An Experience Improving Intrusion Detection Systems False Alarm Ratio by Using Honeypot

Babak Khosravifar, Jamal Bentahar
Department of Electrical and Computer Engineering, Concordia University
b_khosr@encs.concordia.ca, bentahar@ciise.concordia.ca

Abstract

When traditional firewall and intrusion detection systems (IDS) are used to detect possible attacks from the network, they often make wrong decisions and block the legitimate connections. In this paper we propose a new architecture which is composed of distributed agents and honeypot. The main focus of our approach lies in reducing the false alarm rate of the attack detection. Using the honeypot scheme, this system is able to avoid many wrong decisions made by IDS. In this system alarming adversaries, initially detected by the IDS, will be rerouted to a honeypot network for a more close investigation. If as a result of this investigation, it is found that the alarm decision made by the IDS of the agent is wrong, the connection will be guided to the original destination in order to continue the previous interaction. This action is hidden to the user. Such a scheme significantly decreases the alarm rate and provides a higher performance of IDS. In this paper the architecture of the proposed system is described, a theoretical analysis of its behavior is given and its possible extension and implementation are explained.

1. Introduction

By increasing the usage of the Internet and implementing commonly used tasks through it, the concept of distributed applications has been considerably grown. Currently firewall and Intrusion Detection Systems (IDS) have been practically developed to block variety of threats through incoming port connections [6].

Intrusion detection technology in general helps to find out the illegal intrusions from inside and outside of the local network by tracking the intruders' trail, such as the records of failure access trails. A typical IDS consists of the following parts: event generator, event analyzer, response units and event databases [9, 11]. The data are exchanged in form of GIDOs (Generalized Intrusion Detection Objects [7] which are represented by a standard common format CIDF) between the parts. GIDO is the specification of messaging as its encoded content is either some particular occurrence happened at some particular time, or some conclusion about a set of events or an intrusion to carry out an action. Each part might be implemented as a single process on a computer (or even a thread), or might be a collection of many processes on a number of computers. The event generator obtains events from data outside the intrusion detection system, generates events based on the traffic thereon and handovers them in GIDO format to the other parts. Event generators provide events as soon as they occur with the possible exception of transport queuing and based on some predefined rules which are dynamically updated. Storage of events is handled in related event databases. When the event generator obtains the required information, it sends an analysis request to its relevant event analyzer. The packets in the monitored network are compared against a repository of signatures which define characteristics of the intrusion while event analyzer analyzes the data and generates new GIDOs which presumably represent some kind of synthesis or summary of the input events. For example, an analyzer might examine whether events being supplied to it now are statistically unlikely to be from the same time series as events supplied in the past or the analyzer might examine sequences of events looking for particular patterns which represent known misuse of the system. GIDOs generated by event analyzers are processed by event tracking system and when the network load is permitted, it submits the data to the agent server. Agent server receives the data from event generator and submits the analysis task to the event analyzer. It also receives the data from event analyzer and submits the tracking task to the suitable event tracker while it monitors and dynamically balances the load of the

1550-445X/08 $25.00 © 2008 IEEE
DOI 10.1109/AINA.2008.44

network. Figure 1 clarifies the basics of IDS in more details. Meanwhile the duty of firewall which cooperates with the IDS is to block some connections between local servers and remote clients. However there is always the possibility of malfunctioning of the firewall which blocks the legitimate connection on a mistake. These results are false alarms [2, 10].

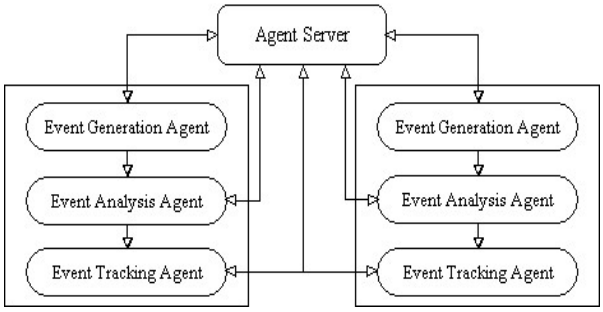

Figure 1. Overall IDS architecture

Honeypot is an unreal network system designed to trap crackers and intruders. This system was introduced initially by Clifford Stoll in 1990 [5]. The honeypot is used as bait in the form of a vulnerable system to trap hackers and keep them away from accessing the critical information in the main system. Using a honeypot system instead of the main network makes it possible to observe the adversary activities, collect information and detect new information coming from the intruders. Some examples of potential malicious traffic can be listed as spoofed IPs, ICMP packets, UDP packets depending on their source and destination and TCP traffic directed to well-known vulnerabilities with unusual characteristics [3]. Honeypots alone are not sufficient for solving or preventing network crimes. Honeypots are hard to implement and they need operators with deep knowledge about computer and network security. If implemented correctly, a honeypot can be an effective tool for information gathering, otherwise by wrong implementation, a honeypot can become another unsecured machine and a tool for the attacker community. Compared to an intrusion detection system, honeypots have a big advantage that the system administrators do not worry about the attacker attempts, because no valuable components are running on the system. This fact enables the system to log every byte and combine this data with other sources to draw a picture of an attack and the attacker.

In this work we use honeypot together with IDS in order to observe the suspected attacker's activities in more details. The reason for this combination is that honeypot is able to log the files and analyze them but it will be overloaded by the normal traffic analysis. However IDS is used for attack detection but it generates lots of false alarms which are not really attacks. The contribution of these two systems strengthens the attack detection mechanism such that only the attacks detected via IDS are directed to the honeypot which deals with the suspected attacks. The architecture avoids blocking legitimate connections on mistake. Such a system considerably decreases false alarm rate. It also records the created signatures of the newly detected attacks which culminate in enhancing the performance of the IDS.

The rest of the paper is organized as follows; Section 2 presents a proposal model of the proposed system. Section 3 describes the structure of the Manager as the main component of the proposed scheme. In Section 4, the detection process is outlined. Section 5 deals with simulation environment and the experimental results and finally, Section 6 concludes the paper.

2. Design and implementation

In this section we outline the requirements of the proposed infrastructure which is considered as a solution to decrease false positives of the network [1]. There are two types of errors concerns, typeI and typeII. TypeI error named false positive is the error of rejecting a null hypothesis when it is actually true, in other word, it is the error of rejecting a hypothesis that should have been accepted. TypeII error named false negative is the error of accepting a null hypothesis that should have been rejected. In this paper we cope with false positive errors. This topology consists of distributed agents (clients or servers), each with associated with an IDS. The architecture can be software resides on an individual computer or a set of computers [8]. The manager, as a main part of the topology, cooperates with honeypot when a suspected threat is detected through the IDS of one of the agents. We will talk more about the manager and its components in section3. As a matter of fact the manager provides an attractive but diversionary playground for the suspected hacker in which honeypot interacts and obtains the required information from the end user.

In this configuration, main network is a "TCP/IP based network" which contains clients and servers. Snort 2.0 is used as IDS and is capable to be configured with an agent. Honeyd is a honeypot system that is configured within the manager in this system. We assume a "switched network" in our approach, with a configurable switch which is

supposed to mirror all the traffic to the specific port; its duty is to send a copy of all packets to the port of the manager.

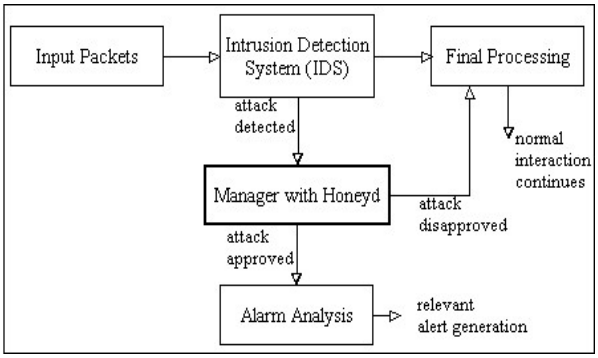

Figure 2. Components of the proposed system

A variety of servers and services can be supported by this system such as application, authentication and authorization, database servers, firewalls, gateways and workstations [5]. Figure 2 illustrates the main structure's functionality.

3. Manager

Since the manager is more complicated than other parts, we describe its partitioned infrastructure separately in Figure 3. The manager consists of four parts: Director to make decisions and rules by creating appropriate template for saving relative data called signatures (explained in section 3.1) [4]; Records as a database which used to save signatures relevant to attacks; Honeyd as a pseudo net used to prevent false blocking; Executive to transform the director's commands into executable orders.

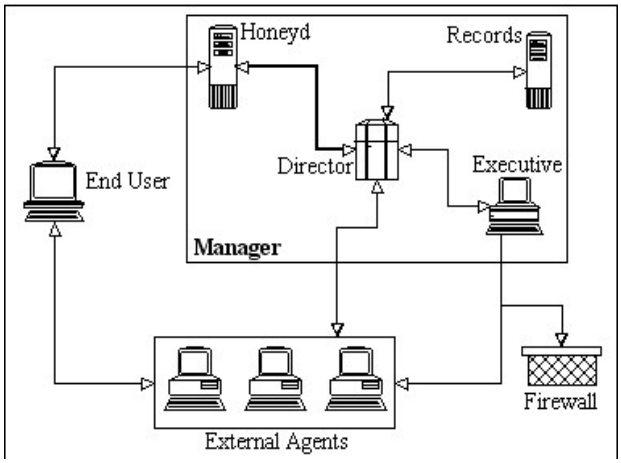

Figure 3. The manager's structure

When the IDS of one of the agents detects a suspected attack, it sends a report concerning the type of the attack and related information to the director of the manager. The manager continues interacting with the end user in order to obtain the required information [1]. While interaction it analyzes the end user's following behavior. This is done based on previous knowledge of data mining which is dynamically updated (by other agents, by administrator or as a result of interaction by an attacker) throughout the system functioning. Generally the data transferred between the components of the manager are categorized as follows; 1)The IP geographical localization of packets' source address which is obtained with tools like Netgeo [3] in honeyd, 2)Passive OS fingerprinting on tcpdump logs which is obtained with tools like POF or Disco in honeyd, 3)Well-known black list of IP addresses which are already saved in the records part, 4)The honeyd environment descriptions for confirmation to variety of services, 5)Analysis of observed sources' ports and other attack features for example whether the attack is parallel or in sequence.

3.1. Director

Director acts as a commander in this system. Once an external agent reports an attack to the manager, the director compares the situation with the "Agent Risk Threshold". Risk threshold is the boundary which is used to warn an attack if interaction exceeds the limit of the specific agent. The director refers to the records to check the threshold assigned to each agent. If the risk of the attack exceeds the threshold, the director engages the manager parts, mainly honeyd, to deal with the connection. Therefore all the important attacks are lead to honeyd for a more close investigation. The honeyd is initially confirmed with the agents through its configuration file provided by the director which will be explained in part 3.3. Honeyd can support different services simultaneously, so it starts responding and interacts with the suspected hacker under the control of the director. Honeyd records in another logbook all activities observed while interaction. It also stores all tools used by the intruders in records as a forensic machine.

The director has a report analysis section which closely investigates all the reports. To do so it refers to records and makes decisions. This is the main duty of director. It also determines whether the interaction has exceeded the "Honeypot Risk Threshold" or not. If the director realizes that the connection is exceeding the risk threshold, it approves the attack and consequently

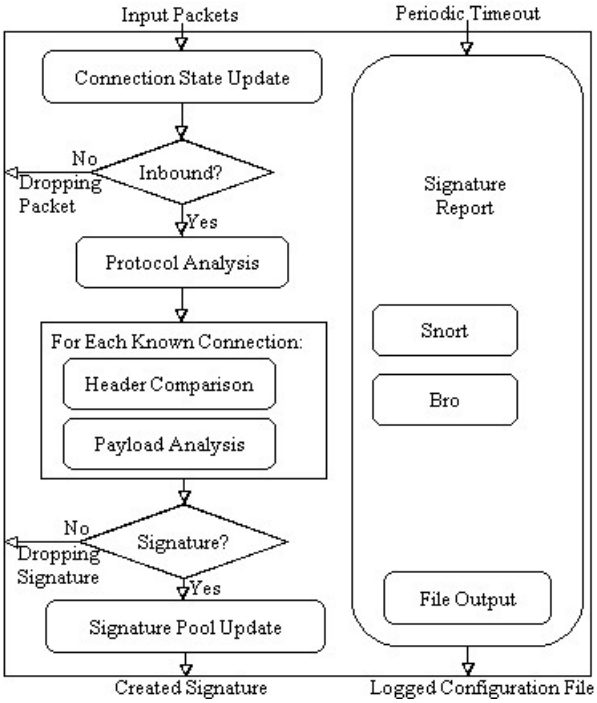

Figure 4. Signature Creation Algorithm [4]

the corresponding report, concerning the appropriate created signature will be ensured to the records. Typical criteria can be listed as follows: when the number of states related to a specific source IP is increasing very fast; when the source IP has exceeded a predefined limit of outgoing connections which is by default 20 connections; when the number of bytes associated with an ICMP state has exceeded a predefined limit. The director applies signature creation algorithm for analysis of the reports. Figure 4 explains the algorithm in more details. Input packets are supposed to be categorized and if they belong to the previously initializes state, the state would be updated. The algorithm performs protocol analysis for each connection and creates a temporary signature to check out to be new and relevant. The accepted signatures are inserted to the signature pool.

3.2. Records

The records provide a database including a crime table, policy table, reports and event logs. The director for making decisions refers to this database and also enters new rules and signatures derived from new attack. Director constantly modifies the signature list that are used by external agents. Records part is equipped with a system which organizes the signatures coming through the database and enables the director to get access to the current signatures and update them or insert new ones.

3.3. Honeypot

As a virtual unreal network, Honeyd [4] has been used in this topology. Honeyd is generally designed for Unix-based operating systems such as OpenBSD or Linux. Its duty is to detect unauthorized activities by monitoring all the unused IPs in the network. It is capable of controlling all the unused IPs at the same time. Any time honeyd generates an alert, it is most likely a real attack not a false alarm, thus the director approves the attack. Honeyd can also detect unknown attacks such as new RPC 0-day as well. With honeyd, we have additional option of creating emulated services that interact with the attacker. The emulated services allow the director to determine what the attacker is attempting to do, what he is looking for. This is done by creating scripts that listen to specific ports and then interact with the attacker in a predetermined manner. For example Ftp script emulates a wu-ftpd daemon on linux, or Telnet connections on a Cisco router. The scripts can be written in almost any languages such as Perl, Shell or Expect. Since attackers often remotely fingerprint operating systems by using tools like Nmap or Xprobe, honeyd takes same fingerprint database used by Nmap to spoof the responses of any operating system which is emulated.

To implement honeyd the director compiles an additional tool Arpd [12]. Honeyd cannot do everything alone and requires the help of Arpd. Arpd is used for ARP spoofing; this is what actually monitors the unused IP space and directs the attacks to the honeyd. Honeyd is only able to interact with attackers. For example if the aim is to monitor all the unused IPs in the 192.168.1.0/24 network, the required commands to start both Arpd and Honeyd are as follows:

arpd 192.168.1.0/24

honeyd -p nmap.prints -f honeyd.conf 192.168.1.0/24

Based on the commands, the Arpd process monitors the unused IPs, directs all corresponding attacks to the honeyd and spoofs the victim's IP address with the MAC address of the honeypot. For the honeyd command *-p nmap.prints* refers to the nmap fingerprint database and *-f honeyd.conf* is the honeyd configuration file. Figure 5 shows a sample of configuration file. Honeyd configuration file starts by creating different types of computers should be

emulated, called templates in honeyd. These templates define the behavior of each emulated operating system. In this system two different computers have been created: default and router. Personality is what operating system will emulate at the IP stack level. The personality does not affect the behavior of the emulated services; it only modifies the behavior of the IP stack. Next step is to define the behavior of each port.

```
## Honeyd configuration file ##
### Windows computers (default)
create default
set default personality "Windows NT 4.0 Server
SP5-SP6"
set default default tcp action reset
add default tcp port 110 "sh scripts/pop.sh"
add default tcp port 80 "perl scripts/iis-0.95/main.pl"
add default tcp port 25 block
add default tcp port 21 "sh scripts/ftp.sh"
add default tcp port 22 proxy $ipsrc:22
add default udp port 139 drop
set default uptime 3284460

### Cisco router
create router
set router personality "Cisco 4500-M running IOS 11.3(6)
IP Plus"
add router tcp port 23 "/usr/bin/perl
scripts/router-telnet.pl"
set router default tcp action reset
set router uid 32767 gid 32767
set router uptime 1327650
# Bind specific templates to specific IP address
# If not bound, default to Windows template
bind 192.168.1.150 router
```

Figure 5. Honeyd configuration file

For example, in the template default, it has assigned all the TCP ports the behavior reset, so they respond with a RST to any connection attempts (for UDP, ICMP connections unreachable). By using different scripts on ports, it actually proxies all SSH connections back to the attacker. Once the template created, it has to decide which IP addresses are bound to which template. In this case, if anyone attempts to connect to IP address *192.168.1.150*, they will be interacted by the honeyd using the router template. The template with the name default is default for all other connections to non-used IP space. So if any connection is made to any unused IP space in the *192.168.1.0/24* network, it will get a windows box emulated by honeyd, except for the IP *192.168.1.150* at which it will get the Cisco router.

3.4. External Agents

The first step of the detection process is done via agents, the real intrusion detection systems used for the primary detection. Intrusion detection system is discussed in section 2. Considerable point in this system is that the agent handles all the network connections of the computer which has been installed on in addition to act as IDS. The IDS records all incoming and outgoing packages in tcpdump binary format. This is done by the OpenBSD logging mechanism. The use of this standard format simplifies all data manipulations; it also allows the use of well-known tools like tcpdump, ethereal and ngrep [3].

4. The detection process

By initializing the system, the director matches honeypot with the main network, then honeyd is ready to interact with the suspected adversaries. In our method, director as a commander supervises data transmission reports. When the external agents detect an attack, they report the event to the director which will decide how to manage it. Such attacks can be listed as follows; Nimda, CodeRed (3 versions), MyDoom, W32/Welchia.D, Attempts to access the IIS-samples, Attempts to get '/etc/psswd, Attempts to execute cmd.exe, Attempts to open a new network socket in order to download further hacking utilities, A warm that starts on each infected system an email relying server which it uses for its further spreading. Director orders switch to send a copy of all packages which their destination is the said agent, and also commands the agent not to answer, thus director will receive the trail of suspended threat packages. All of the process must be hidden to the end-user.

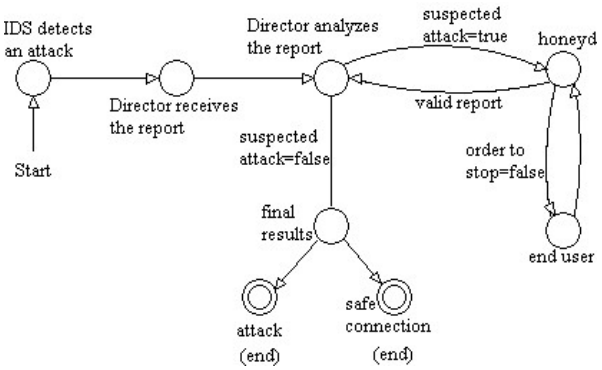

Figure 6. Director's analysis behavior

Indeed, the connection will be shifted to the honeyd. Incoming packets are classified to three major

Internet protocols: ICMP, TCP and UDP. Before processing, the dispatcher of the honeyd queries the configuration database to find a previous configuration that corresponds to the destination IP address. If there is no specific configuration, a default template is used. Thus a packet together with corresponding configuration is handed to the verified protocol. The ICMP protocol supports most ICMP messages. By default, all honeypot configurations are able to respond to echo requests and process unreachable destination messages. For TCP and UDP, the framework establishes a connection to arbitrary services. Services as external applications receive data on stdin and send their response to stdout. The behavior of the service completely depends on the characteristics of the external applications. When a connection request is received, the protocol checks if the packet is established before. Then any new data is sent to the already initiated service application. If the packet request is new, a new process is created to run the appropriate service application. The corresponding protocol responds like a normal network to the end-user, as director monitors all the event reports which have been sent by honeyd. Figure 6 illustrates the behavior of the director using in formal method. Director refers to the database saved in records in order to make decisions. If honeyd by any reasons requires any information to send to the end-user, director via executive will ask external agents to respond and send back. This information will be dispatched to the enduser under the control of the director. When director completely verified the attacker, it would command the executive to break up the connection and also stored information of the attacker will be added to the database. Executive with an SNMP will order firewall to block such packages henceforth. Thus the attack is detected, besides the attacker identified which is substantial information for administrator. If the director found out that the detection was wrong, without breaking up the connection via executive, it will ask external agents to carry on connection. It will also send all saved information relevant to the connection to the previous destination. Therefore the number of false positives will be decreased. It is a crucial factor to show the precision and accuracy of the detection.

5. Simulation and experimental results

The detection capabilities of the system vary in different traffic conditions. Different statistical throughput, delay or even the order of requests for traffic introduces changes in the detection capabilities of the system. Hence to gain better performance in detection capability, we provided a generation block in the simulation which is capable of generating different types of traffic in order to maintain the authentic ”real-time” scenarios. In reality, the real-time scenarios are created by the different environment profiles in terms of scripts. These scripts interact with different servers having services installed on them. The scripts connect to the server and interact with these services on the server. In the simulation package 6 legitimate traffic scripts and approximately 40 attack scripts have been implemented to attack continuously the distributed agents. Generally the system’s detection capabilities are dependent on the number of times the combined IDS and honeyd correctly distinguish an illegitimate traffic from the background traffic. Legal traffic is used to generate the different background traffic to test the system. Six legitimate protocols are HTTP protocol, SMTP protocol, POP protocol, SSH protocol, FTP protocol and TELNET protocol used in the system simulation.

In the simulation mostly the attack traffic was directed towards the system in an isolated test network starting at very low network load range. Therefore we generated a high range of attacking packets which increases the load of the system in order to evaluate the ratio of false positive. In the honeyd system alone, since there is no initial detection, it will automatically ignore the packets when the load is high, but in the proposed system since the packets are filtered initially by the IDS we do not encounter such a bottleneck at least up to very large number of attacking packets generated. The network load was measured as the percent of total network bandwidth occupied by the traffic. Considering D and H respectively to be the set of IDS and honeyd detected intrusions and A to be the set of actual intrusions, with T being the total number of transactions, the corresponding false positive ratio of the IDS and honeyd alone are gained as follows:

$$\text{IDS false positive ratio} = \frac{|D \cap A|}{|T|} \quad (1)$$

$$\text{Honeyd false positive ratio} = \frac{|H \cap A|}{|T|} \quad (2)$$

In the proposed approach the aim is to decrease the false positive ratio. Both IDS and honeyd contribute in this ratio as long as each one has its own attack detection ratio. The reliability of attack detection is defined as the ratio of false positives to total alarms raised. For both IDS and honeyd there is the attack detection ratio and probability of attack as determined in equations 3 to 6. Equation 7 combines the detection ratio of honeyd with IDS as honeyd detects in

Table 1. Creation of 95% Confidence Intervals for the Conventional Systems and the Proposed System

Measures and Characteristics	Conventional IDS	Conventional honeyd	The proposed system
Measured ratio of false positive in six runs	0.089	0.091	0.045
	0.084	0.103	0.063
	0.089	0.086	0.076
	0.079	0.093	0.087
	0.085	0.088	0.135
	0.088	0.085	0.132
Average ratio of false positive	0.088	0.091	**0.085**
Standard deviation of false positive ratio	0.000018	0.000044	0.00096
Half value of the confidence interval	0.0044	0.0069	0.0324
Full confidence interval	(0.084,0.092)	(0.084,0.098)	**(0.053,0.117)**

condition of IDS detection. The corresponding values are gained as follows:

$$\text{IDS attack detection ratio}=\frac{|D-(A\cap D)|}{|D|} \quad (3)$$

$$\text{Honeyd attack detection ratio}=\frac{|H-(A\cap H)|}{|H|} \quad (4)$$

$$\text{IDS prob. of attack}=\frac{|A-(|D-(A\cap D)|)|}{|T-D|} \quad (5)$$

$$\text{Honeyd prob. of attack}=\frac{|A-(|H-(A\cap H)|)|}{|T-H|} \quad (6)$$

$$\text{False positive ratio}=\frac{|H\cap A|_D}{|T|} \quad (7)$$

In equation7 false positive ratio represents the ratio of the attacks which have been wrongly detected by IDS but denied by honeyd. There is still possibility that the honeyd does the same mistake and consider the good data as attack like what IDS did. This is where we see as the number of transaction drastically increase the system performs poor outputs. However there is the chance of mistake by IDS that do not consider the real attack and therefore do not filter and shift to the honeyd. We did not assume this possibility in this system as we believe there should be another solution for this type of error. Moreover there is the possibility of the mistake generating from honeyd as IDS detects good but honeyd on mistake declares the real attack as good data, shown in equation 8. This is so rare because generally honeyd is a good detector as long as in this system we do not bombard honeyd by torrent of data therefore honeyd makes a very good performance. Equation8 shows the honeyd false negative ratio.

$$\text{Honeyd false negative ratio}=\frac{|(D-H)\cap A|}{|T|} \quad (8)$$

During the period of test, the network load was changing between the minimum and maximum range values. The network load input was therefore the average network load experienced during the test period. The testing was carried on until the system broke down and fell into the poor or the very poor category. The value of the test determined the false positive ratio of the system. Figures 7 to 9 declare the results of three different tests.

Our tests show that the proposed system can manage to improve the detection accuracy in comparison with the conventional honeyd and IDS. As it is clear, in the proposed system the objective is to re-check the attacking packets in order to avoid the false positives while both honeyd and IDS have their own false positive ratio which is definitely greater than what we gained as a cooperation of these two systems.

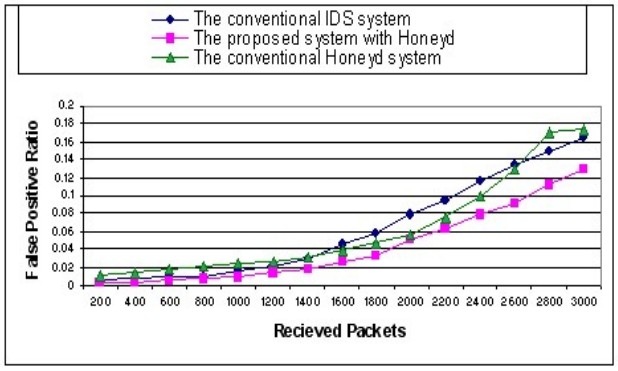

Figure 7. Performance of the three schemes for inter arrival packet time of 0.016

Since in the proposed system, the director decides on the accuracy of the initial detection of the IDS, therefore we have a better performance of false positive ratio in all the network loads. The best improvement is done at the time when 1700 packets have entered the system in figure 7 while the difference of the ratio is 0.041 and at the time when 1800 packets have entered the system in figure 8 while the difference of the ratio is 0.075 and when 2600 packets have entered the system in figure 9 while the

difference of the ratio is 0.125. Table 1 represents the creation of confidence intervals for six runs. As the experimental results show, the best performance measured is 0.077. However this ratio can be decreased even more in the proposed system. But one of the purposes of the proposed systems is to enrich the database with the dynamically generated signatures and rules which are very useful for the analysis of the upcoming network traffic.

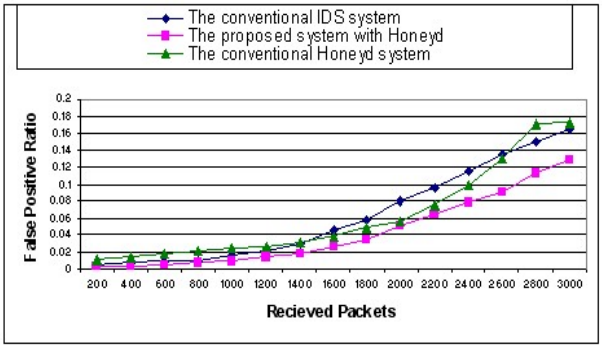

Figure 8. Performance of the three schemes for inter arrival packet time of 0.020

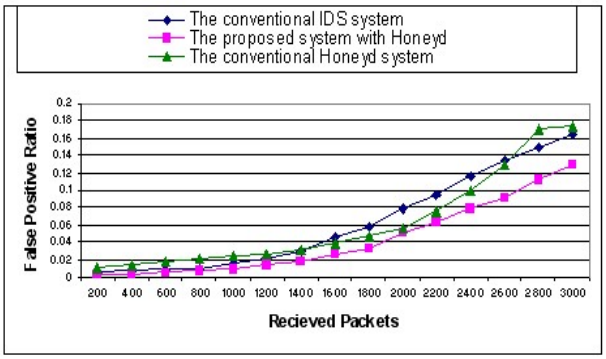

Figure 9. Performance of the three schemes for inter arrival packet time of 0.024

6. Conclusion

The paper presents a new system on a distributed agent based network. The system, in case of suspended adversaries, changes the path of connection to the honeypot system for a more close investigation. Decreasing false alarms' rate is the system's objective. The system also tries to store any necessary information about the identified attacker. In the simulated system we obtained good results which were declaring the decreasing in false positive ratio. Due to the filtering of IDS the load of the system directed to the Honeyd is decreased leading the Honeyd to better performance in packet dropping. We tested the proposed system with different number of packets we generated. In order to test the scalability of the system we did the testing with three different network loads which were the rate for generation of the packets.

We should pay more attention to the following problems: How to shift the connection with the all related data in more details. We have to expose director to more aggressive traffic patterns to get a better understanding of its performance. Director has to intelligently deal with the data communication between honeypot and the end user. The coordinating model of honeypot in relation with agents still needs further discussion.

References

[1] Khosravifar B. and Amintabar A. Decreasing False Alarm Rate by the Collaboration of Honeypot to Agentbased Intrusion Detection System. pages 245–255, Istanbul, Turkey, Oct 2006. Proceedings of the Second IEEE International Conference on Technologies for Homeland Security and Safety.

[2] Hannon C and Richard J.R. Addressing in Geographically Distributed System. Computer Science, ENC. Proceedings of the Fourth Mexican International Conference on 8-12 Sept. 2003 Page(s): 182 189, 2003.Hoepers C. and Steding J.K. Honeynets applied to the CSIRT Scenario.

[3] Hoepers C. and Steding J.K. Honeynets applied to the CSIRT Scenario. 2003.

[4] Kreibich C. and Crowcroft J. Honeycomb: Creating Intrusion Detection Signatures Using Honeypots. volume 34. ACM SIGCOMM Computer Communications review, Jan 2004.

[5] Stoll C. The Cuckoos Egg, Tracking a Spy Through the Maze of Computer Espiounage. pages 181–191. Proceedings. 25th Annual 1991 IEEE International Carnahan Conference, Oct 1991.

[6] Hellman M.E. Diffie W. An Introduction to Cryptography. volume 67, pages 397–427. Proceedings of IEEE, March 1999.

[7] Chao W. Donghai H. and Li Q. Example Anatomy of IDS. volume 2. Tsinghua University, 2002.

[8] Gartner F. Byzantine Failures and Security. Franksurt, Germany, 2003. Proceedings of GI-Jaherstagung.

[9] Xiangliang L. Guangchun L. and Jiong Zhang L. MADIDS: A Novel Distributed IDS Based on Mobile Agent. pages 100–107. Parallel and Distributed Processing Symposium. Proceedings. 19th IEEE International, 2005.

[10] Grosspietch K.E and Silayeva T.A. A Combined Safety/Security Approach to Cooperative Distributed Systems. pages 115–125. Proceeding of the 18th International Parallel and Distributed Processing Symposium, IEEE, 2004.

[11] Dobry R. and Schanken M.D. Security Concerns for Distributed Systems. volume 10, pages 12–20. Computer Security Applications Conference, Dec 1994.

[12] Talabis R. Honeypots 101: A Brief History of Honey-Pots. The Philippine honeynet project, 2002.