A Dynamic Approach for Honeypot Management

Received 7 November 2012, Accepted 1 December 2012

Alireza Saeedi
Master›s student, Department of Computer,
Hamedan Branch, Islamic Azad University,
Science and Research Campus, Hamedan, Iran
+98 912 4451704
saeedi.mail@gmail.com

Mohammad Nassiri
Assistant Prof. in Computer Engineering,
Assistant Professor at Bu-Ali Sina University,
Hamedan, Iran
+98 811 8292505
m.nassiri@basu.ac.ir

Hassan Khotanlou
Assistant Prof. in Computer Engineering,
Assistant Professor at Bu-Ali Sina University,
Hamedan, Iran
+98 811 8292505
hkh@basu.ac.ir

ABSTRACT

Honeypot is a security device the value of which lies mainly in discovering and inspecting, being attacked and being at risk. Most of the present honeypots are configured and installed on the network statically. In some cases, considerations have been made on dynamic configuration of honeypots at the time of installation. However, no study has been carried out on how to instantaneously change the configuration of honeypots based upon the analysis of the collected events from various network elements including routers, firewalls, spam identifiers and honeypots. In this paper, we propose a method that the honeynet is automatically configured so that the conditions are prepared for trapping the threats based on the reports sent from several elements of the network and also the defined rules of the system. Unlike the other methods which wait until the threat reaches the honeypot, the main idea in the proposed method is to configure it to move to attract the attacks. The present scheme has been evaluated in a real environment. The results of the evaluation, illustrated the efficiency of the suggested method.

Keywords

Honeypot, Dynamic Management, Network Security

1. INTRODUCTION

Honeypot is a security element in computer networks which is used to identify the system's weaknesses and to collect the necessary information for following and tracking intruders. In other words, a honeypot is security resource whose value lies in being probed, attacked, or compromised [1].

Today, because of their unique advantages, honeypots are of special interest to many security teams and companies and they have installed several versions of them. Based on the extent to which they interact with users, honeypots can be categorized into two groups of high-interaction and low-interaction. Low-interaction honeypots imitate the real services by installing fake services and save all types of activities on themselves. High-interaction honeypots are real or virtual systems on which real services have been installed. Since honeypots are plug-ins, no traffic should enter or exit them. Besides, no extra activity should take place within high-interaction honeypots, making any traffic or extra activity on them to be detected as suspicious or attack which causes honeypot reports usually brief but useful. Furthermore, more complete information about the attack can be obtained from honeypots, because they allow hackers or malware to communicate with a real system. Also, unknown attacks can be identified by this means. Since honeypots are located at the beginning or the end of encrypted communications, the content of communications can be revealed.

Most of the present honeypots are configured and installed statically (they should be configured by the user and then be placed on the network). However, some methods for making honeypots dynamic have been provided. For example, dynamic honeypots have been introduced in [2] and their installation methods and difficulties in each method have been discussed as well. According to [2], the major problem of dynamic honeypots is lack of information about the network. To resolve this problem, an active and a passive solution have been recommended. With these approaches, it is possible to discover data about the structure of networks including number of stations, types of operating systems, provided services, and stations' communication information through monitoring the network data. This information can be helpful to adjust and develop honeypots. Their configuration difficulties can be reduced and it can be done automatically by means of automatic collection of such information. This feature not only leads to maintaining low costs of maintenance, but also makes honeypots constantly compatible with the environment. In this case, the honeypot identifies the network and selects the free (non-dedicated) IP addresses.

A passive tracing system has been used in [3] to collect information about the network and configure honeypots. The schema of a dynamic honeypot system has been studied in [4] in which both low-interaction and high-interaction honeypots have been used. In addition, both active and passive tools have been utilized to monitor the network. The procedure is that at the beginning, the dynamic honeypot server starts to collect data about the network including the number and the types of operating systems, provided services and communication. The data collection process is done by sending direct messages which are adjusted for the very purpose and tracking the network communications. Having collected the data and identified the structure of the network, the dynamic honeypot server initiates to install and configure to suite the network (the network administrator may optimize the structure).

Other suggestions have been introduced in [5], especially substituting the honeypots with the main server (with the same IP address), when the main server stops working or is interrupted. The main difference of this method with the previous ones is caused by the identification and changing the honeypots based on the data received from the changes in the network and also the added or removed systems which occurs instantly.

Ways to make patterns automatically to identify unknown worms have been discussed in [6]. These suggest installed versions in which low-interaction and high-interaction honeypots have been used to monitor suspicious traffic. After putting packages similar to white list patterns aside, malicious patterns are generated automatically. In order to reduce the reaction time, obtained patterns are distributed in a network which has the identification and prevention of intruders as duty. Consequently – and within certain duration of time, – these patterns are refined to decrease the rate of false identifications. Another advantage of this scheme is that the growing or declining regime of the worms' hostility is analyzed and they are sorted based on their significance.

In [7], dynamic honeypots mechanism has been used to reduce the load on servers. In this method, the healthy flow is distinguished from the attack flow. This is done by analyzing the traffic using characterisation and entropy discoverers. Consequently, honeypots and servers are installed on the network with proportion to the attacks. Besides, the time and location of the activity on the network is identified. Attack flows are divided among the honeypots and the healthy flow is divided between the servers. In [8] a schema of honeypots has been created based on neural systems with prediction of the vulnerable probability and intrusive behaviours in advance as objective. The output of this system is a black list.

Further in this paper, we introduce a theme for a dynamic management of honeypots in an intranet in section 2. In Section 3, we propose a scenario to evaluate the efficiency of the suggested method and discuss the results for each evaluation criterion. Finally, we present the conclusion and further works.

2. OUR PROPOSAL

In this paper, dynamic administration of honeypots and changing their configurations is discussed. The configurations are based on reports obtained from other sensors of the network such as routers, firewalls, intrusion detection systems and the honeypots themselves. For example, as soon as the report of a port being scanned is delivered from the open port of a server, the command of opening that port is given to one of the high-interaction honeypots to become bait for the attacker.

The following elements are used in this design:

- Low-interaction honeypots
- High-interaction honeypots
- Network Sensors
- Central administration system

The overall schema of the system is shown in Figure 1. In this design, low-interaction honeypots connect directly to the central administration system via the network. However, high-interaction honeypots are disconnected from the rest of the system by a firewall called Honeywall. The reason is that if intruders reach these systems, the way through the network remains closed. In the next sections, we discuss each of these elements independently and also explain the method by which different elements communicate.

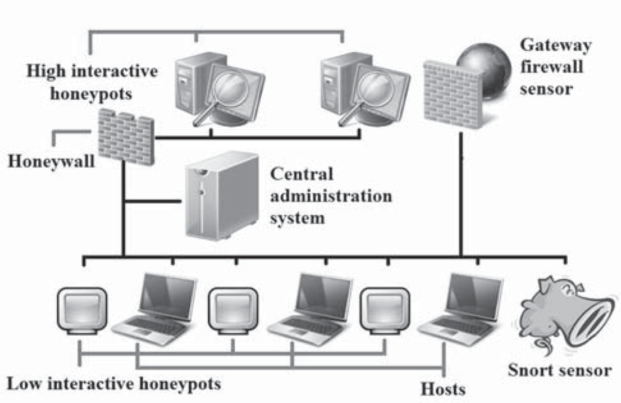
High interactive honeypots
Gateway firewall sensor
Honeywall
Central administration system
Low interactive honeypots
Hosts
Snort sensor

Figure 1. The schema of the proposed design

2.1 Low-Interaction Honeypots

Honeyd is the software we used to install low-interaction honeypots. We chose this software because it is open-source and its installed versions are available on almost every operating system including UNIX and Windows. Also, it has the option of adding new systems by writing new scripts by languages such as Python, Perl and Bash. Another feature is that it is possible to use Nmap software fingerprint database to prevent fake responses of several simulated operating systems [9].

One of the problems with the present low-interaction honeypots including Honeyd is out-of-date services for different service providers. In order to resolve this problem, a piece of software has been developed in Python for Honeyd called script maker software. The routine of this software is that the raw commands which relate to that service are defined by identifying all the input to the software. Then, the false and true values of the inputs are figured out. After that, the interdependency of commands will be identified, since some commands should necessarily precede others. In the next step, we know whether the service needs identification. If so, a username, a proper password and also commands relating to entrance will be issued in the script maker software.

To this stage, the data relating to the service in the software is defined. Now, we execute the service we intend to simulate. For example, it is possible to execute any of the service providing software for simulating purposes in the FTP service. Thereafter, the address and the port of the real service provider are given to the script maker and the procedure of making the script is run. This software sends the defined commands with different combinations to the real service provider. It starts to send commands with true, false and nonparametric inputs for the real service provider. It also submits the commands for each condition – considering the interdependency every another time. In case the service needs to be identified, the whole procedure is transmitted once before and another time after logging. We implement the responses from the real service into form of a decision tree in Python.

Loading the script in Honeyd, the decision tree will be browsed and the proper response will be returned. Therefore, any service can be updated and automatically simulated.

2.2 High-Interaction Honeypots

In order to implement high-interaction honeypots, we used VMware software to build virtual systems. Using virtual systems leads to simplified maintenance of systems. It is also more economical than the real systems. In addition, VMware has provided solutions for controlling virtual systems via other software, a method which has been used for controlling purposes in central administration systems. In order to monitor the events inside a system, another piece of software was installed to control the following tasks inside the honeypots continuously:

- Monitoring the hard disk including adding, removing and changes in all available files on the system.
- Monitoring the system registry in Microsoft Windows including adding, removing, and changes in the registry.
- Monitoring all connections to and from the system
- Providing a list of running processes
- Monitoring CPU usage
- Monitoring memory usage

The possible changes in all the aforementioned tasks are reported to the central administration software. As for the first four tasks, a list of exceptions is defined in the software, which can be updated by the central administration software. This list is prepared to remove false reports in the exceptional cases; for example, the files that are change continuously by the operating system and we do not want to report them. As for the CPU and memory resources usage, a threshold is defined in the software. As such, if the usage is more than the defined threshold, a report will be sent.

2.3 Network Sensors

Any sensor with the capability of sending a syslog can be used as the sensor, since this is the form in which the central administration system receives the sent reports by the sensors. The most important sensors of this type are routers, firewalls, and intrusion detection systems such as Snort and Bro.

2.4 Central Administration System

Central Administration System is the main part of the suggested design that decides besides controlling the honeypots. This system is consisted of three parts:

- The report receiving unit
- The legislative unit

- The deciding unit

The report receiving unit receives two types of reports: reports sent from the honeypots, and reports from other network sensors in the form of syslog.

It is possible to make rules in the legislative part to control the honeypots. The rules are defined as follows:

If {condition(s)}, then {agent(s)} perform this {function(s)}.

For instance, some conditions used in our suggested mechanism are: "If (this) message is (not) included in the text of the alarm "or "always". Besides, the agent can be: "the sender of the alarm", "all high-interaction honeypots", "all low-interaction honeypots", or "the address to a specific honeypot". As for the functions used in this design, "performing a service", "changing the IP address", "file execution", "VMWare revert to the snapshot where the system works flawlessly and is not infected", and "taking a snapshot of the VMWare at the current status" can be among the choices.

The deciding unit in the central administration software analyzes the reported alarms based on the available rules and performs the tasks as defined in the rules.

3. EVALUATION

In order to analyze the efficiency of the system, we first discuss the following four parameters: Deception, False alarm, Identification, and Vision.

In the section relating to monitoring the false alarm and identification, a network with 4 high-interaction honeypots, 4 low-interaction honeypots, 4 Snort sensors and router will be used. The high-interaction honeypots were installed virtually and with 50 gigabytes hard disk and 1 gigabyte RAM per each honeypot. Besides, the virtual machine was mounted on hardware with a Corei7 Intel processor with speed of 2 GHz.

3.1 Deception

One of the main objectives of honeypots is to deceive the intruders or malware. In order to measure the extent of deception, it is necessary to be able to distinguish between the honeypots and the real system. Thus, the less the honeypot is distinguishable, the more it has been able to fake a real system and consequently deceive the intruder. Among the main methods of distinguishing a honeypot are the followings:

- Scanning the ports
- Fingerprint
- Identification of the virtual environment
- Hospitality

3.1.1 Scanning the Ports

One way to doubt cast whether the system is a honeypot is that numerous ports are normally open on the static honeypots and many services are provided by them. However, as honeypots provide services dynamically in the suggested method, services are never provided all at once. The service providing command is received by the central administration system using the events. The defined rules are issued only in case of necessity.

3.1.2 Fingerprint

Fingerprint is another approach for identifying honeypots. Service providers and different operating systems possess their unique fingerprints. On the other hand, some tools can identify the type of the operating system or the service provider in the target station by sending specific commands. Hackers use these tools to ensure the reality of the systems. However, this approach is not beneficial in the case of high-interaction honeypots, since they are real systems and have a normal behavior facing the software. The tools are useful for some low-interaction honeypots and as discussed above, Honeyd uses the Nmap fingerprint database to simulate the responses, which is one of the most important tools in identifying fingerprints. This database is updated constantly and its latest version can be downloaded from the Nmap website directly.

3.1.3 Identifying the Virtual Environment

Hackers can identify high-interaction honeypots, if they figure out whether they are in a virtual system or a real one. Unfortunately, there are almost some ways for identification for every virtualization software. Although, the mere fact that the system is doesn't necessarily mean that the system is a honeypot, because currently many services are installed on virtual systems in order to reduce the costs. Nevertheless, to eliminate this method, real systems should be used that leads to higher costs both from the installation and maintenance aspects.

3.1.4 Hospitality

Hospitality means reducing the security level considerably to entrap the hackers in honeypots. Easy passwords, lack of antivirus, and non-updated software and operating systems are examples of reducing the security level. This might help the hacker guess a honeypot exists. However, our approach is not to reduce the level of security of the honeypots. We believe that the security level of a honeypot should be equal to that of real systems on the network. One of the aims of the script maker software is to install similar services to the real ones on the network. We are looking for hackers and malware that can damage our network with the actual security level, not all hackers or malware on the network. For instance, the operating system of all our real systems are Windows XP with service pack 3, being aware of the fact that the security hole in service pack 2 has been fixed in service pack 3. We believe that malware which abuses this security cavity is not of importance, because they can not harm the real systems on our network. As a result, we use the main systems in installing honeypots and consequently we are not worried about hospitality.

3.2 False Alarm

Another parameter is the amount of false alarms that result from false detection of the system. In this section, the effect of events received from the network in the form of syslog on the false alarms created in the system is discussed. The results have been obtained within a workweek on a network with 30 active computers, 4 high-interaction honeypots, 4 low-interaction honeypots, 4 Snort sensors and a router. The results have been illustrated in Figure 2.

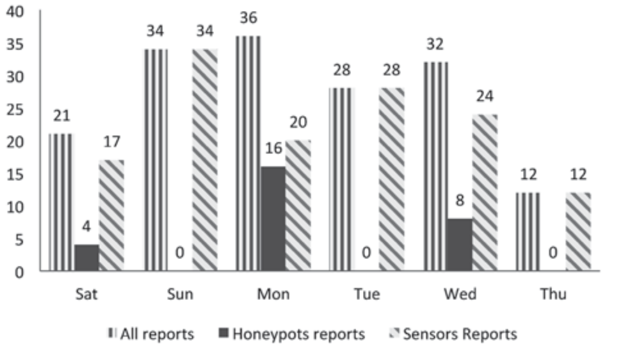

Figure 2. Created reports in the test

The results have been obtained while performing several vulnerability tests: scanning the network, and the normal traffic on the network. Figure 2 shows that 163 alarms have been created and only 28 of them have been from honeypots. Thus, we concluded that the use of sensors can lead to having a wider vision and registering more events.

Another important parameter is the number of false alarms. Figure 3 illustrates the number of false alarms generated in this test. Syslog reports have different severity levels. Three false reports in this test are at the alert degree.

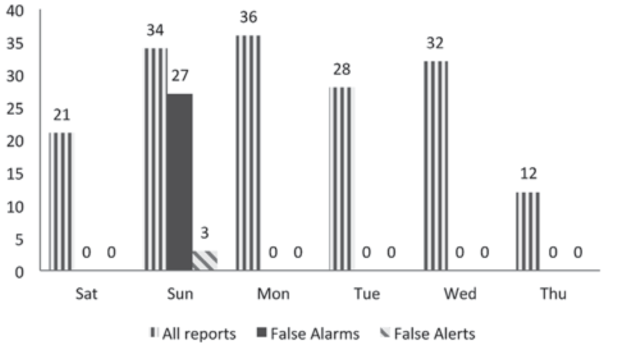

Figure 3. False reports in the test

As shown above in the test results, the more reports are made the more false alarms are generated. The number of false alarms can be reduced by changing the threshold of the severity level of reports. A primary parameter in sending alarms from network sensors especially in intrusion detection systems is their configuration. In addition, the conditions and the traffic of the network are also influential on the results of the test. That answers why many different conditions can be imagined for this test. It can be concluded from the results is that using other sensors on the network increases the true alarms and expands the vision, but meanwhile brings about the occurrence probability of false alarms. This has to be reduced by properly configuring and correctly choosing the threshold of reports accuracy.

3.3 Identification

Malware that distribute on a network or hackers who want to creep into a system perform identification by taking advantage of the vulnerabilities of the software and services. Therefore, Metasploit software has been used in this research to test the different vulnerabilities and the intrusion inside the honeypots. In each case, the time of the first received alarm in the central administration system has been calculated and illustrated in Table 1. It is noteworthy that in these tests, the period of hard disk monitoring has been set to 30 seconds, meaning that the hard disk and registry monitoring process takes place every 30 seconds. Since these algorithms take time, no shorter period was possible using the actual lab system.

Table 1. Results of the identification test

	Vulnerability Description	Initial Report
1	FTP Authentication Scanner	2 Sec
2	EasyFTP Server LIST Command Stack Buffer Overflow	5 Sec
3	GoldenFTP PASS Stack Buffer Overflow	4 Sec
4	HTTPDX h_handlepeer() Function Buffer Overflow	5 Sec

The first test was performed using a dictionary in order to enter the FTP server. Within the duration of 2 seconds, the intention to enter was reported from the honeypot and was registered in the central administrative software.

In the second test, a vulnerability test on the EasyFTP software was used that is an FTP service provider. In this test, sending a false value to the LIST command leads to the stack overflow and eventually the destructive code can be executed on the target computer. Executing the EasyFTP software on the high-interaction honeypot and using this vulnerability, we transferred a file to the honeypot and executed it. This is a method by which hackers and malware copy and run their destructive program on the target computer using vulnerability. The first alarm to announce the addition of a new process to the list was reported in 5 seconds to the central administrative system and the alarm to alert the addition of a file to the hard disk was reported within 18 seconds. The reason for this delay in the file addition alarm is the 30 second cycle of the hard disk monitoring procedure. In this example, the time has been calculated while sampling. This is the reason why it has been detected less than 30 seconds. Both reported alarms are considered dangerous, because as no specific action is usually taken on a honeypot, the appearance of a new process or addition of a file on the disk is suspicious.

In the third test, vulnerability on the GoldenFTP software (an FTP service provider) was used. Sending a false value to the PASS command leads to the stack overflow and the destructive code can be run on the target computer. Also in this test, a file was copied and run in this way on the honeypot. The first alarm to announce the addition of a new process was reported in 4 seconds and the first file addition to hard disk was reported in 15 seconds.

In the fourth test, vulnerability on the HTTPDX (a web service provider) was used. A large http request was sent to the *h_handlepeer( )* function and this lead to overflow in the stack and therefore the destructive code can be run in the target computer. Again, the previous test was performed. The first alarm due to addition of a process was reported within 5 seconds and the first file addition to the disk alarm was reported after 12 seconds.

3.4 Vision

One of the greatest innate disadvantages of honeypots is their narrow field of view. Honeypots are only aware of the events they directly encounter. In this research, we resolved it collecting data from other sensors. As mentioned in the previous tests, adding to the number of sensors leads to an increase in the number of collected events from inside the network that helps putting the honeypots on a proper route by defining appropriate rules; in other words, this approach can widen the vision of honeypots indirectly by means of the central administration system.

4. CONCLUSION AND FURTHER WORKS

In this approach, the property of honeypots deception level was optimized for 3 methods out of 4, the explanation of which was given in section 3.1. Besides, other network sensors have been used to broaden the honeypots vision. The test results illustrate that we have accelerated the process of identifying malicious behaviors. Further, the idea of script maker software was considered to resolve the problem in low-interaction honeypots, i.e. lack of simulation of different versions of service providers and not being up-to-date.

However, the number of false alarms was augmented along with the increase in the reports. To solve this problem, we showed that choosing a proper level of severity and precision of events is a way to reduce the number of false alarms. Besides, correct adjustment of the sensors also reduces the number of false alarms. The aim of this research was to look at the dynamic administration of honeypots, instead of intelligence. Therefore, research on providing a method for automatically producing rules can be the subject of further research in the field, based on the presumptions discussed in the paper in order to achieve the optimum smart approach.

REFERENCES

[1] Bologna Declaration. (1999). Joint Declaration of the European Ministers of Education. *Bologna, 19 June*.

[2] Piasecka, A. & Iskra, G. (2006). The Concept of the Quality in Reference to High Schools. *Proceedings of the Conference: The Quality of Education in the Society of Knowledge*, UMCS, Lublin.

[3] Doroszewicz, S. & Kobyliska, A. (2006). The Model of the Quality of Education in the Main Trade School in the View of Students. *Proceedings of the Conference: The Quality of Education in the Society of Knowledge*, UMCS, Lublin.

[4] Roszak, M. (2008). Management of the Quality of Education. *Proceedings of the Conference: The Influence of Process Management on the Quality and Innovation of the Enterprise*. The Publishing House, UMCS Lublin.

[5] Hernas, A. & Szkliniarz, W. (2007). Experiences in the Sphere of Introduction of the Quality Assurance Systems of Education. *Proceedings of the 12th Seminar of Polish Material Association, Augustów*.

[6] Macukow, B. (2007). The Quality Assurance System of Education in the Warsaw Technical University. Comments. *The Publishing House The Warsaw Technical University, Warsaw*.