A Distributed Agent-Based Approach to Intrusion Detection Using the Lightweight PCC Anomaly Detection Classifier

Zongxing Xie, Thiago Quirino, Mei-Ling Shyu
Department of Electrical and Computer Engineering
University of Miami, Coral Gables, FL 33124, USA
zxie@umsis.miami.edu, grilocanibal@yahoo.com, shyu@miami.edu

Shu-Ching Chen
Distributed Multimedia Information System Laboratory
School of Computing and Information Sciences
Florida International University
Miami, FL 33199, USA
chens@cs.fiu.edu

LiWu Chang
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375, USA
lchang@itd.nrl.navy.mil

Abstract

In this paper, a novel agent-based distributed intrusion detection system (IDS) is proposed, which integrates the desirable features provided by the distributed agent-based design methodology with the high accuracy and speed response of the Principal Component Classifier (PCC). Experimental results have shown that the PCC lightweight anomaly detection classifier outperforms other existing anomaly detection algorithms such as the KNN and LOF classifiers. In order to assess the performance of the PCC classifier on a real network environment, the Relative Assumption Model together with feature extraction techniques are used to generate normal and anomalous traffic in a LAN testbed. Finally, scalability and response performance of the proposed system are investigated through the simulation of the proposed communication architecture. The simulation results demonstrate a satisfactory linear relationship between the degradation of response performance and the scalability of the system.

1. Introduction

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

There are numerous intrusion detection algorithms, frameworks and techniques. Generally speaking, the existing intrusion detection methods could be categorized into two main types: misuse detection and anomaly detection [4]. Misuse detection, which is based on signature modeling of known attacks [11], has the advantage of higher detection accuracy in detecting known attacks while its most obvious shortcoming is its incapacity of detecting previously unobserved attacks. On the other hand, anomaly detection, which is based on signature modeling of normal traffic [10], has the advantage of detecting new types of attacks [7], while it suffers from high false alarm rates. IDSs have undergone rapid development in both power and scope in the last few years. There are various types of architectures for IDSs. [21] summed up these systems into four main categories: monolithic, hierarchic, agent-based, and distributed (GrIDS) systems. However, much improvement could be done for these architectures, as the nature of the artificial attacks keeps changing.

Recently, the agent concept has been widely used in distributed environments because it provides many favorable characteristics including scalability, adaptability, graceful degradation of service, etc. over the non-agent based IDSs [12]. Most of the distributed agent-based IDSs introduce more traffic into their residing network, and the communication protocol between various entities is also an important aspect that has to be considered. At the same time, most of the designed agent-based IDSs [6] require comparatively high processing power in local machines to run the agents and other supportive software. Hence, a lightweight agent system with low network traffic generation requirements is needed.

In this paper, a distributed IDS with agent technology is proposed, where a set of classification agents communicate with each other and with the lower level agents to acquire a global scope of the security state of the network. It integrates anomaly detection and misuse detection, and focuses on the most common scenario of local machines having comparatively low processing power. It intends to detect heterogeneous intrusions in the network, and it seeks multiple sources of information to extract features for intrusion detection. There are two levels of agents in the proposed system. One is the host-agent layer, whose major function is to execute an anomaly detection algorithm and send the detected abnormal data instances to the upper-level agent. The Principal Component Classifier (PCC) developed in our earlier studies [17][18] is selected to take on this task due to its high detection rate and quick response. Our experimental results demonstrate that PCC outperforms the K-nearest neighbor (KNN) method [20] and LOF algorithm [5] with high detection rates and low false alarm rates. The second layer is the classification-agent layer, whose major function is to execute misuse detection on abnormal data instances supplied by the lower layer and notify the specified types of attacks to the host agents and other classification agents in the network.

This paper is organized as follows. Section 2 presents our proposed agent-based distributed IDS. The proposed Relative Assumption Modeling and feature extraction techniques together with the performance comparison among PCC, KNN and LOF are described in details in Section 3. Section 4 demonstrates analysis results on the communication among different classification agents and host agents. Conclusions are given in Section 5.

2 The proposed agent-based distributed IDS

Our proposed agent-based distributed IDS consists of a host layer and a classification layer. Figure 1 illustrates the proposed IDS and the LAN testbed network setup.

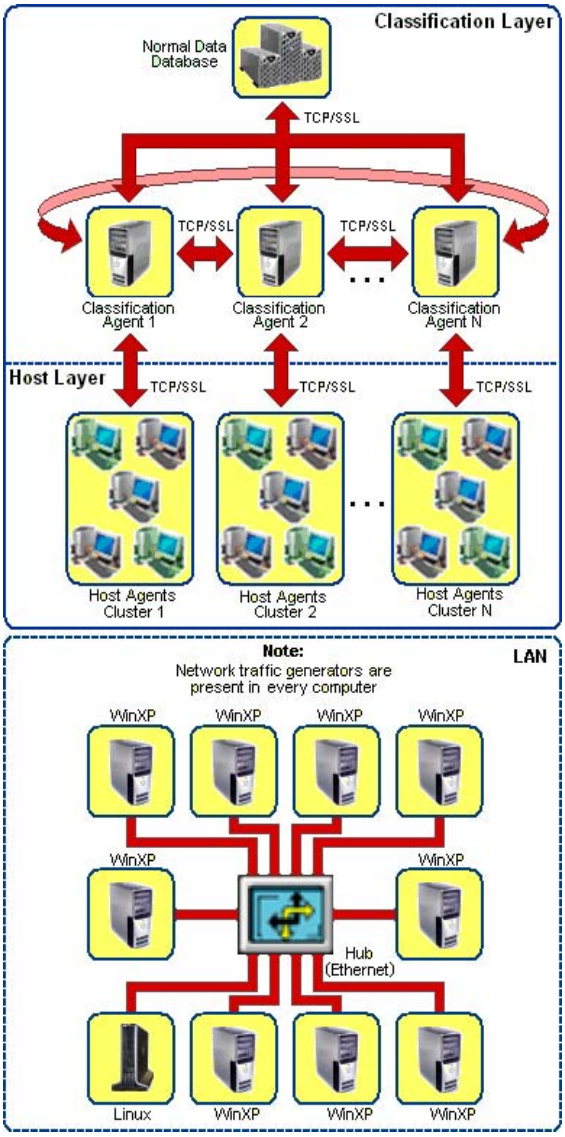

Figure 1. Architecture of the proposed system and LAN testbed network setup

2.1 Host layer

The host layer consists of lightweight host agents that run as background processes in end-user machines. These agents collect information about the network connections in their hosts and classify these connections using the PCC classifier. Each host agent is connected to an upper-layer agent called a classification agent, to whom the host agents report connections found to be abnormal by PCC. Host agents are mainly concerned with detecting abnormal activities occurring in their hosts and properly responding to them. Virtually, every machine in a network can be con-

sidered as a host agent. Since every connection in a host, whether normal or abnormal, is analyzed for abnormality, the host agent can keep the information of a few normal connection instances and pass it along with abnormal connection instances to the upper layer classification agents to be saved in a database so that the classifier can be re-trained at a later time.

PCC, which has proven to be a fast, highly accurate, lightweight classifier [17][17] suitable for the implementation of the lightweight agents required by the limited processing power end-user machines, was chosen as the classifier at this layer. PCC is derived from Principal Component Analysis [9], a mathematical technique that tries to capture the variability in a data set into the so-called Principal Components. Principal Components are perpendicular vectors that point in the direction of the largest variances found in the distribution of a data set, and can be utilized as coordinate axes to which the original data set can be projected upon. Each principal component describes a certain percentage of the total variability found in the data set, and thus contains useful information about the distribution of the data. Upon training a PCC classifier with an instance set of normal connections, the classifier differentiates very well between normal and abnormal connection instances. The PCC classifier training time and classification time are considerably shorter than other methods such as KNN and LOF, and our experiment results demonstrate that PCC achieves higher classification accuracy than that of KNN and LOF at similar false-alarm rates.

2.2 Classification layer

In the classification layer, the classification agents attend to the concern of host agents and their suspicion of a possible attack. Multiple host agents connect to a single classification agent and rely on their respective classification agents to classify the abnormal connection instances found in their hosts through PCC into known attack types, a task performed by the classification agent through a misuse detection algorithm. This is important as the attack type will determine the proper response of the IDS to the intrusion. Research effort is currently invested on the development of a suitable misuse detection algorithm at the classification agents, and hence in this paper, our focus is given to other aspects of our proposed architecture. Agents in this layer are assumed to be running in dedicated machines capable of delivering the processing power required to handle all classification requests of their host agents, in addition to handling the communication burden of warning other fellow classification agents of a possible network attack in progress in their node. This communication between classification agents is important 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 authorization.

2.3 Communication between two layers

Communication is an important factor in the design of an effective distributed agent-based system. In our proposed system, a message passing protocol through TCP/IP Security Server Lever (SSL) is used for the implementation of secure communication among agents. Every possible communication event that can arise between the layers was taken into consideration in the design of our proposed communication protocol. We propose a standard message composed of the following four descriptive fields and a payload space that can be used to carry the parameters of the different message types:

1. Message ID: A message number that the sending agent provides and uniquely describes that message.
2. Source Agent: The IP address of the source agent that can be used for verification and reply.
3. Destination Agent: The IP address of the destination agent so that an agent can verify that the incoming message was truly intended for it.
4. Message Type Number: A number describing what the message is about: a request for classification, a warning of a possible attack, an acknowledgement, among other possible events.
5. Payload: The Message Type Number field described above identifies what the message is about. Suppose the message type number identifies a request from a host agent for classification of an abnormal connection instance by a classification agent. The abnormal instance information would go in the payload section. Thus, the payload section is structured according to the requirements of the message type and carries different parameters and fields depending on the message type number.

Every communication event is finalized by an acknowledgment (ACK) message. An agent, upon receiving a message, sends an ACK message to the source agent to inform the receipt and further processing of the message.

3 Host agents

The major function of the host agent layer is to execute anomaly detection and send the abnormal connections to the upper level. Different anomaly detection systems may use different types of data in the process of intrusion detection.

Depending on the quality of the data, anomaly detection systems behave differently in detecting intrusions. Therefore, it has become a vital aspect for anomaly detection to identify suitable data types. Recently, most anomaly detection research is based on KDD [2] or MIT [1] data sets, both of which provide labeled or separate well-predefined data sets including Normal, DoS, Probe, R2L, U2R, etc. It is unavoidable in research studies to make use of only some familiar attack models to represent all anomaly data models for either training or testing, which leads to the fact that the studies always make hasty generation to analyze and estimate accuracy, efficiency, functions, techniques or other criteria of various methods, algorithms and frameworks. At the same time, acquiring real network intrusion data is difficult due to security, privacy and other realistic issues. In order to avoid this dilemma, an agent-based distributed network testbed is developed and employed to simulate the real network environment based on our proposed Relative Assumption Modeling.

3.1 Relative assumption modeling

It is a basic fact that both the so-called normal and abnormal data sets are relative concepts, which are decided by network bandwidth, server processing capability, average network load, and other factors. 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 less 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 fleetly.

Next, 5 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 which are in the form of opposite core phases from 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. Finally, these abstract phases with different numeric values in different networks should yield good relative and adjustable generated traffic patterns.

In our experiments, as shown in details in Section 3.3, the focus is on TCP connection generation since most attacks are executed via the TCP protocol. This is due to TCP's frangibility and instability. For example, a survey showed that 90% to 94% of Denial of Service (DoS) attacks, which are the major threats to the whole Internet, are employed via TCP connections [15].

3.2 Feature extraction (FE)

For any intrusion detection algorithm, feature extraction (FE) is important since it can drastically affect its performance [14]. The network data contains all the required analysis information and can be extracted from data packets transferred through the network. Before the features are extracted from the network data, the information from the network traffic needs to be collected and stored. Usually, the network data is in a raw format and further processing is needed to extract useful features before the data is suitable to be fed into intrusion detection methods. In this paper, the following three steps are proposed to extract features that are critical in network intrusion detection:

- Windump: the windows version of Tcpdump [8] which uses libcap [13] 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 [3]: a tool used to produce several different types of output containing information such as elapsed time, bytes and segments sent and received, retransmissions, round trip times, window advertisements, throughput and more by analyzing the Windump files, is used to extract basic information on each TCP connection from the Windump data.
- Our own FE techniques are used to extract basic features and create time-based, connection-based, and ratio-based features form the Tcptrace output file.

Four main feature sets are extracted from Tcptrace to describe a connection:

1. *Basic Features*: basic information related to the connection. Totally, seventeen basic features are extracted which include duration, IP, port, total packets, ack packets, throughput, etc.
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 which have the same IP and port. Totally, 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 these features correspond to one or more of the core phases described in Section 3.1. For example, the first feature "duration" represents the core phase ("during a moderate time period", "during a very short or very long time period") directly. Yet another example is the ratio features which reflect the core phase ("suitable rate", "increasing rate"). Detailed information about these relations is shown in Table 1

Table 1. 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

3.3 Anomaly detection

Three anomaly classification methods, namely PCC, LOF and KNN (k=5), are used to test the feasibility of our Relative Assumption Modeling and FE techniques in addition to providing a comparative study between PCC and the other anomaly detection methods. The experiments are organized in a similar manner as mentioned in [17][18].

1. All the outlier thresholds are determined from the training data.

2. The false alarm rate is varied from 1% to 10%.

3. For the PCC method, the thresholds are chosen based on $\alpha_1 = \alpha_2$ false alarm rate values.

4. The accuracy of a classifier is measured by the percentage of correct classification.

5. The Precision and Recall values [22] are used: Let TP and FP be the numbers of correctly and falsely detected abnormal connections, and TN and FN be the numbers of correctly and falsely detected normal connections.

   Precision for abnormal = TP/(TP+FP);

   Recall for abnormal = TP/(TP+FN);

   Precision for normal = TN/(TN+FN); and

   Recall for abnormal = TN/(TN+FP).

Based on the practical experience of observing traffic conditions and parameters such as network delay, CPU usage, and memory allocation associated with our LAN testbed network as well as the proposed Relative Assumption Modeling approach, the relevant typical normal values for our LAN testbed network are set as follows:

- Generate no more than ("not much") $3.5k/s$ (5 $packets/s \times 700\, bytes/packets$) data transfers "during a moderate time period" of about 5 seconds in "suitable rate" (ratio < 1.5), "suitable frequent" (< 5 in the last 100 connections), and "suitable pace" (< 3 in the 3-second sliding Window).

- Next, the typical abnormal values are set by the proposed relatively reversing method, i.e., generating "much" increasing data transfers ($> 3.5k/s$) at an "increasing rate" (ratio > 1.5) "during a very short or very long time period" (< 5 seconds or > 5 seconds), "continually" (> 5 in the last 100 connections) and "fleetly" (> 3 in the 3-second sliding window).

- Finally, based on the above values, the IP-Traffic traffic generator tool [19] is employed to generate and label one group of network connections as "normal" (12,932 TCP connections) and thirty-one groups as "abnormal" traffic data sets (7,618 TCP connections). This tool is capable of generating full-duplex TCP connections in random, increase, or decrease modes within an appointed range of values, and of controling network parameters by varying the inter-departure time between packets and the packets' sizes. These generated data sets serve as the training and testing data sets of our experiments, from which 3,000 "normal" connections are randomly chosen to train the classifiers, and the remaining others are used for testing.

In addition, in order to further test the feasibility of our Relative Assumption Modeling and FE techniques, and to determine the universality of the "normal" connections generated through our methods, 4,000 DoS connections from the MIT [1] tcpdump data set (LLDOS2.0.2.) are added to the testing set as abnormal instances.

Table 2 shows the detection rates of the three anomaly detection methods, i.e., the accuracy of abnormal data detection in the testing set. From this table, it can be seen clearly that PCC maintains a high detection rate ($> 90\%$) and always outperforms LOF and KNN, especially in the lower false alarm rate range of values. These differences can also be observed in the ROC curves in Figure 2.

Table 2. Detection rates of the three anomaly detection methods

False Alarm	PCC	LOF	KNN (k=5)
1%	90.91%	77.26%	76.95%
2%	93.68%	80.32%	77.33%
4%	95.75%	81.01%	77.90%
6%	97.35%	88.02%	78.67%
8%	97.97%	96.25%	78.98%
10%	98.42%	97.19%	82.20%

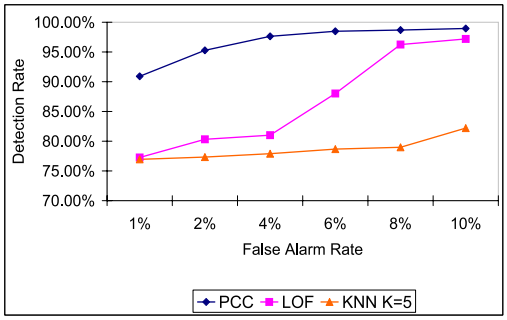

Figure 2. ROC curves of the three anomaly detection methods

Table 3 shows the observed false alarm rates of PCC when compared to the rate used to train the method, which is the percentage of misclassification of the normal data in the testing set. From this table, it is clear that the observed false alarm rate of PCC is below the values used to train the classifier. This result demonstrates that PCC fulfills its performance expectations.

Whenever the classification methods present similarly high detection rates (e.g., 8% and 10% false alarm rates), their corresponding recall and precision values are compared as shown in Table 4. It can be easily seen from this table that PCC always maintains high recall and precision ($> 90\%$) values, and also outperforms LOF and KNN (k=5). Furthermore, the training and classification speeds of the three methods are also observed, where PCC performs much faster than LOF and KNN, especially in comparison to the LOF method. Classification speed is a crucial factor that must be considered in the implementation of a system capable of real-time response. All these experimental results show why PCC is selected as the anomaly detection method in our host agents.

Table 3. Observed false alarm rate of PCC

Initialized False Alarm Rate	PCC: Observed False Alarm Rate
1%	0.91%
2%	1.46%
4%	2.34%
6%	3.08%
8%	3.88%
10%	4.67%

Table 4. Precision and Recall Comparison of the three anomaly detection methods

	False Alarm	PCC	LOF	KNN (k=5)
Recall (Abnormal)	8%	98.77%	96.25%	78.98%
	10%	99.04%	97.19%	82.20%
Recall (Normal)	8%	93.65%	91.08%	92.28%
	10%	92.49%	86.14%	90.24%
Precision (Abnormal)	8%	96.12%	92.66%	92.29%
	10%	95.33%	89.13%	90.79%
Precision (Normal)	8%	97.97%	95.40%	78.96%
	10%	98.42%	96.33%	81.25%

In addition, one of the significant contribution of this study is that the generated data sets which are based on the proposed Relative Assumption Modeling method and the FE techniques in a large extent can also be successfully used as either training or testing data sets in the experiments where the standard KDD data sets were used [17][18]. Such practical results validate the feasibility and universality of our proposed Relative Assumption Modeling method and FE techniques.

4 Agent communication performance

To assess the relationship between scalability and the burden of the increased agent communication on a real network environment through the use of our proposed communication protocol, the following experiment is conducted. First, a host agent detects an abnormal connection instance. Next, the host agent requests the classification of the attack type from the classification agent to which it is connected. Upon classifying the attack type, the classification agent sends classification results back to the host agent, and then informs the other classification agents of the possible network attack.

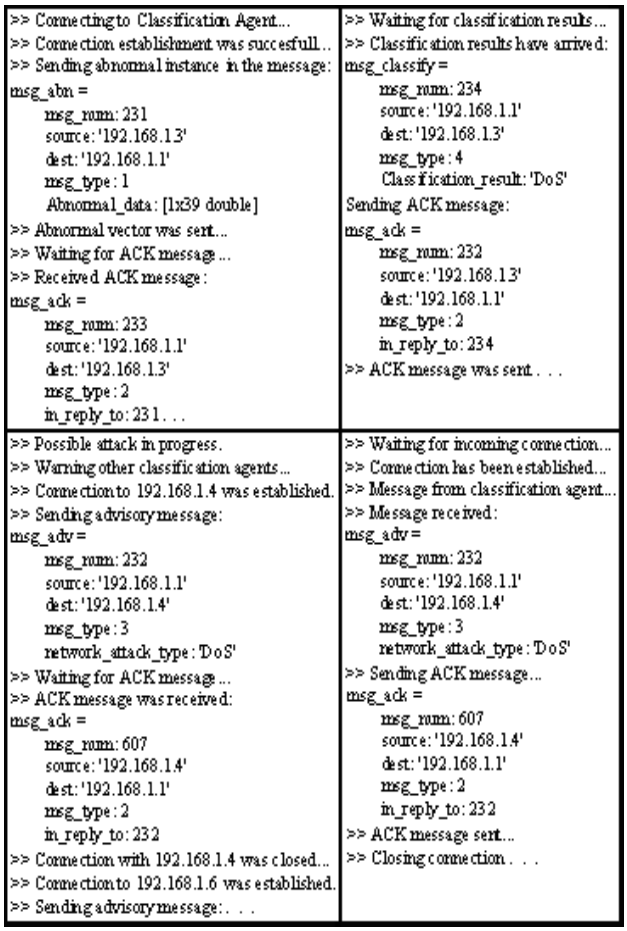

Figure 3. Screen shots of the communication between agents

Let the time period between the detection of an abnormal instance by the host agent and the classification agent informing the entire network of classification agents of the possible attack be the response time of the IDS. The experiment makes a few assumptions such as the fact that the effectiveness of the agent communication will be tested in the absence of real network traffic or network attacks, and misuse detection is ignored at the classification agent layer. Nevertheless, by varying the number of classification agents that have to be informed of a possible attack from the low to high values, the experimental results illustrate the pattern in the increase of response time as more classification agents are introduced into the IDS grid, thus revealing the practicality of the proposed communication protocol and the scalability of our proposed distributed IDS. To perform this experiment, the communication between agents was simulated by using Matlab-TCP/UDP/IP Toolbox [16]. Each computer in the LAN testbed network runs an equal number of multiple Matlab sessions to simulate the required number of classification agents for the experiment in addition to the host agent.

The simulation yielded as a result the generation of numerous messages that follow the communication protocol described in Section 2.3. The simulation was absent of errors and the messages reached their destination agents properly, thus eliciting the appropriate response by the agents. An illustration of the messages generated during the simulation is shown in Figure 3, which is divided into two parts: the top-half illustrates the messages generated by the host agent in response to an abnormality detection and the communication with a classification agent, and the bottom-half illustrates the messages generated between classification agents during the task of warning the network of a possible attack. Though simple and compact in sizes, the messages were found to properly convey the request, warning, or acknowledgment information they were meant to.

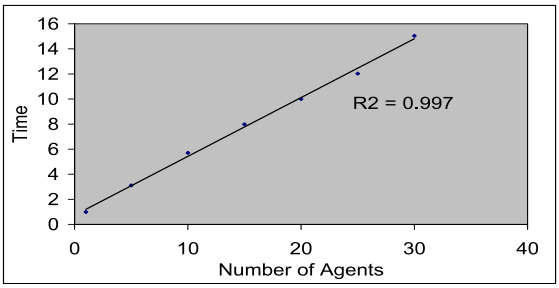

Figure 4. Response time versus scalability

The experimental results also illustrates a linear relationship between the response time and the number of classification agents introduced into the system, an indication that the response time performance of the system will degrade linearly with scalability. Figure 4 illustrates the high linearity of the result through the regression line correlation coefficient of $R^2 = 0.997$. This is an important result as mature knowledge on network programming techniques can lead to the development of much faster communication between the agents than the one achieved through Matlab simula-

tion, thus decreasing the slope of the line in Figure 4 and the degrading effect that scalability has on the response performance of the system.

5 Conclusion

In this paper, a two-layer architecture for an agent-based distributed intrusion detection system (IDS) was proposed. Our proposed architecture provides an efficient communication between agents, introduces a small amount of network traffic by the distributed intrusion detection procedure, and takes the full advantage of distributed network resources to achieve an excellent performance in intrusion detection. The concept of the lightweight agent, an agent software that does not require high processing power, plays an important role in the design of our system as it takes into account the finite and shared processing power that most end-user machines connected to a network can provide. Based on the proposed distributed architecture, a testbed local area network (LAN) is built. The proposed Relative Assumption Modeling method, FE techniques, and PCC classifier method were employed to generate the test traffic and to simulate anomaly detection in the host agent layer. The performance comparison among the PCC, LOF, and KNN methods shows that PCC can work efficiently and with high accuracy, low false alarm rates, and timely response. Thus, PCC was found to be desirable for use as the standard anomaly detection method to detect the abnormal connection instances in the host agent layer of the proposed IDS architecture. Due to these inheritant features of the proposed IDS architecture, a system which only lightly loads the network it resides in and still maintains a global view of the events taking place in the network is provided. The experimental results on the increased agent communication versus response time verify that our proposed IDS provides many favorable characteristics such as scalability, adaptability, and graceful degradation of service.

6 Acknowledge

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.

References

[1] The information systems technology group (IST) of mit lincoln laboratory, darpa intrusion detection evaluation data sets. *http://www.ll.mit.edu/*, 1998.

[2] Kdd cup 1999 data. *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[3] Tcptrace. *http://www.tcptrace.org*, 2005.

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

[5] M. M. Breuning, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *ACM SIGMOD Conference*, pages 93–104, May 2000.

[6] D. Dasgupta and H. Brian. Mobile security agents for network traffic analysis. *DARPA Information Survivability Conference and Exposition*, 2:332–340, June 2001.

[7] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, and J. Ford. Nadir: An automated system for detecting network intrusions and misuse. *Computer and Security*, 12(3):235–248, May 1993.

[8] V. Jacobson, C. Leres, and S. McCanne. tcpdump. *anonymous@ftp.ee.lbl.gov*, June 1989.

[9] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, second edition edition, 2002.

[10] K. Labib and V. Vemuri. Detecting and visualizing denial-of-service and network probe attacks using principal component analysis. *Third Conference on Security and Network Architectures*, SAR'04, June 2004.

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

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

[13] Libcap. *http://www.tcpdump.org*, 2005.

[14] H. Liu, L. Yu, D. Manoranjan, and H. Motoda. Active feature selection using classes. *Seventh Pacific-Asia Conference on Knowledge Discovery and Data Mining*, LNAI2637:474–485, May 2003.

[15] D. Moore, G. Voelker, and S. Savage. Inferring internet denial-of-service activity. *Usenix Security Symposium*, pages 9–22, August 2001.

[16] P. Rydesater. Matlab-TCP/UDP/IP toolbox. *http://www.mathworks.com/matlabcentral/*, 2005.

[17] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang. Principal component-based anomaly detection scheme. *Foundations and Novel Approaches in Data Mining*, pages 311–329, In press.

[18] M.-L. Shyu, K. Sarinnapakorn, I. Kuruppu-Appuhamilage, S.-C. Chen, L. Chang, and T. Goldring. Handling nominal features in anomaly intrusion detection problems. *The 15th International Workshop on Research Issues on Data Engineering (RIDE), in conjunction with The 21st International Conference on Data Engineering*, pages 55–62, April 2005.

[19] Z. Telecom. Ip-traffic: an ip network monitoring and testing software. *http://www.zti-telecom.com/pages/main-ip.htm*, 2005.

[20] J. Tou and R. Gonzalez. Pattern recognition principles. 1974.

[21] T. Verwored and R. Hunt. Intrusion detection techniques and approaches. *Computer Communications*, 25:1356–1365, 2002.

[22] Y. Yang. An evaluation of statistic approaches to text categorization. 1999.