Scalable Honeypot Architecture for Identifying Malicious Network Activities

Gokul Kannan Sadasivam, Chittaranjan Hota
Department of Computer Science and Information Systems
BITS, Pilani - Hyderabad Campus, Hyderabad, India
{gokul, hota}@hyderabad.bits-pilani.ac.in

Abstract - Server honeypots are computer systems that hide in a network capturing attack packets. As the name goes, server honeypots are installed in server machines running a set of services. Enterprises and government organisations deploy these honeypots to know the extent of attacks on their network. Since, most of the recent attacks are advanced persistent attacks there is much research work going on in building better peripheral security measures. In this paper, the authors have deployed several honeypots in a virtualized environment to gather traces of malicious activities. The network infrastructure is resilient and provides much information about hacker’s activities. It is cost-effective and can be easily deployed in any organisation without specialized hardware.

Keywords - Honeypots, Server honeypots, Distributed honeypots, Dionaea, Glastopf, Kippo, J-Honeypot, HoneyD

I. INTRODUCTION

Server honeypots are services run in a physical machine for the sole purpose of capturing attacks and malwares. These honeypots take up unused IP address space of an enterprise network and continuously listen on all the malicious activities performed by a hacker. Mostly, these are capable of collecting a vast amount of automated attacks caused by botnets and other automated tools. Even though, there are a couple of commercial products; server honeypots are still evolving and are being researched widely.

Server honeypots are of two types: research honeypots and commercial honeypots. Research honeypots are used solely for research purpose in academia and research organisations. They are not designed to prevent attacks. Commercial honeypots find its place in the organisations. Organisations deploy honeypots to know the extent of attacks their systems are vulnerable to. It gives an awareness of the different types of attacks performed on an organisation.

Server honeypots can also be classified based on the level of interaction as low-interaction honeypots and high-interaction honeypots. Low-Interaction honeypots simulate well-known services, whereas, High-Interaction honeypots are real services running in a real operating system. This paper discusses about using low-interaction honeypots for efficient detection of attacks and malwares, as low-interaction honeypots give a reasonable amount of attacker’s activities within the network.

There are several low-interaction honeypots available for research work. Some of them are Dionaea, Glastopf, Kippo, HoneyD, and J-Honey. Each of these honeypots has a different architecture with the sole purpose of capturing attacks cum malwares. As mentioned before, these are low-interaction tools with minimal interaction with attackers. Nevertheless, these tools are efficient enough to understand about the nature of attacks on the internet.

The work done by Saurabh Chamotra et al. [1] discusses about deployment of low-interaction honeypots in an organisational private network. Moreover, HoneyD is the only low-interaction honeypot deployed in a network. Hamid Mohammazadeh [2] et al. discusses about the types of honeypots and placement of a virtual honeypot in the peripheral system. They argue the placement of honeypot to be behind or in-front of the firewall within an organisation's network.

Sanjeev Kumar et al. [3] propose a virtual network of honeypot environment. The honeypots employed are a mixture of low-interaction and high-interaction honeypots. They make use of Gen III architecture to capture malwares. Moreover, each virtual machine is assigned a different public IP address. The capture surface is increased by having more number of public IP addresses. Honeywall is deployed to prevent attacks; Snort is used to get attack statistics; Sebek is used to capture data from high-interaction honeypots.

Our work in this paper is similar to the above references in terms of some of the honeypot deployments. However, our major contribution in this work is the development of a novel distributed architecture that is resilient to outsider attacks. In our work, attack surface is increased by deploying honeypots in a distributed manner. In order to make system more versatile and efficient, only low-interaction honeypots are deployed in a virtual environment. Each and every honeypot captures attack packets specific to a set of ports. Hence, attack surface is boosted. It is efficient because of the low-interaction part of the honeypots. Low-interaction honeypots do not give full access to the operating system; therefore, one does not need to care about honeypots attacking other internet systems.

The motivation of our work in this research is to build a distributed architecture of honeypots that is resilient to failures. If any of the honeypot VM is brought down by an attacker, the system still works with the help of other honeypots. The VMs are segregated from one another. Even though a VM is entirely compromised it would not have sufficient access privileges to break into other VMs in the network. Lastly, the distributed architecture is scalable. The
The amount of traffic captured in a distributed manner is relatively high compared to individual honeypot systems [4], and the traffic increases with respect to the number of Virtual Machines without degrading the performance of these VMs.

The rest of the paper is organized as follows. Section 2 discusses the various low-interaction honeypots deployed in our test-bed. Section 3 presents the distributed honeypot system, and the results obtained using network capture tools. Section 4 is the summary of the work done. It also tells about the plans to expand the attack surface to capture severe malwares and attacks.

II. LOW-INTERACTION HONEYPOTS

A. Dionaea Honeypot

Dionaea [5] is a successor of Nepenthes honeypot, which is a low-interaction tool to collect malwares. Dionaea is built using python and C, like its predecessor, but additionally supports IPv6 protocol and TLS service. Dionaea uses many libraries to emulate several vulnerabilities [6]. Two of them are:

- vuln-sasserftpd (Sasser Worm FTP Server Buffer Overflow)
- vuln-openssl (Buffer Overruns in OpenSSL)

Dionaea simulates these vulnerabilities as services (FTP, MS SQL, HTTP, etc.). When an attacker exploits one of the services, his activities are captured and stored in MySQL database.

Compared to many of the low-interaction honeypots found in the market, Dionaea provides better services to trap attackers. Attackers are enticed to break into the system. Dionaea interacts with the attacker to make him believe that it is a real and complete service.

B. Glastopf Honeypot

Glastopf [7] is a low-interaction web honeypot written completely in python and primarily developed for research purpose. Glastopf listens on port 80 and port 8080 by capturing all HTTP and HTTPS traffic. It goes one step further in understanding the HTTP request by executing the request query and replying with an appropriate HTTP response to the attacker. The attacker gets a feel that it is a real system responding to requests.

Glastopf simulates various known web vulnerabilities, and it is easier to capture web attacks. The web attacks captured by Glastopf are local file inclusion attacks, remote file inclusion attacks, and SQL injection attacks. The primary advantage of Glastopf is to capture all types of web attacks. Unlike Dionaea, it does not listen on various ports.

C. HoneyD

HoneyD [8, 9] is a low-interaction honeypot developed by Niel Provos as a leisure project. HoneyD simulates various Operating Systems in a virtual network environment. The OS runs services to capture attack packets and log them accordingly. HoneyD calls each of its system as a personality engine, where the response packets are framed to come from a particular Operating System. This feature makes the attacker feel that the system is a real one.

There are numerous advantages in a HoneyD system. It avoids evasion by fingerprinting tools through the presence of different personality engines. HoneyD emulates many operating systems using these personality engines. HoneyD allows creation of virtual routers and virtual networks. HoneyD can also be connected to real physical machines to provide a complete interaction.

HoneyD interaction is constrained only to the TCP/IP layer. It simulates network stack of various operating systems like Windows, Linux and Mac OS X. It does not provide any application services. It also doesn’t provide a complete operating system.

D. Kippo Honeypot

Kippo [10] is a medium-interaction honeypot. It listens on port 22 (Secure Shell service) and gathers shell attacks like login attempts and remote file downloads. It provides a complete shell environment to the attacker to perform these attacks. It can be used also to record session interactions (hacking attempts) and obtain malware website URLs.

Initially, Kippo server establishes a TCP connection with the attacker. After negotiation of security algorithms, an attacker sends a login request. If the login username and password matches with the “userdb” file, then a successful command-line connection is established. Otherwise, a login failure message is sent back to the attacker. If the login is successful, the attacker continues executing various commands like ls, w, cd, echo, and so on in the server. Appropriate responses are given back by the server to the client machine.

A clever attacker can detect a Kippo system. Most of the attackers can perform “w” command to know the list of logged in users. A text file provides the list of logged users, which is static information. Hence, a smart attacker could know that the system is a fake one. Moreover, Kippo does not support many of the UNIX commands like cp, mv, “ls –l”, etc.

E. J-Honeypot

J-Honeypot is a low-interaction honeypot designed and developed by Yuging et al. [11] using Java. It has a web-based monitoring interface and a rule-based intrusion detection engine. It provides interfacing support to SQL database.

J-Honeypot listens on several of the ports configured in the system. If the SYN packet is sent to the port, it replies with an SYN-ACK to establish a connection that is similar to the way TCP connections are established. Once the remote system sends application layer data, J-Honeypot acknowledges and sends an RST to reset the connection. Now, the connection is closed.

Like all other low-interaction honeypots, it does not provide much interaction with the hacker. It does not help in gathering malware samples. The only information available is the network attack packets.
III. EXPERIMENTAL RESULTS

A. Distributed Honeypot Architecture

We deployed honeypots in a distributed architecture as shown in Fig. 1. A set of distributed honeypots run in a DELL Blade Server (Intel Xeon processor, 3GHz, 8GB RAM, 400 GB Hard Disk). The server machine runs Ubuntu 12.02 upon which a hypervisor exists. The hypervisor runs five virtual machines (all mini-Ubuntu 12.02) with each one of them running a different honeypot. The traffic to the virtual machines is controlled using a firewall and a NAT to facilitate building a DMZ with private IP addresses used for the machines. The server machine runs in the DMZ (De-Militarized Zone) situated at Information Security Laboratory of BITS, Pilani–Hyderabad campus network. Importantly, the server machine is listening on a public IP address.

![Fig. 1. Distributed Honeypot Architecture](image)

The management of the honeypots is a tedious task unless the honeypots are totally segregated from the real systems.

![Fig. 2. Management Console in the Information Security Lab](image)

In this architecture, the management console is placed in Local Area Network with sufficient firewall rules. Traffic can go from the research lab PC to the honeypots and vice-versa. Any other computer system in the LAN cannot access the server honeypot machine. This feature is implemented to prevent a possible attack on the internal university network. In Fig. 2, the management console is connected via several switches to the remote system deploying honeypots.

Each and every honeypot is running a couple of services. The services are chosen based on the architecture of the honeypot and its capability in collecting network traffic for a particular port. Table I shows the list of services for each honeypot.

![Table I. Open Ports in the Honeypot Architecture](image)

The honeypot system that was connected to Internet was active for a total of 21 days capturing packets day and night. Tcpdump tool was used to capture packets going to and from the DELL blade server. The amount of network traffic collected was nearly 100 Megabytes (in pcap format). There are approximately 6,70,000 packets in the pcap file. We analyzed these captures, and the following metrics were evaluated.

B. Geo-location of attackers

In total, approximately 4700 unique IP addresses were captured by the system. These IP addresses correspond to nearly 206 countries. A couple of online services [12, 13] provided geolocation information.

![Fig. 3. Geo-location of Attackers](image)

In general, an attacker can fake his IP address by using proxy servers. So, the results discussed below for geo-location is not accurate. But, it has similar statistics with the work done by Katerina Goseva-Popstojanova et. al.[14].

---

**TABLE I. OPEN PORTS IN THE HONEYPOT ARCHITECTURE**

<table>
<thead>
<tr>
<th>Honeypots</th>
<th>Open ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dionaea</td>
<td>21, 69, 445, 1433, 3306, 5060, 5061</td>
</tr>
<tr>
<td>Glastopf</td>
<td>80, 8080</td>
</tr>
<tr>
<td>Kippo</td>
<td>22</td>
</tr>
<tr>
<td>HoneyD</td>
<td>23, 25, 53, 110</td>
</tr>
<tr>
<td>J-Honey</td>
<td>135, 137, 138, 139, 1080, 1243, 12345, 12348, 27374, 31337</td>
</tr>
</tbody>
</table>

---

In this architecture, we deployed honeypots in a distributed structure as shown in Fig. 1. A set of distributed honeypots run in a DELL Blade Server (Intel Xeon processor, 3GHz, 8GB RAM, 400 GB Hard Disk) with a running Ubuntu 12.02. The hypervisor runs five virtual machines (all mini-Ubuntu 12.02) with each one of them running a different honeypot. The traffic to the virtual machines is controlled using a firewall and a NAT to facilitate building a DMZ with private IP addresses used for the machines. The server machine runs in the DMZ (De-Militarized Zone) situated at Information Security Laboratory of BITS, Pilani–Hyderabad campus network. Importantly, the server machine is listening on a public IP address.

The management of the honeypots is a tedious task unless the honeypots are totally segregated from the real systems.

In this architecture, the management console is placed in Local Area Network with sufficient firewall rules. Traffic can go from the research lab PC to the honeypots and vice-versa. Any other computer system in the LAN cannot access the server honeypot machine. This feature is implemented to prevent a possible attack on the internal university network. In Fig. 2, the management console is connected via several switches to the remote system deploying honeypots.

Each and every honeypot is running a couple of services. The services are chosen based on the architecture of the honeypot and its capability in collecting network traffic for a particular port. Table I shows the list of services for each honeypot.

The honeypot system that was connected to Internet was active for a total of 21 days capturing packets day and night. Tcpdump tool was used to capture packets going to and from the DELL blade server. The amount of network traffic collected was nearly 100 Megabytes (in pcap format). There are approximately 6,70,000 packets in the pcap file. We analyzed these captures, and the following metrics were evaluated.

B. Geo-location of attackers

In total, approximately 4700 unique IP addresses were captured by the system. These IP addresses correspond to nearly 206 countries. A couple of online services [12, 13] provided geolocation information.

In general, an attacker can fake his IP address by using proxy servers. So, the results discussed below for geo-location is not accurate. But, it has similar statistics with the work done by Katerina Goseva-Popstojanova et. al.[14].
Most of the attackers (approximately 40%) belong to the province of China, as shown in the Fig. 3. Next to China, most of the attackers (20%) came from United States. In general, many of the attackers are from Asia (China, Taiwan, Korea, Hong Kong, Japan and India). Within “Others”, there were nearly 200 countries that had attackers less than 1% of the total attackers.

C. Most Wanted Services

Fig. 4 illustrates the total number of sessions for a couple of services. As seen in Fig.4, services like SSH, MySQL, MSSQL and Telnet are of great interest to attackers. The reason might be because of the login services provided by these ports. Attackers try to login to the system trying to get complete access.

Fig. 4. Ports vs. Number of sessions

We observed that SSH is the most affected service with an approximate 11,000 sessions. As mentioned in the previous paragraph, services like MySQL and MS SQL are of high interest. Telnet service also provides remote access and comes in the fourth place. Even HTTP service is of some interest to the hackers. In Fig.4, “OTHERS” refer to several other ports that the system was listening to.

The system was running on some malware ports as given in Table II, to check the occurrence of old malwares in the Internet. The malwares (SubSeven, NetBus, Back Orifice) are Remote Administration Tools/Trojans (RAT). RAT tends to install a server program in a vulnerable computer, so that, an attacker can use a client program to command the server. However, none of the honeypots captured any traffic meaning that these Trojans are not used anymore by hackers.

<table>
<thead>
<tr>
<th>Malware</th>
<th>Port No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubSeven Trojan</td>
<td>1243</td>
</tr>
<tr>
<td>NetBus Trojan</td>
<td>12345</td>
</tr>
<tr>
<td>BioNet Trojan</td>
<td>12348</td>
</tr>
<tr>
<td>SubSeven Trojan</td>
<td>27374</td>
</tr>
<tr>
<td>Back Orifice Backdoor</td>
<td>31337</td>
</tr>
</tbody>
</table>

Something that is of interest to the hackers is the “PROBE PACKETS” as discovering vulnerabilities become easier using these packets. There were nearly 2700 sessions attempting different port numbers that were closed by distributed honeypots. These are probe packets used to check the availability of a port.

The distributed honeypot didn’t run any services in the ports listed in Table III. But, there were several packets to these ports. RDP stands for Remote Desktop Protocol. With a remote login service attackers would be provided with a GUI of the remote server on their desktop. This is more convenient to execute attacks. In order to hide themselves, attackers connect to proxy servers like squid service which runs in port 3128.

Table III. Ports of Interest to Port Scanners

<table>
<thead>
<tr>
<th>Port No.</th>
<th>No. of Sessions</th>
<th>Service Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>3389</td>
<td>315</td>
<td>RDP</td>
</tr>
<tr>
<td>443</td>
<td>171</td>
<td>HTTPS</td>
</tr>
<tr>
<td>3128</td>
<td>165</td>
<td>squid</td>
</tr>
</tbody>
</table>

D. SSH Interaction

Since, most of the sessions are encrypted using public key encryption algorithms, authors have employed Kippo log files to analyse the shell interactions. Fig.5 portrays the command-line interaction with one of the attackers. As shown in the Fig.5, the attacker tries to stop the firewall so that he has full control over the network services. Then, he downloads .32 non-executable code from 23.95.34.215. The IP address corresponds to host.colocrossing.com, which is a cloud-based web hosting server. Finally, the hacker executes .32 file in background mode. Also, he makes sure that the code runs during every system boot-up.

```
root@hosta:~$ service iptables stop
bash: service: command not found
root@hosta:~$ root@hosta:~$ wget http://23.95.34.215:8080/.32
Sorry, SSL not supported in this release
root@hosta:~$ root@hosta:~$ chmod u+x .32
chmod: cannot access .32: No such file or directory
root@hosta:~$ root@hosta:~$ bash: /./.32: command not found
root@hosta:~$ root@hosta:~$ cd /tmp
root@hosta:~$ root@hosta:~$ tmp
root@hosta:~$ root@hosta:~$ echo `cd /root`>/etc/rc.local
cd /root>/etc/rc.local
cd /root>/etc/rc.local
cd /root>/etc/rc.local
root@hosta:~$ root@hosta:~$ root@hosta:~$ root@hosta:~$ root@hosta:~$
```

Fig. 5. SSH Interaction of an attacker

E. SSH Session Statistics

As shown in Table IV, nearly 10,500 unique login attempts were made to the Kippo server. A single login attempt is a combination of a username and a password. Only 304 different attackers out of 4700 attackers have made these attacks. The login attempts should not be confused with 11,000 sessions shown previously. In a given TCP stream, multiple login attempts are possible. In the 11,000 SSH
conversations, some of them did not make a login attempt. These streams had a maximum of two to three packets.

<table>
<thead>
<tr>
<th>TABLE IV. KIPPO STATS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Login Attempts</td>
</tr>
<tr>
<td>Unique Usernames</td>
</tr>
<tr>
<td>Unique Passwords</td>
</tr>
<tr>
<td>Unique sources</td>
</tr>
</tbody>
</table>

In general, most of the attackers tried “root” as the username with different passwords. Nearly 48,000 login attempts had “root” as its username and nearly 14,000 had “admin” as its username. Fig.6 displays the count of different usernames attempted on the system. Also, approximately 1300 login attempts had “admin” as the password.

![Username count](image)

Table V displays the list of usernames and passwords configured in the Kippo system. If any attacker tries anyone of the combination, then he will be successfully authenticated to proceed further. If the username/password is wrong, then a login failure message is displayed, and the attacker is prompted to re-enter the login credentials.

<table>
<thead>
<tr>
<th>TABLE V. USERNAME-PASSWORD CONFIGURED IN THE KIPPO SYSTEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>root</td>
</tr>
<tr>
<td>root</td>
</tr>
<tr>
<td>root</td>
</tr>
<tr>
<td>root</td>
</tr>
</tbody>
</table>

Finally, analysis was done on the different combination of usernames and passwords used by the attackers. Attackers frequently used “root” or “admin” as the username and, “admin” or “123456” as the password. In fact, the honeypot is configured to authenticate “root” username and “123456” password successfully (as shown in Table 5). The Table 6 illustrates the Top 5 combinations of a username and a password.

<table>
<thead>
<tr>
<th>TABLE VI. USERNAME-PASSWORD COUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Username/Password</td>
</tr>
<tr>
<td>-------------------</td>
</tr>
<tr>
<td>root / admin</td>
</tr>
<tr>
<td>admin / admin</td>
</tr>
<tr>
<td>root / 123456</td>
</tr>
<tr>
<td>admin / 123456</td>
</tr>
<tr>
<td>admin / admin123</td>
</tr>
</tbody>
</table>

IV. CONCLUSION

In this work, we deployed distributed honeypots to capture malicious network traffic and malwares. Our experimental results showed that most of the malicious activities were brute force logging into the service to access the system. Malwares captured using this mechanism was minimal. The distributed architecture is well suited for organisations to know the effect of attacks that are on their network. The future work would include state machine models to analyse the behaviour of attackers. Also, we will explore the methods to classify these network attacks in real-time.

V. REFERENCES