IOSR Journal of Engineering
Apr. 2012, Vol. 2(4) pp: 915-922

Implementation of Fault Tolerant Architecture in Decentralized Intrusion Detection System.

Surbhi Chauhan[1],Kamal Kant[2],Abhay Bansal[3] and Arjun Singh[4]

[1] *Department of CSE, Amity University, Noida, INDIA,*
[2]*Department of CSE, Amity University, Noida, INDIA,*
[3]*Department of CSE, Amity University, Noida, INDIA,*

[4] *Department of CSE, Sir Padampat Singhania University, INDIA,*

ABSTRACT

The aim of this paper is to detect anomalous usage of legitimate applications by authorized users in Windows environment and to implement a fault – tolerant architecture which can continue providing detection service even in case of failure of one or more detecting servers. This paper also aims to implementing mobile agent technology for gathering the information from various monitored hosts for a period of every 10 seconds. And to build per – application based profile for authorized users. This paper implements the architecture that continues providing detection services even in case of busy state or failure of one more detecting server.

[I]. INTRODUCTION

Most of research works in intrusion detection were implemented in UNIX – based systems, where the data source is just the user's command line. Such data has the advantage of being read by user. Moreover, open – sourced environment provides less complexity while implementation. But in today's point and click environment this type of data is increasingly rare. Jude Shavlik and Mark Shavlik [9] made the first attempt and proposed anomaly – based intrusion detection system that created statistical profiles of the usage for a given computer running on Windows 2000.The most common shortcoming in typical IDS is that they were built around a monolithic architecture, where data gathering, processing and reporting were built as a single entity at each host in the network. Later, in centralized approach, the data processing and reporting components alone were isolated and built as a single entity at a dedicated detecting server. In existing hierarchical architectures there were two or more detecting servers dedicated for each segment in the network. In either of the architectures, failure of a server leaves part of the network unprotected.
Moreover, such architectures degrade the performance of the detecting system when the network scales. Earlier work on a fault – tolerant architecture had the detecting component embedded on static agents at each host and exchanged traces of intrusions using mobile agents . A static agent on receiving intrusion traces from its predecessor host, would learn the intrusion patterns at other system.Previous profiling methods targeted either the user or the system profiling. In user profiling [5], [18], the detecting system periodically or continuously monitored the behavior of a user at various monitored hosts. Modeling the activity would represent the user's normal behavior. The model would provide a form of authentication that would be very hard to impersonate. It should come as no surprise that this turns out to be a highly non trivial problem, because how to model human behavior is far from obvious. Even more difficult is to try and determine whether a legitimate user is doing anything malicious. Whereas, in system profiling, the detecting system continuously captured system parameters of each monitored hosts and modeled the normal behavior of those systems [9]. Model developed represents the system normal behavior. But, gathering system parameters continuously then performing detection operation would bring down the performance of various other processes in that host, and hence the system. A related discipline is Program Profiling e.g. as in [4], in which the normal behavior for an application program is modeled, usually for the purpose of detecting whether the program is doing anything it was not designed to do. In this paper it was believed that program profiling would somewhat be an easier problem because unlike humans, programs come with specifications and their behavior would therefore be very limited by comparison.

[II]. OVERVIEW OF FAULT – TOLERANT ANOMALY DETECTION SYSTEM

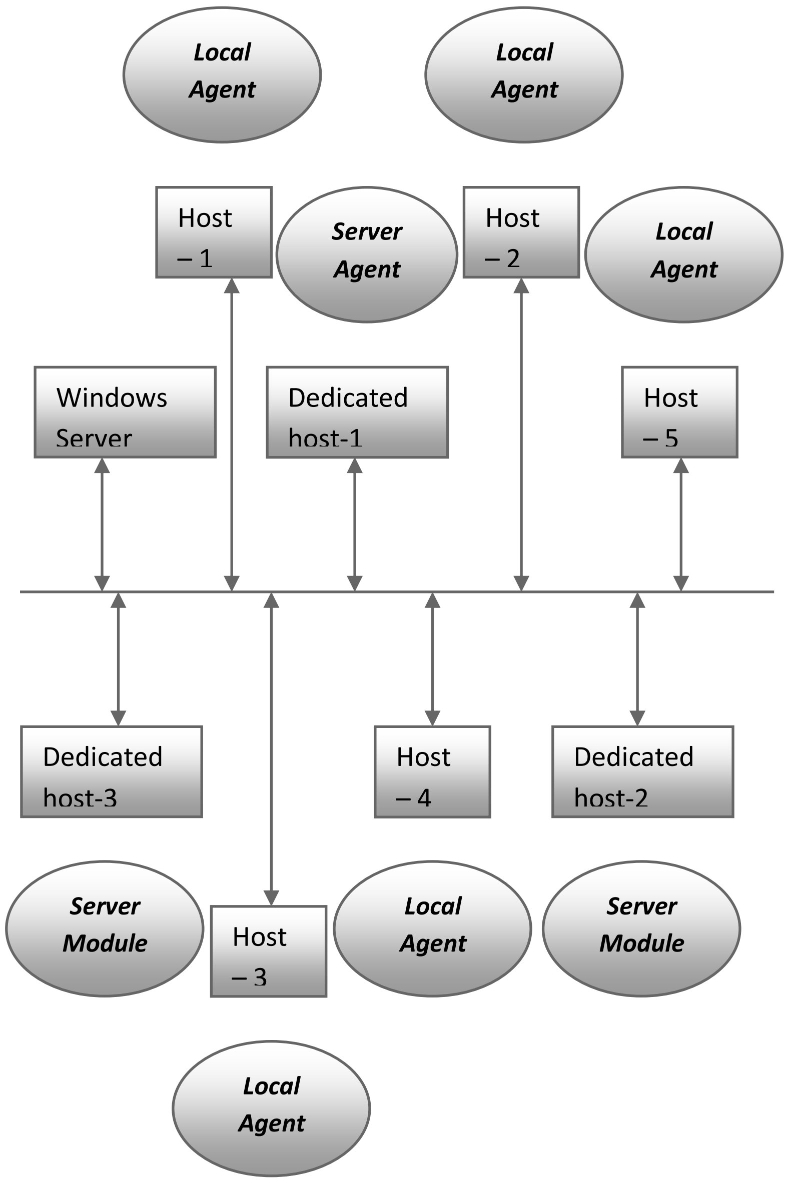


Figure 2.1: Overall architecture of Fault-tolerant Anomaly Detection System.

This paper presents mobile agent based de – centralized and fault – tolerant IDS to detect user anomalies in Windows environment that address some of the issues with the existing IDS models as mentioned in the following sessions. The major components of any intrusion detecting system are data gathering, data processing and report generation. Jude and Mark proposed the first IDS for Windows NT, adopted a monolithic architecture [9]. In the architecture, all the three components were considered as a single entity. In this paper, the data processing and report generation components were built together as a single entity, Server Module. Several such entities were deployed in more than one dedicated system across the network and were called the detecting server or detecting system. The data gathering component deployed as mobile agent, travel from the detecting system to perform specified task at each monitored host. The local agents were deployed in every host in the network. The core responsibility of this local (static) agent is only to receive and execute the mobile agent locally.

[III] OUR WORK

The proposed Fault Tolerant Architecture is being simulated in a network environment . The software for each type of module was written in Sun Java JDK Version1.5 on a Microsoft Windows environment. The Simulation of the Fault Tolerant architecture involves the following

- Information Source
- Analysis Engine
- Report Manager

The experiment was accomplished at an isolated network with Five hosts and one Windows server. The detecting and reporting modules together were installed in three dedicated hosts (detecting servers) Alpha, Beta and Gamma. The local agent was installed in all the other hosts. The IP addresses of the detecting servers were then updated to all local agents at every monitored host. The local agents maintained this information at a random order. When a user logged in at any host, after the traditional authentication procedures, the local agent at that host send request to the detecting server whose IP address was at the first in its list. The proposed system was deployed in the network for five days and its services were tested against activities of Two volunteers. After the initial configurations, the first five sessions, from login to logout of each user was considered as the training phase for that user.

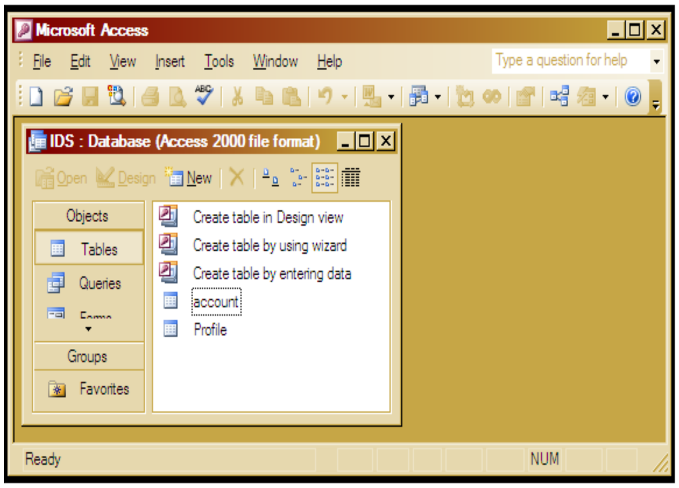
During the training phase, the user' activities at a host was monitored periodically using mobile agents, which were dispatched periodically from the detecting server. Program profile was built to every user based on the information collected during their first five login session. This profile of every user carried the threshold values of parameters which determine when an alarm should be generated.

During the testing phase, the detecting servers have with it the profile of every user who had then actively logged in. The mobile agents periodically visited the active hosts and reported the activities of users at various hosts. The agent, for every 10 seconds, collected information such as the : User name, Host IP address, Time, Name of the applications that were then currently running at that host, Number of simultaneous but same applications, Time since the applications were activated and Active time spent on each application. The last four parameters were instantaneously compared with the corresponding

thresholds at the profile of that user. If there occurred any deviation, that is, if currently received values exceed the threshold values then intrusion alert messages were displayed at the detecting server The scope of threshold was limited only for the next login session. Latter, the thresholds were built using the history of information collected during recent five sessions. The 'recent-past' collected information helped in 'dynamic profiling'.
Winnow-based algorithm is used to build application based profile for each user, Program profiling(subset of system profile). It is a machine learning algorithm proposed by Jude and Mark to build system profile. It was stated that the parameter of each application was tested against the profile using Winnow Based Algorithm . For each violation a variable called *WeightFor* was incremented and each normal activity a variable called *WeightAgainst* was incremented. An alert message was to be generated only if the value of *WeightFor* is greater than *WeightAgainst*. And, after 10 seconds, these variables would again set to zero. But while implementation, this system did not use these variable. Because, if a user launch an intrusion along with few normal behaviors then the system fail to detect such anomalous. Hence, in this system, even if one parameter exceeds the threshold once, then alert messages were displayed. FADS (Fault Anomaly Detection System) was also successful in supporting the fault – tolerant architecture. The detecting server Alpha, which was providing detection service to two users, was purposefully switched down. The hosts that were initially receiving services from server Alpha, at random, sent request to servers Beta and Gamma. The results of threshold violation and server failure were discussed in next Section.

[IV] SAMPLES OF EXPERIMENTATION

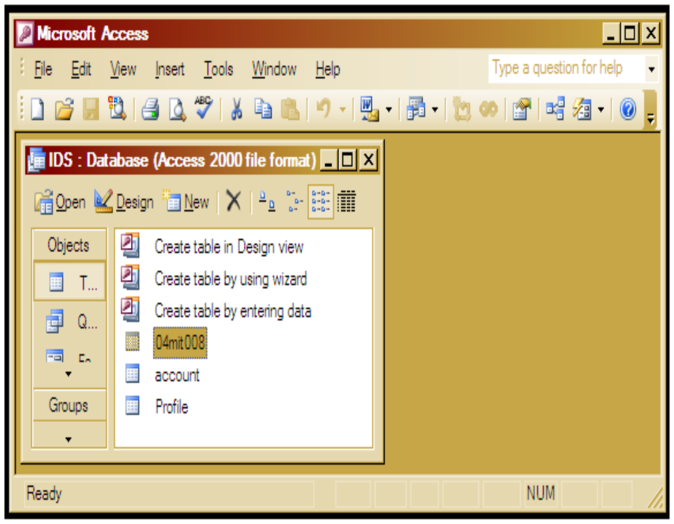
For documentation purpose, behavior of one user '04mit008' alone was recorded. The system functionalities and results were discussed based on the information collected while monitoring this user during his various logins at various hosts in the network. The figure 4.1 shows the initial status of the database with two tables: 'account' and 'profile'. The table 'account' maintains the login details of the users and the 'profile' maintains the thresholds for various users during various login sessions. Once after the user '04mit008' proves authenticity at machine: 192.168.1.10, the local agent at that system sends request to the detecting server ALPHA. The detecting system verified the user name in the 'account' table.



Create table in Design view
Create table by using wizard
Create table by entering data
account
Profile

Fig 4.1 : Initial Status of Database

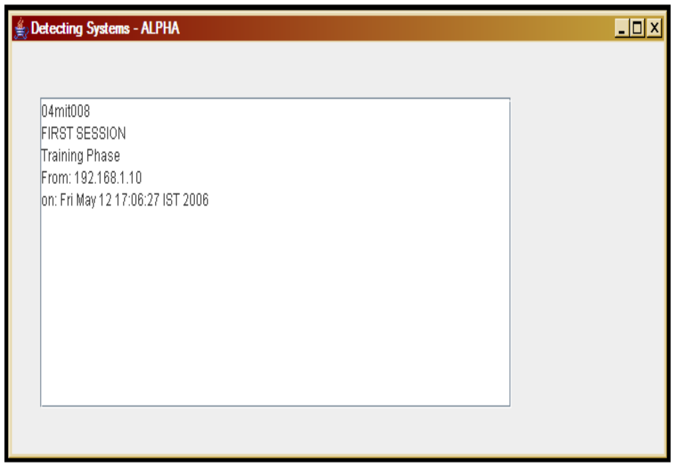
Since the user logged in for the first time, a table was created for that user and is shown in figure 4.2.



Create table in Design view
Create table by using wizard
Create table by entering data
04mit008
account
Profile

Fig 4.2: Table Created for New User

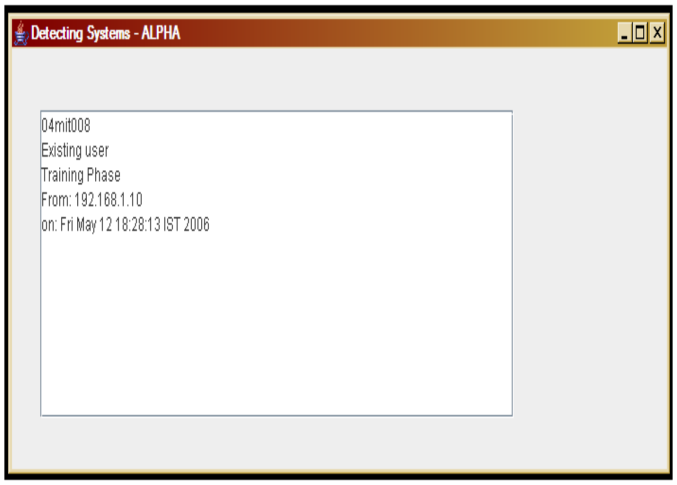
The Figure 4.3 shows a window at the detecting system ALPHA alerting the administrator that a new user has logged in for the first time from 192.168.1.10 with few other details . The first few login sessions were considered as training phase and hence no detection will be done during these sessions. For experimental purpose, First 5 sessions were considered as training period and it was assumed that the user would use only legitimate applications during these training periods.



04mit008
FIRST SESSION
Training Phase
From: 192.168.1.10
on: Fri May 12 17:06:27 IST 2006

Figure 4.3: Alert message when a new user logged.

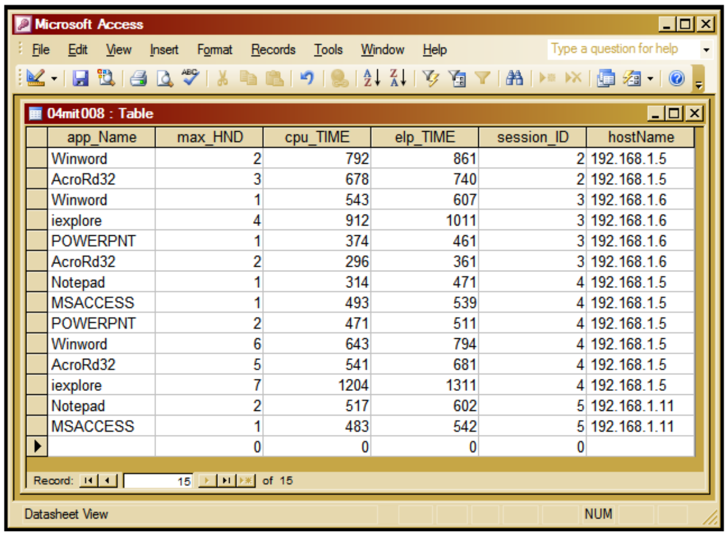
The figure 4.4 shows a window which displays each time when an existing user logged in through of the monitored hosts.



04mit008
Existing user
Training Phase
From: 192.168.1.10
on: Fri May 12 18:28:13 IST 2006

Figure 4.4: Alert message when an existing user logged.

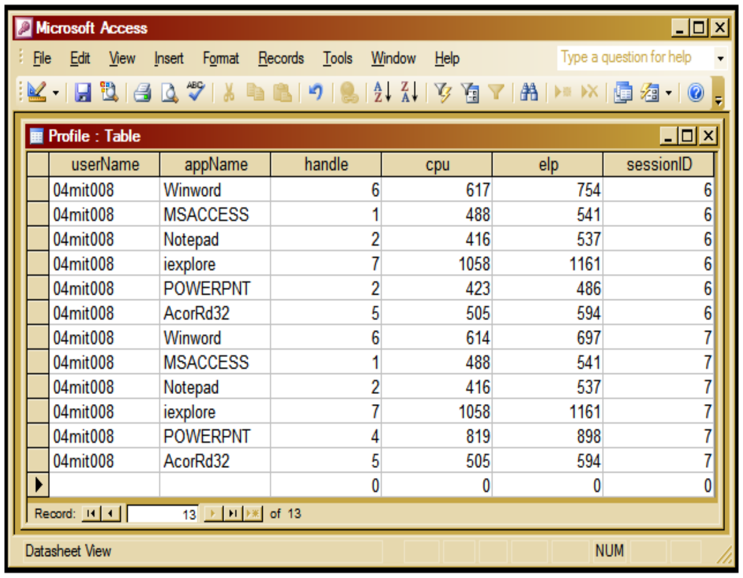
The user activities collected during four different login were recorded is shown in Figure 4.5. In this figure, the column *app_Name* has the list of name of applications that the user 04mit008 used during various logins. The *max_HND* is a list of maximum number of similar simultaneous applications used during that session. The column *cpu_TIME* and *elp_TIME* are lists of active time in seconds spent on each application during that session and time in seconds since that application was opened. The *session_ID* and *hostname* are the session information.



app_Name	max_HND	cpu_TIME	elp_TIME	session_ID	hostName
Winword	2	792	861	2	192.168.1.5
AcroRd32	3	678	740	2	192.168.1.5
Winword	1	543	607	3	192.168.1.6
iexplore	4	912	1011	3	192.168.1.6
POWERPNT	1	374	461	3	192.168.1.6
AcroRd32	2	296	361	3	192.168.1.6
Notepad	1	314	471	4	192.168.1.5
MSACCESS	1	493	539	4	192.168.1.5
POWERPNT	2	471	511	4	192.168.1.5
Winword	6	643	794	4	192.168.1.5
AcroRd32	5	541	681	4	192.168.1.5
iexplore	7	1204	1311	4	192.168.1.5
Notepad	2	517	602	5	192.168.1.11
MSACCESS	1	483	542	5	192.168.1.11
	0	0	0	0	

Figure 4.5: Parameters gathered during end of session 5

The *userName* is the name of the user from whom the per – application based profile is built. The *appName* is the name of the application for which the threshold was computed for the next session.
Figure 4.5 shows the information collected during the first five sessions. The *handle* is a list of maximum number of similar simultaneous applications used during the recent past. The *CPU* and *elp* are lists of average of active time in seconds spent on each application during the recent past and average of time in seconds since that application was opened. The figures 4.6 show the user activities during his sixth login sessions and profile built using the recent past information. That is, while building the profile for sixth session the system considered the information collected during the recent five sessions .



userName	appName	handle	cpu	elp	sessionID
04mit008	Winword	6	617	754	6
04mit008	MSACCESS	1	488	541	6
04mit008	Notepad	2	416	537	6
04mit008	iexplore	7	1058	1161	6
04mit008	POWERPNT	2	423	486	6
04mit008	AcorRd32	5	505	594	6
04mit008	Winword	6	614	697	7
04mit008	MSACCESS	1	488	541	7
04mit008	Notepad	2	416	537	7
04mit008	iexplore	7	1058	1161	7
04mit008	POWERPNT	4	819	898	7
04mit008	AcorRd32	5	505	594	7
		0	0	0	0

Figure 4.6: Thresholds for session 6.

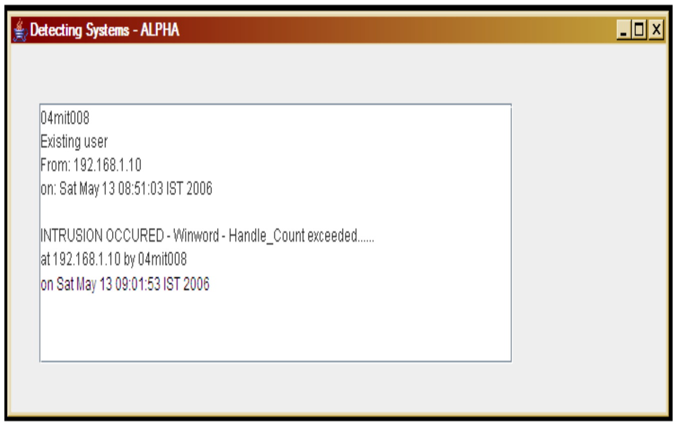
The table 4.7 shows an overview of various activities which the proposed system considered as anomalous behavior. It also explained why the activity was considered as intrusion and the nature of intrusion.

Table 4.7: Varieties of Anomalies considered.

TYPE OF ANOMALY	EXPLANATION	NATURE OF INTRUSION
Exceeding Handle Count	Number of similar simultaneous application	Abnormal behavior Misusing Resources
Exceeding CPU Time	Effective time spent on that application	Abnormal behavior
Exceeding Elapsed Time	An application was invoked but was not used	Misusing Resources
Attempting to use New Application	Application that was not used during past	Abnormal behavior

5. RESULTS AND DISCUSSIONS

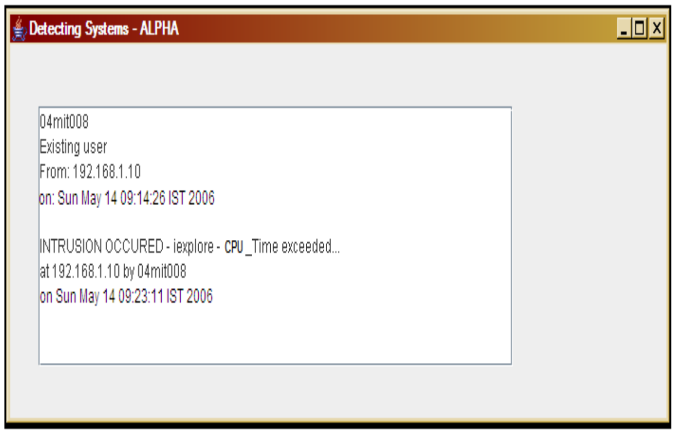
The proposed system also supported fault – tolerant architecture, where hosts in one segment can receive detection services from other detecting servers in same or different segment. The Figures 5.1 and 5.2 were alert messages to administrator at the detecting system ALPHA during different sessions of the same user from different host. The Figure 5.1 notifies that the user '*04mit008*' attempted to use more number of *MSWord* applications than his usual behavior.



```
04mit008
Existing user
From: 192.168.1.10
on: Sat May 13 08:51:03 IST 2006

INTRUSION OCCURED - Winword - Handle_Count exceeded......
at 192.168.1.10 by 04mit008
on Sat May 13 09:01:53 IST 2006
```

Figure 5.1: Alert message for Handle violation

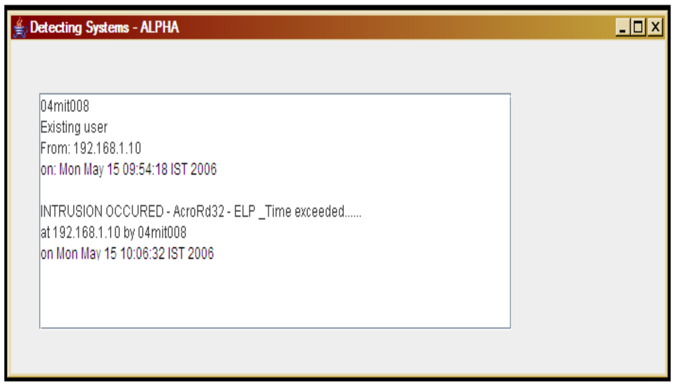


```
04mit008
Existing user
From: 192.168.1.10
on: Sun May 14 09:14:26 IST 2006

INTRUSION OCCURED - iexplore - CPU _Time exceeded...
at 192.168.1.10 by 04mit008
on Sun May 14 09:23:11 IST 2006
```

Figure 5.2: Alert message for CPU time violation

The Figure 5.2 was alert generated when the same user tried working on *iexplore* more than profiled value. The screenshot also carries with it the system ' IP address from where the intrusion was launched and the date and time of intrusion. The Figures 5.3 and 5.4 were alerts generated during a particular session of the same user. The figure 5.3 was an alert generated when the user attempted to use *Acrobat Reader* for a longer period than the profiled value.

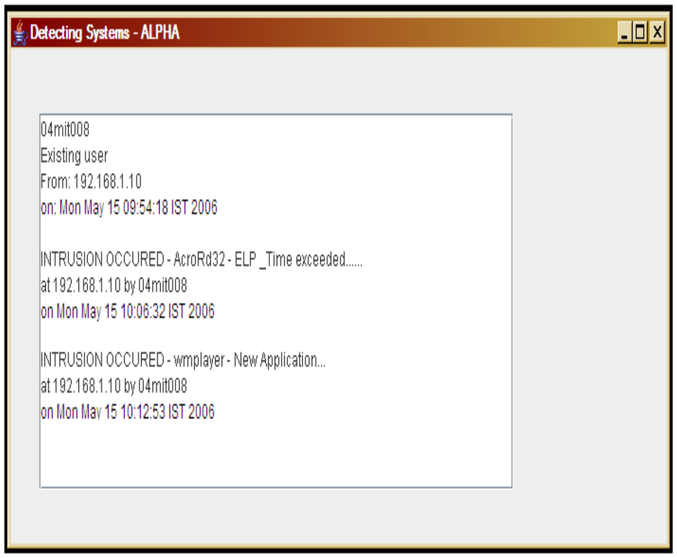


```
04mit008
Existing user
From: 192.168.1.10
on: Mon May 15 09:54:18 IST 2006

INTRUSION OCCURED - AcroRd32 - ELP _Time exceeded......
at 192.168.1.10 by 04mit008
on Mon May 15 10:06:32 IST 2006
```

Figure 5.3: Alert message for Elapsed time violation

Approximately 6 minutes after the intrusion, the user attempted to use *Windows Media Player*, which he had not used earlier, shown in figure 5.4.



Detecting Systems - ALPHA

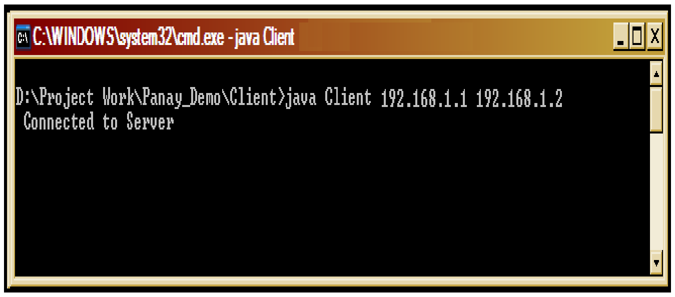
```
04mit008
Existing user
From: 192.168.1.10
on: Mon May 15 09:54:18 IST 2006

INTRUSION OCCURED - AcroRd32 - ELP _Time exceeded......
at 192.168.1.10 by 04mit008
on Mon May 15 10:06:32 IST 2006

INTRUSION OCCURED - wmplayer - New Application...
at 192.168.1.10 by 04mit008
on Mon May 15 10:12:53 IST 2006
```

Figure 5.4: Alert message for attempting to use New application

The figure 5.5 shows the initial configuration of the local agent (192.168.1.10) with a list of IP addresses of two detecting servers ALPHA (192.168.1.2) and BETA (192.168.1.2). The local agent initially receives detecting services from ALPHA.

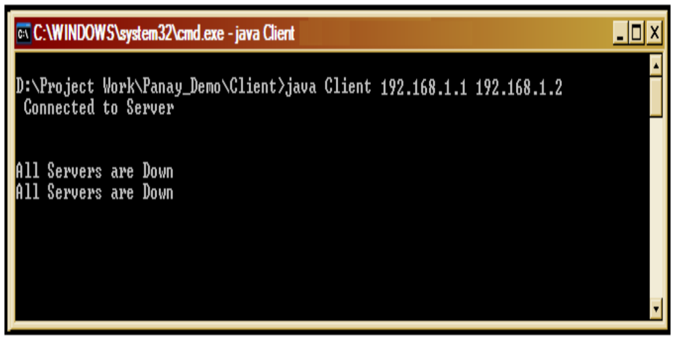


C:\WINDOWS\system32\cmd.exe - java Client

```
D:\Project Work\Panay_Demo\Client>java Client 192.168.1.1 192.168.1.2
 Connected to Server
```

Figure 5.5: Host configuration window

The figure 5.6 is message at the client when all the detecting servers become inactive.



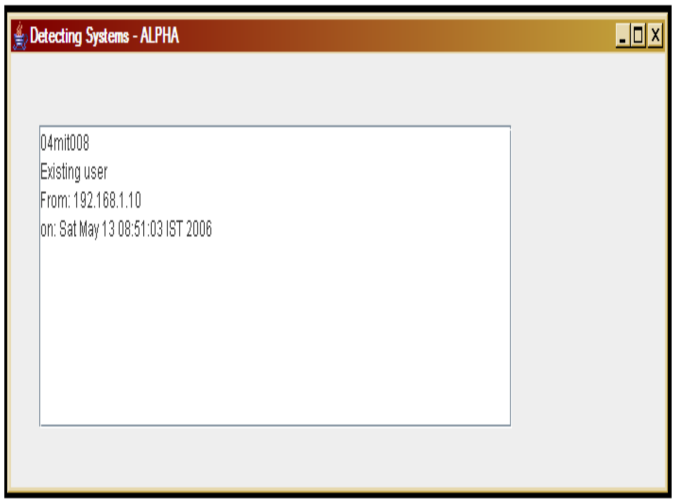
C:\WINDOWS\system32\cmd.exe - java Client

```
D:\Project Work\Panay_Demo\Client>java Client 192.168.1.1 192.168.1.2
 Connected to Server


All Servers are Down
All Servers are Down
```

Figure 5.6: Servers failure message at Host

The figure 5.7 and 5.8 were alert message given to the administrator, stating that a new user has logged in, with respect to that server.

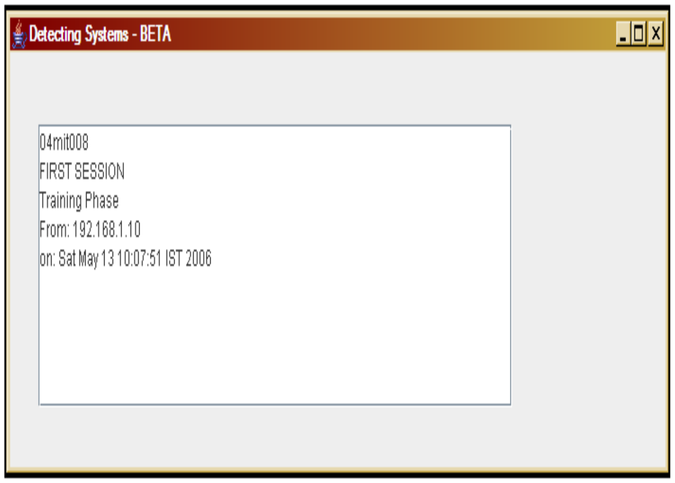


Detecting Systems - ALPHA

```
04mit008
Existing user
From: 192.168.1.10
on: Sat May 13 08:51:03 IST 2006
```

Figure 5.7: Request accepted at server ALPHA

The Figure 5.8 was observed when the local agent at client 192.168.1.10 requested service to server BETA, because of the failure of the server ALPHA.



Detecting Systems - BETA

```
04mit008
FIRST SESSION
Training Phase
From: 192.168.1.10
on: Sat May 13 10:07:51 IST 2006
```

Figure 5.8: Request accepted at server BETA

The table 5.9 is a summary of results observed during different login sessions of different users in different host at different time. The detection rate is defined as the ratio of the number of intrusions detected to the number of intrusions launched. The overall detection rate of the system is: 95.83%

Table 5.9: Detection rate for various anomalies

Type of intrusion	No. of intrusion detected	No. of intrusion launched	Detection rate
Handle count exceeded	42	45	93.33%
CPU time exceeded	34	35	97.14%
Elapsed time exceeded	24	25	96%
New application	10	10	100%

CONCLUSIONS

In this "Modeling Intrusion Detection" a fault-tolerant anomaly detecting system was proposed to identify anomalous usage of legitimate applications by authorized users in Windows environment. Mobile agents were used to collect three application related parameters that were then currently running in the kernel at various hosts. The gathered information was periodically reported to the detection server. A machine learning approach called Winnow-based algorithm was used to learn and built per – application based *program profile* for each authorized user. Latter, irrespective to the host, any application accessed by that users and its related properties were periodically gathered. Such gathered information was simultaneously compared with users' program profile. Any deviations were considered as anomalous activities and were reported. From the experimental evaluation, the information collected during the recent five sessions, helped in 'dynamic profiling'. Because of dynamic profiling the system was able to tune itself with recent behaviors of users. The fault – tolerant architecture, wherein a single point failure will not leave the network unprotected, was successfully implemented. And, a detection rate of 95.83% was achieved.

REFERENCES

[1] Adriano M. Cansian, Artur R. A. da Silva and Marcelo de Souza, **"An Attack Signature Model to Computer Security Intrusion Detection"**, *IEEE*, pp: 1368-1373, 2002.

[2] Balasubramaniyan. J, J. O. G.Fernandez, D. Isacoff, E., H. Spafford, and D. Zamboni, **" An Architecture for Intrusion Detection using Autonomous Agents"**, *Technical report no. TR 98-05*, Purdue University, USA, 1998.

[3] Bernardes, M.C and dos Santos Moreira, E., **"Implementation of an intrusion detection system based on mobile agents"**, *Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems*, pp. 158-164 June 2000.

[4] Chan. P. C and Victor K. Wei, **"Preemptive Distributed Intrusion Detection using Mobile Agents"**, *Proceedings of the Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE'02), 2002.

[5] Debapriyay Mukhopadhyay and Satyajit Banerjee, **"User Profiling for Host Based Anomaly Intrusion Detection in Windows NT"**, *Emerging Applications Information Technology*, pp.193-196, 2006.

[6] Guy Helmer, Johnny S.K . Wong, Vasant Honavar and Les Miller, **"Automated discovery of concise predictive rules for intrusion detection"**, *The Journal of Systems and Software*, vol. 60, pp: 165–175, 2002.

[7] Guy Helmer , Johnny S.K. Wong, Vasant Honavar, Les Miller and Yanxin Wang, **"Lightweight agents for intrusion detection"**, *The Journal of Systems and Software*, vol. 67, pp: 109-122, 2003.

[8] Gorodetski. V, and Kotenko, **"The Multi-agent Systems for Computer Network Security Assurance: Frameworks and Case Studies"**, *Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems* (ICAIS'02), 2002.

[9] Jude Shavlik and Mark Shavlik, **"Selection, Combination and Evaluation of Effective Software Sensor for Detecting Abnormal Computer Usage"**, *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining*, pp. 276-285, 2004.

[10] Kymie M. C. Tan and Roy A. Maxion, **" Determining the Operational Limits of an Anomaly-Based Intrusion Detector"**, IEEE Journal on Selected Areas In Communications, vol. 21, 2003

[11]. Ozgur Depren, Murat Topallar, Emin Anarim, M. Kemal Ciliz, "**An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks"**, *Expert Systems with Applications*, vol. 29, pp: 713–722, 2005.

[12] Pradeep Kannadiga, M. Zulkernine, and S. Ahamed, **"Towards an Intrusion Detection System for Pervasive Computing Environments"**, *Proceedings of the International Conference on Information Technology* (ITCC), Las Vegas, Nevada, USA, April 2005.

[13] Pradeep Kannadiga and Mohammad Zulkernine, **"DIDMA: A Distributed Intrusion Detection System Using Mobile Agents"**, *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International*

Workshop on Self-Assembling Wireless Networks, 2005.

[14] Peng Ning, Sushil Jajodia, Xiaoyang Sean Wang, **"Design and Implementation of a decentralized prototype system for detecting distributed attacks",** *IEEE Transactions on Computer Communications*, vol. 25. pp: 1374-1391, 2002.

[15] Jing Xu,Yongzhong Li " A New Distributed Intrusion Detection Model Based on immune Mobile Agent Proceedings of IEEE ,2009

[16] Tao Peng , Christopher Leckie and Kotagiri Ramamohanarao , **"Information Sharing for Distributed Intrusion Detection Systems",** *Journal of Network and Computer Applications,* 2005.

[17] Mo Xiu-liang, Wang Chun-dong ,Wang Huai-bin " A Distributed Intrusion Detection System Based on Mobile Agents"Proceeding in IEEE ,2009

[18] Yoshinori Okazaki and Izuru Sato, **"A New Intrusion Detection Method based on Process Profiling",** *Proceedings of the IEEE Symposium on Applications and the Internet,* 2002.

[19] Liu Jianxiao 1, Li Lijuan 1 "Research of Distributed Intrusion Detection System Model Based on Mobile Agent" Proceedings of IEEE International Forum on Information Technology and Application,2009

[20] MO Xiu-liang, WANG Chun-dong , WANG Huai-bin"A Distributed Intrusion Detection System Based on Mobile Agents" Proceedings of IEEE,2009

[21] Nita Patil, Chhaya Das, Shreya Patankar,Kshitija Pol "Analysis of Distributed Intrusion Detection Systems using Mobile Agents" Proceedings of IEEE First International Conference on Emerging Trends in Engineering and Technology ,2008

[22] Saidat Adebukola Onashoga , Adebayo D. Akinde, Adesina Simon Sodiya " A Strategic Review of Existing Mobile Agent-Based Intrusion Detection Systems" Proceeding of Issues in Informing Science and Information Technology Volume 6, 2009.