time intrusion detection system.

In the first place, ASEM requires significantly lower training and classification times than all of the other compared methods, as is illustrated in Table VI, which shows the average combined time in seconds for the training and classification times required by the investigated methods under the same execution environment. The combined, rather than the individual, training and classification times are used because, for instance, for NN and KNN, the training model generation time is negligible, as it involves simple operations

Table VI. Comparison of the Average Combined Time (in Seconds) for Training and Classification Among ASEM, C4.5, SVM, KNN, Logistic, NN, and DT

Combined Times	Group 1	Group 2	Group 3
ASEM	12.1 *s*	7.6 *s*	14.3 *s*
C4.5	17.3 *s*	14.4 *s*	19.6 *s*
SVM	278.6 *s*	112.3 *s*	312.7 *s*
KNN	83.5 *s*	48.3 *s*	98.4 *s*
Logistic	115.6 *s*	78.9 *s*	136.5 *s*
NN	75.4 *s*	37.5 *s*	90.7 *s*
DT	53.6 *s*	47.4 *s*	59.2 *s*

such as data normalization, thus the overall combined time reflects solely the classification time of the methods. However, for the methods other than NN and KNN, the classification time was found to be a significantly small portion of the overall combined time, in other words, most of the methods perform the classification task in a very efficient manner. The drawback though is in the requirement of a larger training time. Thus, the results presented in Table VI primarily reflect the training time costs associated with the methods other than NN and KNN. From Table VI, it can be observed that ASEM uses a significantly lower training time than all other methods, especially for the Group 3 data set, which has the largest amount of data instances and attributes.

Additionally, ASEM requires less memory storage to maintain the classification components obtained during the training phase. In contrast, methods such as NN and KNN, which are instance-based, require large memory storage to maintain a predictive model that can, at times, be composed of hundreds of training data instances. Also, rule-based methods such as C4.5 and DT may have large memory storage requirements for the hundreds of rules generated from a data set, thus, leading to inefficient classifiers. For example, as illustrated in Table VI, KNN's and NN's time response performances present huge deteriorations for large data sets.

7. CONCLUSION

In this article, a novel distributed multiagent IDS architecture is presented, which incorporates the desirable features of the multiagent design methodology with highly accurate, fast, and lightweight ASEM-based anomaly and intrusion detection schemes. Even as a larger number of agents are introduced into the network, our proposed architecture provides effective communication between its two comprising layers, through an efficient agent communication scheme that requires only a small and manageable generation of network traffic overhead, as shown in the experiment results. A key concept in the design of the proposed IDS architecture was the concept of lightweight agents, which takes into account realistic assumptions regarding the host machines with possibly low processing power and memory in the network. To test the feasibility of a realistic employment of all the salient aspects of the proposed IDS architecture, a private LAN testbed is built to facilitate both the generation of realistic normal and anomalous network traffic, and common network attack generation

tools, in order to appropriately verify the performance of the ASEM-based intrusion detection schemes when compared to other well-known anomaly detection and supervised classification methods, and to assess the important scalability aspects of the proposed architecture such as system response time and agent communication generated network traffic overhead. From the experimental results, we can conclude that (i) the ASEM-based anomaly detection scheme achieves a satisfactory combination of both high detection rate ($> 88\%$) and good operational merits such as low training and classification times, (ii) the ASEM-based misuse detection scheme achieves $> 97\%$ accuracy for all three groups of data, and (iii) the ASEM-based misuse detection scheme also has the lowest combined training and classification times. Possessing high detection rates in both the anomaly and misuse detection tasks, and equipped with good operational merits, ASEM was found to be suitable for use in both layers of the proposed IDS architecture. Finally, the experimental results on the scalability of the proposed architecture yielded promising results indicating a satisfactory linear scalability of both the system response time and average bandwidth requirements as a larger number of agents are introduced into the IDS network. These results are indicative that our proposed IDS architecture provides many favorable characteristics such as being lightweight, and having good scalability, adaptability, and graceful degradation of service.

REFERENCES

ANDERSON, D., FRIVOLD, T., AND VALDES, A. 1995. Next-generation intrusion detection expert system (NIDES): A summary. In *SRI International Technical Report*. Vol. 95. Menlo Park, CA. 28–42.

BRANDEN, K. AND HUBERT, M. 2004. Robust classification in high dimensional data. In *Proceedings in Computational Statistics*. Prague, Czech Republic, 1925–1932.

BRANDEN, K. AND HUBERT, M. 2005. Robust classification in high dimensions based on the SIMCA method. *Chemometrics and Intelligent Laboratory Systems* 79, 10–21.

BREUNING, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. 2000. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD Conference*. Dallas, TX. 93–104.

CLARK, D. 2001. Rethinking the design of the internet: end to end arguments vs. the brave new world. *ACM Trans. Inter. Tech. 1*, 1 (Sept.), 70–109.

DARPA 2007. Intrusion detection evaluation data sets. available at http://www.ll.mit.edu/.

D-ITG. 2006. Distributed internet traffic generator. available at http://www.grid.unina.it/software/ITG/.

DASGUPTA, D. AND BRIAN, H. 2001. Mobile security agents for network traffic analysis. In *DARPA* Information Survivability Conference and Exposition (DISCEX II'01). Vol. 2. Anaheim, CA. 332–340.

ERTOZ, L., EILERTSON, E., LAZAREVIC, A., TAN, P., SRIVASTAVA, J., KUMAR, V., AND DOKAS, P. 2004. *The MINDS—Minnesota Intrusion Detection System, Next Generation Data Mining*. MIT Press, Cambridge, MA.

ETHEREAL. 2007. Ethereal—A network protocol analyzer. available at http://www.ethereal.com.

FOUKIA, N., HULAAS, J., AND HARMS, J. 2001. Intrusion detection with mobile agents. In *Proceedings of the 11th Annual Internet Society Conference*. Stockholm, Sweeden.

GREENACRE, M. AND BLASIUS, J. 2006. *Multiple Correspondence Analysis and Related Methods*. Chapman and Hall, Boca Raton, FL, USA.

GREENACRE, M. J. 1984. *Theory and Applications of Correspondence Analysis*. Academic Press, London.

HAN, B. 2003. Support vector machines. available at http://www.ist.temple.edu/~vucetic/cis526fall2003/lecture8.doc.

HELMER, G., WONG, J., S. K., J., HONAVAR, V., MILLER, L., AND WANG, Y. 2003. Lightweight agents for intrusion detection. *J. Syst. Softw. 67*, 109–122.

HOCHBERG, J., JACKSON, K., STALLINGS, C., MCCLARY, J., DUBOIS, D., AND FORD, J. 1993. NADIR: An automated system for detecting network intrusions and misuse. *Comput. Secur. 12*, 3 (May), 235–248.

HOOPER, P. 1999. Reference point logistic classification. *J. Classif. 16*, 91–116.

INSECUREORG. 2006. Nmap free security scanner, tools and hacking resources. available at http://insecure.org.

JACOBSON, V., LERES, C., AND MCCANNE, S. 2007. Tcpdump. available at anonymous@ftp.ee.lbl.gov.

KANNADIGA, P. AND ZULKERNINE, M. 2005. DIDMA: A distributed intrusion detection system using mobile agents. In *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel and Distributed Computing*. 238–245.

KDD. 1999. KDD Cup 1999 Data. available at http://kdd.ics.uci.edu/databases/kddcup99/.

KONE, M., SHIMAZU, A., AND NAKAJIMA, T. 2000. The state of the art in agent communication languages. *Knowl. and Inform. Syst. 2*, 3, 259–284.

LABIB, K. AND VEMURI, V. 2004. Detecting and visualizing Denial-of-Service and network probe attacks using principal component analysis. In *Third Conference on Security and Network Architectures (SAR'04)*. La Londe, France.

LAZAREVIC, A., ERTOZ, L., KUMAR, V., OZGUR, A., AND SRIVASTAVA, J. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of the Third SIAM Conference on Data Mining*. San Francisco, CA.

LEE, W. AND STOLFO, S. 2000. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inform. Syst. Secur. 3*, 4 (Nov.), 227–261.

LIAO, Y. AND VEMURI, V. 2002. Use of K-nearest neighbor classifier for intrusion detection. *Comput. Secur. 5*, 5, 439–448.

LIBCAP. 2007. Libcap. available at http://www.tcpdump.org.

LIU, H. AND MOTODA, H. 1998. *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers, Boston, MA.

LIU, H., YU, L., MANORANJAN, D., AND MOTODA, H. 2003. Active feature selection using classes. In *Proceedings of Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Seoul, Korea, 474–485.

MATHWORKS. 2007. Matlab. available at http://www.mathworks.com/matlabcentral/.

MOORE, D., VOELKER, G., AND SAVAGE, S. 2001. Inferring internet Denial-of-Service activity. In *Usenix Security Symposium*. Washington, D.C. 9–22.

MORENO, A. 2005. Medical applications of multi-agent systems. available at http://cyber.felk.cvut.cz/EUNITE03-BIO/pdf/Moreno.pdf.

OXID. 2006. Irs. available at http://http://www.oxid.it/irs.html.

PENTLAND, A., MOGHADDAM, B., STARNER, T., OLIYIDE, O., AND TURK, M. 1994. View-based and modular eigenspaces for face recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*. Seattle, WA, 84–91.

QUINLAN, J. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA.

SINGH, M. 1999. A social semantics for agent communication languages. In *Proceedings of IJCAI-99 Workshop on Agent Communication Languages*. Stockholm, Scandinavia, 75–88.

SNAPP, S., BRETANO, J., DIAS, G., GOAN, T., HEBERLEIN, L., HO, C., LEVITT, K., MUKHERJEE, B., SMAHA, S., GRANCE, T., TEAL, D., AND MANSUR, D. 1991. DIDS (distributed intrusion detection system)—motivation, architecture, and an early prototype. In *Proceedings of the 14th National Computer Science Conference*. Washington D.C. 167–176.

SPAFFORD, E. AND ZAMBONI, D. 2000. Intrusion detection using autonomous agents. *Comput. Netw. 34*, 4 (Oct.), 547–570.

TCPTRACE. 2007. available at http://www.tcptrace.org.

TOU, J. AND GONZALEZ, R. 1974. *Pattern Recognition Principles*. Addison-Wesley, MA.

VAIDEHI, K. AND RAMAMURTHY, B. 2004. Distributed hybrid agent based intrusion detection and real time response system. In *Proceedings of the First International Conference on Broadband Networks (BROADNETS'04)*. 739–741.

VERWORED, T. AND HUNT, R. 2002. Intrusion detection techniques and approaches. *Comput. Comm. 25*, 1356–1365.

WEKA. 2007. Weka. available at http://www.cs.waikato.ac.nz/ml/weka/.

XIE, Z., QUIRINO, T., SHYU, M.-L., CHEN, S.-C., AND CHANG, L. 2006. 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)*. Taichung, Taiwan, R.O.C, 446–453.

Received March 2006; revised January 2007; accepted May 2007