Agent-based Intrusion Detection for Network-based Application

Jianping Zeng[1] and Donghui Guo[2]
(Corresponding author: Donghui Guo)
Department of Physics, Xiamen University, Fujian 361005, China[1]
Department of Electronic Engineering, Xiamen University, Fujian 361005, China[2]
(Email: dhguo@xmu.edu.cn)
(Received Apr. 20, 2006; revised and accepted Sept. 25, 2006)

Abstract

Now days, different kinds of IDS systems are available for serving in the network distributed system, but these systems mainly concentrate on network-based and host-based detection. It is inconvenient to integrate these systems into distributed application servers for application-based intrusion detection. An agent-based IDS that can be smoothly integrated into the applications of enterprise information systems is proposed in this paper and we discuss the system architecture, agent structure, and integration mechanism. Our IDS system consists of three kinds of agents, namely, client agent, server agent and communication agent. This paper also explains how to integrate agents with an access control model for getting better security performance. By introducing standard protocols such as KQML, IDMEF into the design of agent, our agent-based IDS shows how to build more flexible software applications.

Keywords: Agent-based, IDMEF, intrusion detection, KQML

1 Introduction

Many application services, such as e-business, remote education and Internet-based design, etc, are necessary to be distributed over the Internet. However, because the Internet is an open society so that anyone can access the resource on it, then the application system may confront with all kinds of attacks or intrusions, such as Denial of Service (DoS), port scan, illegal intrusion by hacking user information, etc.

Of all these security events, illegal intrusion is a more serious issue. But standard security deployments such as firewalls are limited in their effectiveness because of simple access control mode and also the intrusion methods are evolving fast. Once an attacker has breached the firewall, he can roam at will through the network [13]. This makes intrusion detection system (IDS) very important and necessary. Traditionally, there are two main classes of IDSs: host-based and network-based systems. A host-based IDS monitors the detailed activity of a particular host, while network-based IDS monitors networks of computers and other devices such as, routers, gateways, and primarily detects intrusion by sniffing and analyzing data from network traffic. Network and host-based IDSs, can be further classified based on two methods of detection [17]: anomaly detection and misuse detection.

However, Masquerading attack is a typical intrusion and it can be a more serious threat to the security of computer systems and the computational infrastructure [15]. By this kind of attacks, an assailant attempts to impersonate a legitimate user after gaining access to this legitimate user's account. So, the assailant can fully understand the information he gets. While by other kinds of attacks, he just gets some segment of data or encrypted data, which is much more difficult to be understood. A well-known instance of masquerader activity is about a FBI mode [15]. Since masquerading attack happens exactly at application layer, we call the methods to detect this kind intrusion as application-based intrusion detection. Application-based intrusion is more difficult to be detected and the detection program is usually unable to get satisfactory performance for the following reasons:

1) Data about user action on system is much more difficult to be collected than network-based or host-based detection, because of the independency of application system.

2) Application-based detection can degrade the performance of corresponding application system due to the extra work that should be done in the process of data collection and analysis.

3) A masquerader may happen to have similar behavioral patterns as the legitimate user, therefore it can escape detection and successfully cause damage under the cover of seemingly normal behavior [4].

Fortunately, there have been several attempts to tackle the problem of detecting masqueraders. Several masquerade-detection algorithms, such as Sequence-Match, IPAM, Bayes 1-Step Markov, etc are presented by Schonlau together with his colleagues and many other researchers [15, 18, 24, 27, 31]. However, the performance of those detection methods is still unsatisfied, for example, the detection rate is still low in order to keep the false alarm rate at an expected lower level. One of the main reasons is that user feature is improperly selected. Most of those detection methods only use the sequence of system call that is related to the user sessions as the feature of user action. Obviously, the insufficient knowledge about users will lead to frequent wrong decision on whether the user is masqueraded or not. So, extra user feature should be collected and used in the detection process. On the other hand, when building an effective system for masquerade detection, several issues should be considered in addition to the detection algorithm, such as data collection, system structure and the ability of smooth integration with current application systems. They are particularly important for getting a good performance of the detection algorithm and relieving the burden, such as higher memory or too much CPU time requirement, on the computers where user application systems are running.

And we believe that agent technology can be a better choice to deal with these issues when implementing an intrusion detection algorithm. Agent is a software entity that functions continuously and autonomously in a particular environment, and is able to carry out activities in a flexible and intelligent manner that is responsive to changes in the environment [3]. So, agent can improve the means of applying detection techniques, for example, we can deploy agents at different user computers to collect extra feature data and agents can also provide an interface to user application systems for smooth integration. So, applying agents to the intrusion systems can provide a good mechanism for implementation of detection algorithm on network-based application systems.

In this paper we mainly focus on building an agent-based system for detecting masquerade intrusion in network-based application systems. And the specific objectives and the main contributions of this research are:

1) propose an agent-based architecture that can be smoothly integrated into a network-based application systems, without affecting the performance the user systems too much while keeping a good detection performance.

2) design a new mechanism for acquiring extra data about user action from client machines or from access control module in server applications.

3) design and implement a detection algorithm and show its effectiveness in network-based application systems.

To reach these goals, AIDSI (Agent-based Intrusion Detection System for Integration) is proposed in the paper. The system consists of three kinds of agents, i.e. client agent, server agent and communication agent. The system architecture, agent structure, integration mechanism, etc, are mainly discussed. Being different from other agent-based IDSs, AIDSI can be integrated with enterprise information system very well and achieve better security performance.

The paper is organized as follows. Section 2 describes related work and discusses previous efforts to utilize agent based or non-agent based techniques for intrusion detection, and provide a comprehensive discussion of them. Section 3 introduces our system that can be integrated with user application systems and performs application-based intrusion detection. And we mainly focus on system structure, detection mechanism, etc. Section 4 describes the implementation of the system in detail, discusses the integration of AIDSI with a real network-based application system, and presents some performance of testing. Finally, in Section 5 we briefly conclude the paper and point out the future work.

2 Related Work

As far as the realization methods of the intrusion detection systems are concerned, there are two main classes of approaches, namely, agent based and non-agent based methods.

Non-agent based IDS usually monitors the activities of a single host or collects information of a special network with the deployment of a network IDS in a critical point of the network, and then and the data is analyzed by a single module using different techniques [8, 10].

Obviously, the data collection, analysis and detection may spend too much CPU time and memory, which can cause the user application systems running in the computer to become bluntness. On the other hand, the central analyzer is a single point of failure, which will make the reliability of the system very poor. And the system scalability is limited, it is difficult to reconfigure or add capabilities to the IDS. So, the non-agent based IDS is difficult to be successfully used in a larger scale network.

In order to perform a good detection in the network, there has been a trend to build agent-based IDS for network applications in recent years.

Agent can be used to collect data from network or hosts by several data acquiring technology. For example, sniffing agent monitors in network [9]. SNMP agent reads event from MIB database [7]. Sensor agent directly reads from files [23]. However, the reliability of the agent is mostly important to make sure that the data is correct, so mobile agent is employed into IDS to help the design of data collecting agent. Because mobile agent is able to migrate from host to host on a network under its own control and it do not require network connectivity with remote service to interact with it [26], so it is more reliable.

On the other hand, agent is usually used to analyze the data provided by data collection agent in IDS. To make a correct decision on input data, this kind of agent is usually required to have intelligence. And this is realized by incorporating intelligent technology into the design of the agent, such as immunity [19], genetic algorithm [29] and fuzzy computation [6].

Also, there may be other kinds of agent, such as reporting agent, alert generating agent. Although these agents can run separately in a IDS, it is a trend to organize these kinds of agents into a well group with capabilities to interact with each other to detect intrusion by distributing agents reasonably or making agent autonomous [2].

Several agent-based IDSs have been built or proposed in recent years. Autonomous Agents For Intrusion Detection (AAFID) [1] is proposed in early year and focuses on network-based intrusion. The AAFID system consists of three essential components: agents, transceivers and monitors. Agents are used as the lowest-level element for data collection. All agents in a host report their findings to a single transceiver. However, the communication protocol between agents or transceivers is based on SNMP, System V IPC, this may lead to extra work when developing a new agent.

In 2003, Hegazy built a multi-agent based intrusion detection system [9]. It is also a network-based detection system. The system employs sniffing agent, analysis agent, decision agent and report agent to detect three kinds of attack: the Denial of Service attack, the ping swept attack and the secure coded document theft.

Although there are many other network-based or host-based detection systems based on agent technology now [11, 20], the application-based detection systems are still rarely found since there exists many difficulties in an application-based detection, such as we list in the previous section.

In 2002, an intelligent agent security intrusion system is built [21]. The system is an application-based detection one. It utilizes the Bayesian multivariate statistical model to predict user action. In such a system, a user agent resides in a user workstation, while a core agent resides on the server. User agent monitors and analyzes the user action according user profile that is downloaded from server via core agent. The file contains rules that describe the legal past behavior of the user and the statistical predictions. Therefore, the file must be kept secure by itself; however, it's difficult to be ensured in this system.

Although these agent-based IDSs may be effective in detecting some kinds of intrusion in different operating system respectively, but as we can see, the IDS and the user application system are usually difficult to be integrated together, and this will lead to the delay in data collection and thus lengthen the response time in an application-based detection system. As a result, how to integrate the IDS with all kinds of application in enterprise information system is a critical problem to be resolved. In the research of AIDIS, we focused on the ability for the AIDSI to be smoothly integrated into user application systems. However, there are many issues to be considered, such as the communication mechanism between the two systems, how to keep a good overall performance when the detection system is running on the application system.

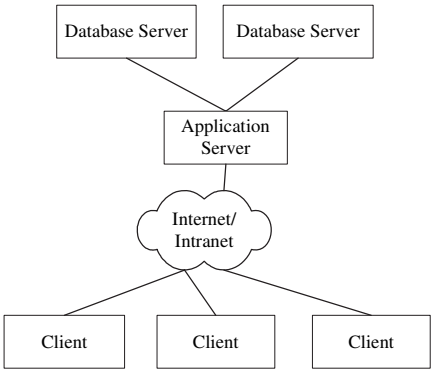

Figure 1: Structure of a typical network-based application

3 Architecture and Algorithms of Aidsi

3.1 System Architecture

In a typical network-based application, there are one or more application servers, database servers and clients. And the corresponding logical architecture is shown in Figure 1. Several application tools are running on application server, while any data or configuration information is usually stored in databases. Thus application server and database server becomes a critical part in such a structure. To ensure the security of the system, clients are not allowed to visit the database server directly and the visiting to the application server is based on strict user authentication.

In order to ensure the security for such a system, especially to detect to anomaly intrusion into the server, we can design a system for building IDSs that can be distributed over network-based applications. In this system, we use HMM (Hidden Markov Model) to model the normal user's action, thus a user model can reflect the user's activities in the application statistically. By computing the probability of a user action sequence that is comprised of commands on the application, we can then make decision on whether the user is illegal or not. The user model is automatically learnt, created and updated by using the audit data of user action on the application servers. There are three types of agents in this system, i.e. client agent, communication agent and server agent. The relationship among them is shown in Figure 2.

Client agents are installed on a client workstation, and responsible for collecting extra user information and then sending it to server agents with the help of communication agents.

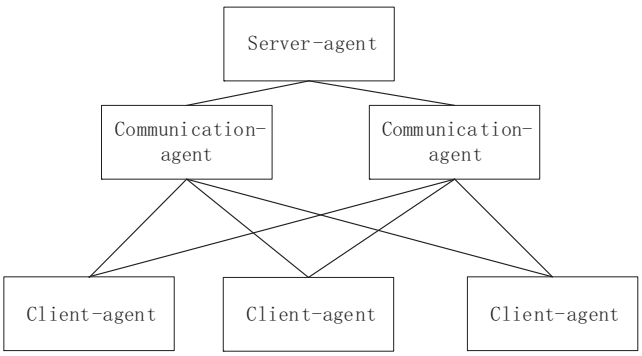

Figure 2: Relationship among agents

Server agents are running in the server where masquerade intrusion is to be detected. They process the message sent from client agents, read from and write to user model. However, most important of all, server agents can make a decision on whether the current user is a legal one or not according to the prediction model.

Communication agents monitor the client agent's request. After the received message is parsed, the useful message is forwarded to the server agents. Also, the communication agents can accept other message that generated from access control module.

In the following section we describe each component in detail.

3.2 Server Agent

The server agent has several main tasks to be performed, and we describe them in details as follows:

1) Create a suitable model for each user

Although intrusion can be detected on line or off line, the AIDSI focuses on on-line detection, so we need a proper description of normal user, namely, a user model. The AIDSI defines different models for each user. There are several alternative choices of model that can be used to describe the action of a normal user. For example, the sequence database model, markov model, etc. Warrender represented a normal user by a database of user activities short sequence [28], then, many kinds of intrusion can be detected by matching the user action sequence with the database. The main disadvantage is that the size of the sequence database will increase too much in order to get a good detection performance, and the maintenance of the database becomes very difficult. Although, the size of database can be reduced by transferring short sequence to HMM hidden state sequence [30], the process for updating and detecting become complex.

If we exam the system calling sequences that are related to a user session, we can find that the sequences can be described by a Hidden Markov Model very well [28].

The HMM model is described by a quintuplet, i.e. $\lambda = (N, M, A, B, \pi)$, and these parameters are explained as follow, and detailed description can be found in [22].

N is the number of hidden states in the model, M is the number of distinct observation symbols per state, π is the initial state distribution, A is the transition probability distribution between hidden states and B is the observation symbol probability distribution.

HMM can be treated as two stochastic processions, the first one is a Markov model described by parameter A, and its output is feed to another stochastic procession described by B. Furthermore, there are many extended HMM models, such as IO-HMM, Factor HMM. These HMMs can be used to model much more real-world application, therefore, we adapt HMM to model user's activities on server.

The HMM model is trained by using Baum-Welch algorithm [22]. The algorithm can make sure that the likelihood of sequence with respect to the HMM increases after each iteration in the training. Action events sequences with fixed-length are extracted from each trace of training set by moving a window of specified width. And these training sets are constructed from the event database. In our system, these events are recoded by BSM (Basic Security Module) provided by SUN Solaris system [25]. However, we use an advance data mining algorithm to filter those events that may seems abnormal, because abnormal user sequence are not allow to be included in train set. So this data-mining algorithm should produce the sequence of normal user action event with high certainty, we use certainty factor to evaluate the similarity of stochastic patterns of sequence data. The detail of this algorithm is described in our other paper [33].

After creating the HMM user model, we perform a simple computation to generate the probability of sequence with certain length, i.e. suppose

$x = p(O|\lambda), O$ is the sequence.

Then, a probability distribution $P(x)$ on x is generated for this model.

2) Get the events of user action in real time

By reading the audit data generated by BSM in a certain sliding window size, we can construct an event sequence on line, which includes all events since last reading. However, how to determine the window size is an important issue. If the value is set to a larger numeric, the complexity of the computation will increase, while the detection accuracy will degrade if the value is set to a small numeric. In this system, we use theory of entropy to adaptively determine the optimistic value for different users [34].

Another method to get user action events in real time is through access control module. After the communication agent gets the KQML (Knowledge Query and Manipulation Language, which is a relatively mature agent communication language [12]) package about user action from the instance of access control module, the server agent can simply perform a standard action on KQML package to acquire the user action in real-time.

3) Compute the distinguished value for the sequence and make decision

This is calculated by HMM by the following equation for a given sub-sequence O, that is,

$$\begin{aligned} x &= p(O|\lambda) \\ Dis(x) &= \frac{P(x)}{\max_{\forall y}(P(y))} \end{aligned}$$

However, to avoid making wrong decision on the sudden action of normal users, an average $Des(x)$ is computed on a sequence with a certain length. So the final distinguished value for the sequence is

$$T_1(x) = \frac{\sum_{i=1}^{lin-win+1} Des(x_i)}{len - win + 1}$$

where, win is the sliding window size, and len is the sequence length.

Another two features are calculated in the similar way. Suppose $P_1(t_1, t_2)$ is the probability distribution of user login time between t_1 and t_2. $P_2(x)$ is the probability distribution of client ID where the user login. And $P_1(t_1, t_2)$ and $P_2(x)$ should be set up first. Then, for a given user login time and client ID, we can calculate their corresponding distinguished value, that is,

$$\begin{aligned} T_2(x) &= \frac{P_1(t_i, t_j)}{\max_{t_x, t_y}(P_1(t_x, t_y))} \\ T_3(x) &= P_2(x). \end{aligned}$$

After getting the value of $T_1(x), T_2(x), T_3(x)$,the user's final distinguished value can be obtained by the following equation,

$$T(x) = \frac{\sum_{i=1}^{3} T_i(x)}{3}$$

Then, a decision can be made by checking whether formula

$$T(x) \leq \xi$$

is satisfied or not, where ξ means the belief of the normality of user action provided by the operator.

3.3 Communication Agent

The main function of communication agent is to receive the message from client agent. The message contains extra user information and it is in KQML format. So the agent should extract the useful data and send it to the server agent.

Another main task done by the agent is to receive user action event data sent from the instance of access control model.

We define a standard event data acquiring protocol based on IDMEF (Intrusion Detection Message Exchange Format). IDMEF [5] is a XML-based language to describe intrusion events. Thus, access control module in user application system can encapsulated the user login information and access information into IDMEF package in real time, and the communication agent provides a mechanism for receiving message. An example of message about user's access is as follow:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE IDMEF-Message PUBLIC
   "-//IETF//DTD IDMEF v0.1//EN"
   "idmef-message.dtd">
<IDMEF-Message version="0.1">
  <Alert alertid="101" impact="attempted-user">
    <Time offset="0">
      <date>2004/10/09</date>
      <time>10:08:07</time>
    </Time>
    <Analyzer ident="800">
      <Node category="wfw">
        <name>myserver.eda</name>
      </Node>
    </Analyzer>
    <Source>
      <Node>
        <Address category="ipv4-addr">
          <address>192.168.1.90</address>
        </Address>
      </Node>
    </Source>
  </Alert>
</IDMEF-Message>
```

However, in order to improve the scalability of AIDSI, all kinds of IDMEF message is further encapsulated into KQML package by assigning IDMEF as the content part of KQML package.

3.4 Client Agent

The main function of client agent is to collect extra user information, such as Operating System, network card, etc. which is used to improve the detection accuracy.

The structure of client agent is shown in Figure 3. Layer 1 collects extra information by calling operating system API when the user login into the application. Layer 2 translates the message into a special format de-

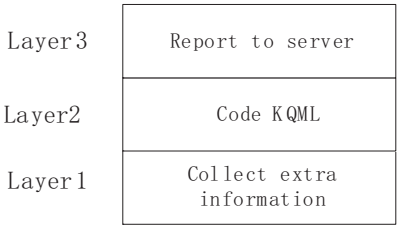

Figure 3: Structure of client agent

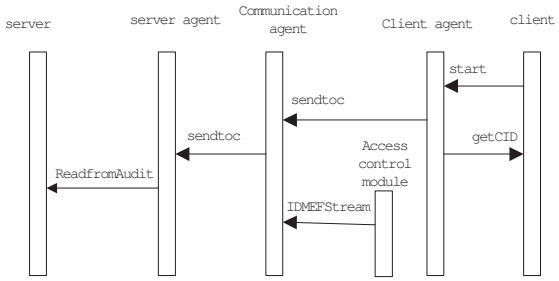

Figure 4: activity diagram of AIDSI and user application system

fined by KQML and Layer 3 simply sends the KQML message to communication agent.

Another important feature of the client agent is reliability in communication. Each client agent is connected to a default communication agent. However, when the communication agent is busy or fails to response, the client agent may select another communication agent from the local lists and then tries to connect to it.

4 Implementation and Performance

4.1 Implementation

The three agents in the AIDSI detection system have been developed in JAVA language, and they should run in JVM (Java Virtual Machine). Because Java is a kind of more flexible and portable language, so AIDSI can run on many kinds of operating systems.

The three agents are composed of several Java classes respectively, and some of the main interface definitions are described in Tables 1-3.

These agents can work together to get a good performance of detection. The activities diagram of the three agents and user application is shown in Figure 4.

First, client agent is called by user application software, and it collects the extra user information and sends it to the communication agent. The communication agent can then send the message to the server agent. The server agent can initialize an instance for the user and set up a user model and compute the three feature values of the user, then make a decision on whether the user is masqueraded or not. There are two methods in AIDSI to get the user action event sequence, that is, from log files or from access control module of user application system.

4.2 Case study

In order to explain how to integrate the AIDSI with a network-based application system and detect masquerade intrusion, we give an example for integrating AIDSI with a network-based application system, W-EDA, (WEB-based Electric Design Automation), which provides a common design platform for many electric design engineers [32]. The W-EDA has three basic components, i.e. Designer,

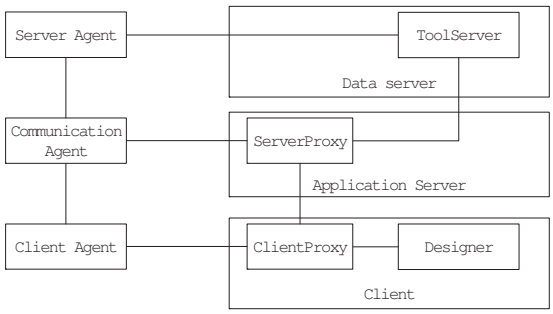

Figure 5: Integration of AIDSI into W-EDA

Application Server and Tool Server. Because, the data in Tool Server is required to keep security, so W-EDA makes use of mainframe-hidden technology so as not to allow any Designers to access it directly. And this is realized by deploying a proxy on the side of Designer and building a proxy in Application Server respectively. And the Designer can visit the Tool Server only through these two proxies.

In order to perform masquerade intrusion detection, we integrate the AIDSI into the design of W-EDA, and the software structure for integration is shown in Figure 5. Three agents can perform task by invoking the public interface from user application.

Now we provide a sample scenario to show that how the masquerade detection is going on. Suppose the user model has been created, then, we can simply follow these steps:

1) the client proxy is started up by user at the Designer, then, the client agent in AIDSI is instanced by the client proxy's calling the interface start().

2) after the server proxy receives the connection requirement from client proxy and the user authentication is successful, it send the user information to the communication agent.

3) extra user feature is collected at Designer side and sent to communication agent by client agent.

4) the communication agent then sends these information to the server agent.

Table 1: Interface description of client agent

Public interface	Function description
public void start();	Start the agent
public String getCID();	Get the ID of network interface card
public sendtoc(String);	Send the ID to the communication agent
public void end();	Close and destroy itself

Table 2: Interface description of communication agent

Public interface	Function description
public void start();	Start the agent, and wait for the request from client agent or access control module
public sendtoc(String);	Send the user information to the server agent
public String IDMEFStream();	Get the user events in IDMEF from access control module
public String encapKQML(String)	Encapsulate the message from access control module into KQML package
public void end();	Close and destroy itself

Table 3: Interface description of server agent

Public interface	Function description
public void start();	Start the agent, and wait for the request from Communication agent
public String parseKQML(KQML)	Parse a KQML package into a string
public String ReadfromAudit(String file)	Read user audit records from a predefined file
public HMM InstanceUserModel(String Userid)	Created a HMM user model by reading from model file
public Boolean detect(double threshold);	Performance detection with a threshold value
public void end();	Close and destroy itself

5) the server agent initializes the user model and continually checks the audit records in ToolServer and encodes them into a sequence in a certain length that can be recognized by HMM.

6) server agent calculates the user's distinguished value and makes decision on whether masquerade intrusion happens or not by comparing the distinguished value with a threshold value.

4.3 Experiments and Analysis

In this section, we do some experiments to show the effectiveness of the AIDSI. First, BSM data is collected by monitoring BSM audit daemon. Then the data set is parsed and encoded into sequence that can be dealt with HMM. We collect 50 users' data and design an experiment. We randomly select two users A and B, suppose user A as masquerade and select several event sequence with a probability and inject the data into user B. Then we perform the experiment for all 50 users and calculate the overall performance, which includes hit rate and false alarm rate. And in the experiment, we vary the length of sequence from 30 to 100, the result is shown in Figure 6. As we can see, as the length of the sequence increases, the performance of our detection algorithm is becoming better and better.

To verify the effectiveness of the automatically selected sliding window size, we perform another experiment, which uses different method of sliding window size selection. The performance is shown in Figure 7, and we can see that the detection performance using adaptive length selection is the best.

To show the impact of user feature selection on detection algorithm, we do another two experiments with different feature selection. The first one just uses the system calling sequence as the user feature; the second one also includes two kinds of extra user information, that is, the identity of client's network card and the user's login time. And the testing result is show in Figure 8. The abscissa in the figure stands for different user, while the ordinate indicates the ROC-1 value [27]. The larger it is, the better the performance is.

As we can see there are 41 users' detection performance is better when using multi-feature. So, the performance

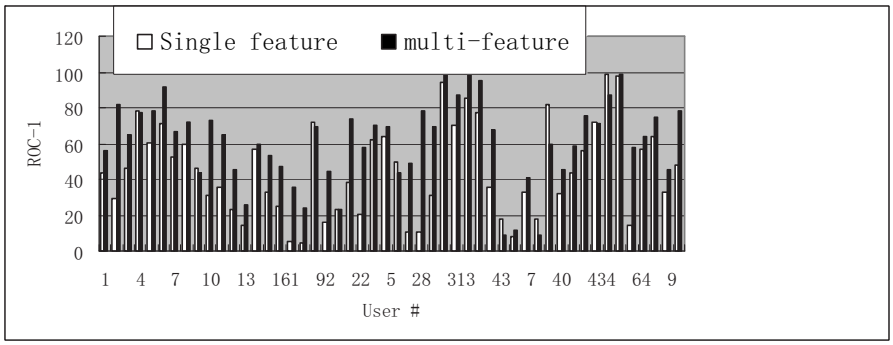

Figure 8: Performance comparison on feature selection

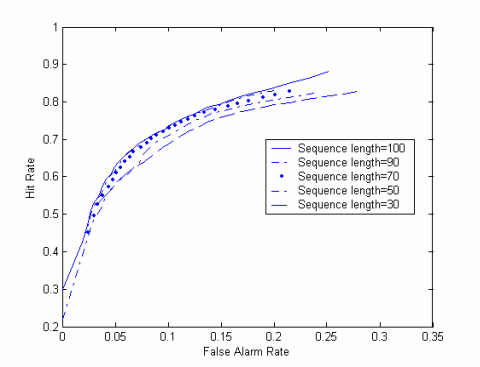

Figure 6: Performance under different sequence length

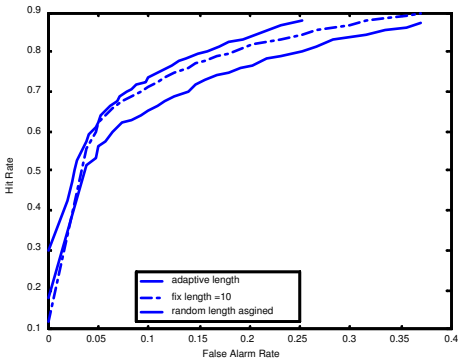

Figure 7: Performance under different sliding window size.

is much better when more user features are included in detection.

Now, we turn to test the impact of AIDSI on user application system. In the experiment, we record the mean memory and CPU usage in server under two circumstances, that is, whether AIDSI is running in the servers or not. The results are shown in table 4. As we can see that the mean memory required and the CPU usage increase when AIDSI is running on the server, however, the increasing value is much lower as the number of users increases. So AIDSI's impact on user application system is minor.

From the above experiment and discussion, we can see that the proposed AIDSI is of following merits.

Ability for integration. The AIDSI is an agent-based intrusion detection system, which is easy to be integrated.

The client agent just acts as an extra information collector, and it need not exchange data with the client of user application. The three agents are written in Java language so that they can be deployed at different operation system. The agent and access control module is loosely coupled because they use standard protocol, such as KQML, IDMEF, in the process of data exchanging. The impact of AIDSI on application system is minor as far as memory and CPU usage are concerned.

Detection performance. In AIDSI, the performance of the proposed detection algorithm can be better if the length of sequence and the size of sliding window are properly selected. Extra user information can effectively improve the performance of the detection algorithm.

5 Conclusion

Application-based intrusion is a kind of more serious intrusion since masqueraders can get the data meaning in a visual mode. AIDSI, which is different from other IDS, is

Table 4: impact on user application system

Number of users	5	10	15	20	25	30
Memory required when AIDSI is running (K)	208	331	402	471	525	601
Memory required when AIDSI is not running (K)	205	324	390	453	502	572
CPU Usage when AIDSI is running (%)	44	53	57	67	73	80
CPU Usage when AIDSI is not running (%)	40	48	55	63	69	75

proposed to deal with such kind of intrusion. Agent technology is introduced into the system design. Server agent detects the masquerade intrusion using HMM model. Client agent gets extra user information for detection algorithm to improve the detection accuracy. AIDSI is more flexible and can be integrated with enterprise information systems well.

Also in this paper, we present an example of integration AIDSI with a WEB-based application system W-EDA to detect an application-based intrusion. And experiment shows that AIDSI can work well with W-EDA and get a satisfactory performance. In fact, for an application system, if its logical structure is a three tiers type like Figure 1, the AIDSI can be applicable.

Although it is effective in some kind of network-based application systems, there may be still some disadvantages in AIDSI, which will enable us to research further. Server agent is the center of AIDSI and its impact on user application may be notable when the user number increases. So, how to distribute the user model on different servers and perform a cooperative detection is a main task in future. On the other hand, mechanism of response when the server agent detects an intrusion should be designed to improve the performance of AIDSI.

Acknowledgements

This paper is supported by the funding of Fujian province of China NSFC Project No. A0410007 and Corporation Start-up Project. We would like to thank the anonymous reviewers for their comments, which is helpful to improve the quality of our paper.

References

[1] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," *14th Annual Conference on Computer Security Applications*, pp. 13-24, 1998.

[2] A. Berqia and G. Nacsimento, "A distributed approach for intrusion detection systems," *International Conference on Information and Communication Technologies: From Theory to Applications*, pp. 493-494, 2004.

[3] J. M. Bradshaw, *Software Agents*, AAAI Press/The MIT Press, 1997.

[4] S. Coull, J. Branch, B. Szymanski, and E. Breimer, "Intrusion detection: a bioinformatics approach," *19th Annual Conference on Computer Security Applications*, pp. 24-33, 2003.

[5] D. Curry, H. Debar, and M. Huang, *IDMEF Data Model and XML DTD*, http://www.oasis-open.org/cover/IDMEF-provisional-draft-ietf-idwg-idmef-xml-02.html, 2000.

[6] J. E. Dickerson, J. Juslin, O. Koukousoula, and J. A. Dickerson, "Fuzzy intrusion detection," *Joint 9th IFSA World Congress*, pp. 1506-1510, 2001.

[7] L. P. Gaspary, E. Meneghetti, and L. R. Tarouco, "An SNMP agent for stateful intrusion detection inspection," *Integrated Network Management*, pp. 3-16, 2003.

[8] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A network security monitor," *IEEE Symposium on Research in Security and Privacy*, pp. 296-304, 1990.

[9] I. M. Hegazy, T. Al-Arif, Z. T. Fayed, and H. M. Faheem, "A multi-agent based system for intrusion detection," *Potentials, IEEE*, vol. 22 , no. 4, pp. 28-31, 2003.

[10] J. Hochberg, K. Jackson, C. Stallings, J. F. McClary, D. DuBois, and J. Ford, "NADIR: An automated system for detecting network intrusion and misuse," *Computers and Security*, vol. 12, no. 3, pp. 235-248, 1993.

[11] V. Kasarekar, and B. Ramamurthy, "Distributed hybrid agent based intrusion detection and real time response system," *First International Conference on Broadband Networks*, pp. 739-741, 2004.

[12] Y. Liu, C. F. Xu, W. D. Chen, and Y. H. Pan, "KQML realization algorithms for agent communication," *5th World Congress on Intelligent Control and Automation*, pp. 2376-2379, 2004.

[13] P. Loshin, *Intrusion detection, Computerworld*, http://www.computerworld.com/hardwaretopics/hardware/story/0,10801,59611,00.html, 2001.

[14] Y. Maruyama and K. Yamanishi, "Dynamic model selection with its applications to computer security," *IEEE Conference Information Theory Workshop*, pp. 82-87, 2004.

[15] R. A. Maxion and T. N. Townsend, "Masquerade detection augmented with error analysis," *IEEE Transactions on Reliability*, vol. 53, no. 1, pp. 124-147, 2004.

[16] L. I. Millett and S. H. Holden, "Authentication and its privacy effects," *IEEE Internet Computing*, vol. 7, no. 6, pp. 54-58, 2003.

[17] B. Mukherjee, T. L. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no.3, pp. 26-41,1994.

[18] M. Oka, Y. Oyama, H. Abe, and K. Kato, "Anomaly detection using layered networks based on eigen co-occurrence matrix," *Proceedings of Seventh International Symposium on Recent Advances in Intrusion Detection (RAID)*, LNCS 3224, pp. 223-237, 2004.

[19] T. Okamoto, T. Watanabe, and Y. Ishida, *Mechanism for generating immunity-based agents that detect masqueraders*, *LNAI 3214*, pp. 534-540, 2004.

[20] D. Pi, Q. Wang, W. Li, and J. Lv, *APA: An interior-oriented intrusion detection system based on multi-agents*, *LNCS 3619*, pp. 1227-1233, 2005.

[21] J. Pikoulas, W. Buchanan, M. Mannion, and K. Triantafyllopoulos, "An intelligent agent security intrusion system," *IEEE International Conference and Workshop on Ninth Annual the Engineering of Computer-Based Systems*, pp. 94-99, 2002.

[22] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[23] A. J. Rocke and R. F. DeMara, *CONFIDANT: Collaborative object notification framework for insider defense using autonomous network transactions*, *Autonomous Agents and Multi-Agent Systems*, http://netmoc.cpe.ucf.edu:8080/internal/conversion /2004/CONFIDANT_framework.pdf .2005.

[24] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," *Statistical Science*, vol. 16, no. 1, pp. 58-74, 2001.

[25] J. Sun, *BSM Security Auditing for Solaris Servers, Ver. 1.4 (option 1)*, http://www.sans.org/rr/ whitepapers/solaris/1098.php, 2003.

[26] University of Ottawa, Intelligent mobile agents research, 2000. http://deneb.genie.uottawa.ca/ web-data/research/

[27] K. Wang and S. J. Stolfo, "One-class training for masquerade detection," *3rd IEEE Conference Data Mining and Workshop on Data Mining for Computer Security*, pp. 1-10, 2003.

[28] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: alternative data models," *IEEE Symposium on Security and Privacy*, pp. 133-145, 1999.

[29] Q. Xue, J. Sun, and Z. Wei, "TJIDS: an intrusion detection architecture for distributed network," *IEEE CCECE*, pp. 709-712, 2003.

[30] Q. Yan, W. X. Xie, G. Song, J. P. Yu, *System Call Anomaly Detection Method Based on HMM*, *ACTA Electronica Sinica (in Chinese)*, vol. 31, no. 10, pp. 1486-1490, 2003.

[31] K. H. Yung, *Using Self-Consistent Naive-Bayes to Detect Masquerades*, PAKDD, pp. 329-340, 2004.

[32] J. P. Zeng and D. H. Guo, "A prototype of Web-based middleware system for EDA tools sharing," *8th International Conference on Computer Supported Cooperative Work in Design*, pp. 26-28, 2004.

[33] J. P. Zeng and D. H. Guo, *A new clustering algorithm for time series analysis*, *International Conference on Intelligent Computing (ICIC 2006)*, pp. 759-764, 2006.

[34] J. P. Zeng and D. H. Guo, *Method for masquerade intrusion detection based on HMM and genetic algorithm*, *Mini-Micro Systems (in Chinese)*, vol. 28, no. 7, pp. 1200-1215, 2007.

Jianping Zeng, PhD. He got a PhD degree in Xiamen University, China in 2006, and now he works as a post-doctoral researcher in Fudan University in China. His research mainly focuses on Artificial Intelligence, Information Security and Software Engineering. E-Mail: zeng_jian_ping@hotmail.com.

Donghui Guo, BSc, MSc, PhD, MIEEE. He joined Xiamen University in 1994 and became as a full professor in 2001. He had been to City University in Hong Kong, University of Ulster in UK, UC Berkeley in USA as visiting researcher or visiting professor for several years. He is now the head of circuit and system institute in Xiamen University and doing researches on Telecommunication, IC design and Artificial Intelligence.