Project Report

Honeypots

16-05-2011

Rohan Kulkarni
Shashank Sahni

Under the guidance of
Saurabh Barjatiya

AIM

This project aims to set up a honeypot to study and gather various attack patterns used by crackers to compromise a machine. This will help us realize the techniques and methodologies used by attackers and the steps they take to make the machine part of their compromised network.

Motivation

We always hear systems of an organization or individual being compromised. Some of them are focused attacks and target unknown vulnerabilities, but a majority of them are bots looking for known flaws and loopholes - searching and attacking systems with payloads, burte-forcing passwords etc. Once the system is compramised, it becomes a part of their botnet.

We wanted to monitor the latter kind of attacks.a Through a honeypot we can learn various techniques and attack vectors involved in building a botnet.

Introduction

A honeypot is a trap for cyber attackers. Its a system, built carefully to attract them and monitor their moves. As part of the project, we decided to deploy a linux based high interaction honeypot to study the attack patterns on its ssh server.

The resulting information contains

- IPs of malicious systems which attacked the honeypot
- user names and corresponding passwords attempted by attackers,
- steps taken by them once they break into the system,
- rootkits and softwares used by them,
- servers they were contacting.

List of available resources

For the deployment of the honeypot, we decided to go for Virtualization.

- This is both flexible and convenient.
- Everything could be taken care of in a single machine.
- Backups and snapshots are possible.
- Small networks can be easily setup between virtual machines.

We chose KVM hypervisor on Ubuntu 10.04 as the virtualization solution and Ubuntu 8.04 server as our virtual machine.

Problems Faced and adopted solutions

Challenge 1

For information gathering through ssh server, the logs easily provide the usernames and IP attempting to log in. But the attempted passwords are never logged. In order to get this information, we patched the ssh server on our linux based honeypot such that it logs passwords.

Patching SSH server

We downloaded the ssh source code - <version> and changed the auth_passwd.c file. We added the following to the auth_passwd() function in that file, just before its return statement.

```
logit("Password used %s for user %s", password, authctxt->user);
```

Then we compiled the code and installed the patched ssh server.

```
[shell]$ sudo apt-get build-dep openssh-server
[shell]$ ./configure --with-privsep-user=<user>
[shell]$ make
[shell]$ make install
```

Challenge 2

The other challenge was to be able of monitor the activities of an attacker once he is in the system. We took two approaches to make it work.

1) Using Script command to save a typescript of the attacker's shell session

This tool creates a typescript of the shell session. The output is not just a set of commands, but the exact impression of shell as it would have appeared to the attacker.

We added the script commad in the ~/.bash_profile of every user, so that it doesn't matter how he logs in, the session will always be saved. The following lines were added.

```
[shell]$ script -q -a .session_file
exit
```

As a result, as soon as someone logs in script session starts and the output is saved to the *.session_file*. The "-q" option is used to prevent script from printing the "script start" & "script end" messages. "exit" is used to automatically end the parent shell when script exists. This way

it was hard for an attacker to identify the command.

Because of the sophisticated techniques developed recently to monitor malware activities, an attacker might miss simple traps. But this technique will fail if the attacker uses his own shell. We'll see a little later how an attacker managed to do so.

2) Using Sebek to log all the keystrokes of an attacker

Sebek is a kernel module which provides extensive options for designing a honeypot. Its prime features are:

- Its able to log all the keystrokes of an attacker. This feature works even in an encrypted environment.
- A sebek daemon/server on the other system receives and displays all the messages.
- It can create a copy of all the tools/data copied over ssh/scp by the attacker. This is very useful since this way we can get our hand on the actual rootkits or tools used by an attacker.

Installing Sebek on Ubuntu 8.04, Hardy Heron

Sebek project is not under development currently and hence due to lack of support for newer kernels we decided to deploy it on old but yet supported kernels. Ubuntu 8.04, released in April 2008, is a "Long Term Support" release and hence is a good choice for deploying a server. Here are the steps taken to install Sebek on Ubuntu 8.04.

1) In order to install the sebek client on the server, we'll have to compile its source for the appropriate kernel. We can download the source from the project's website

```
[shell]$ svn co https://projects.honeynet.org/svn/sebek/linux-2.6/trunk/
sebek
```

2) Now, go to the src directory and run

```
[shell]$ ./configure --disable-raw-socket-replacement
```

3) Install all the required packages until ./configure stops prompting and returns success. Now, run make to finally compile the code

```
[shell]$ make
```

4) After successful compilation, the file "sebek-lin26-3.2.0b-bin.tar.gz" will be created in the sebek folder. Extract the file.

```
[shell]$ tar -xvzf sebek-lin26-3.2.0b-bin.tar.gz
```

5) Go to the directory sebek-lin26-3.2.0b-bin/ and open the script sbk_install.sh. Fill up all the required parameters corresponding to the client machine, i.e. Ubuntu 8.04 server. The parameters and their meaning are.

a. **Interface** - Mention the interface which sebek will use to send the messages.
a. **Destination Mac Addr** - Set the destination MAC for the sebek packets. If the sebek daemon is running on the LAN then use its MAC to be the destination addr.
b. **Source and Destination UDP ports -** Mention the src and dest. port to be used for sending sebek packets. The desktination will be used at the other end(daemon) to catch and parse the packets.
c. **Magic Value -** Its just a way to identify honeypots of the same group by sebek(both client and daemon).
d. **Keystroke and Socket tracking -** This lets sebek know if you are interested in only keystroke tracking or want to track the network connections too.
e. **write tracking -** This lets you monitor all the write activities on the client machine.
f. **testing -** This is set when we are dealing with

6) Now, run the script sbk_install.sh as root. You are good to go, if it says installed successfully.

```
[shell]$ ./sbk_install.sh
```

Installing Sebekd(server) on another machine

Extract the compressed sebekd file and run the following commands to install it on your system.

```
[shell]$ ./configure
[shell]$ make
[shell]$ make install
```

Sebekd will install three tools on your machine

1. sebek_extract - This is used to sniff over the network on a port and extract information from sebek packets.
2. sebek_ks_log.pl - This is a perl script designed to display the keystroke information from the information extracted by sebek_extract. This is used for viewing the data in real time.
3. sebek_upload.pl - This script is used to save the information retrieved in a mysql database. This way we can view the data later.

Once the honeypot is up we can use the tools in sebekd to sniff the sebek packets and extract the information from the headers.

```
[shell]$ sbk_extract -i eth0 -p 1413 | sbk_ks_log.pl
```

If an attacker somehow logs in to the machine, all his activities(keystrokes) will be sent to the server machine in real time and hence we'll be able to monitor everything.

Challenge 3

The next challenge was to keep track of all the communication going to and from the honeypot. This way we can log all the IPs that try to communicate or scan the honeypot. The ports they are attacking etc. This is where we used a CentOS based customized distro - Honeywall.

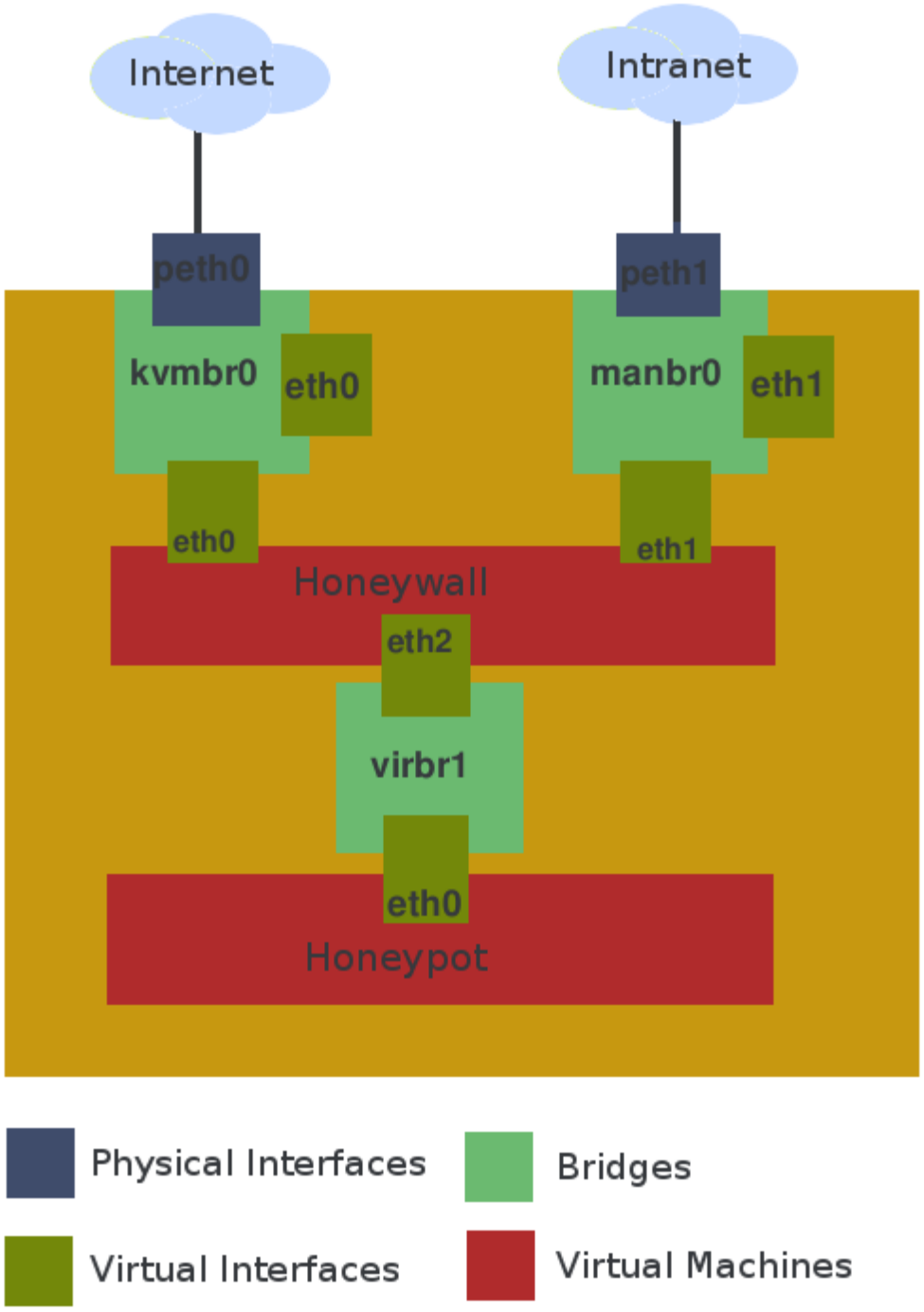
Using Honeywall to log all the traffic going to and from the honeypot

Honeywall is a CentOS based distribution built to capture and analyze extensive data collected from honeypot(s). Some of its features are:

- It is setup between the honeypot(s) and the Internet. This way it can capture and monitor all the traffic coming through and going out.
- It has a web-based interface named Walleye which can be used for administration, data analysis and system configuration.
- It supports sebek 3.*. This way we can also log all the activities inside the honeypot too.

Installation and setup of honeywall

A typical honeywall installation requires three interfaces - one connected to Internet, other to the honeypots and the third one is the management interface which is used by us to access and configure it. Please note that only the management interface gets the IP.



In order to install Honeywall all we have to do is boot the ISO. It automatically takes care of the rest. Once it is installed, we'll changed the dom0's(ubuntu 10.04) network configuration to make it work. Here is how the configuration looks like.

In order to configure the honeywall, we need to first log into the system.

```
username - roo
password - honey
```

Root has the same password. As soon as we log in as root, we'll see the a menu which can be used to configure honeywall as per our needs. The official guide for the configuration can be directly referred to complete this configuration.

At this point(once honeywall is configured),

- the two interfaces are in the bridged configuration, relaying packets between Internet and Honeypot to and fro,
- the third interface should have an IP to be able to connect to the internal network,
- Walleye should be working too. Visit the URL **`https://<managementIP>`** to access the web interface. Once again the credentials are roo and honey.

Challenge 4

The next challenge was to allow access to the attacker. Our earlier strategy was to learn different passwords tried by attackers and then set the most common as the one for all general users - test, guest, root, staff etc. This doesn't seem to work either because the stats show that honeypot was attacked by different sets of IPs everytime. Probably different bots and thats why they were never used the same sets of passwords again.

As a result, we decided to make a few changes in the strategy. We decided to patch the ssh server such that it allows access after maximum number of tries.

Open the file auth_passwd and add the following lines at the end of the function auth_password(), right before its final return call.

```
if (authctxt->failures >= options.max_authtries/2)
     return 1;
```

This didn't work either. That's because the attacker start a new session for every attack, hence max_authtries parameter doesn't affect anything.

Therefore, we decided to just allow them access and monitor what do they do once they have the access. We patched the code to authenticate for any password.

In the above code, instead of adding an if statement, just set the final return call of auth_password() to return 1. This worked pretty well and we got some very useful information from an attacker who logged in and tried to setup his territory.

Results from Honeypot

IPs that attacked the ssh server

IPs	Number of Attacks
113.106.18.238	1
125.88.128.158	1
116.125.70.150	1
58.64.167.156	2
221.143.48.7	3
218.14.203.205	7
202.137.6.69	9
81.92.157.210	12
140.113.173.171	13
221.2.163.252	14
187.60.35.139	20
60.211.179.188	32
67.205.76.176	37
113.105.167.70	40
112.65.246.75	50
58.211.5.178	69
125.223.205.22	91
46.45.147.173	525
112.140.186.138	3768

More than 8000 log in attempts were made. The most common attempts include. Rest of the thousands included regular names of people.

Users	Attempted logins
root	1247
user	565
test	32
admin	21
mysql	13
guest	12

Most command passwords attempted are.

Passwords	Number of Attempts
password	351
123456	292
qwerty	88
12345	31
root123	12

Rest of the passwords were various combination of words and numbers.

These were the attempted attacks, but while monitoring the honeypot we also came across situations where authentication was granted. Most of the users just exited after log in probably realizing that its a honeypot, but one of them stayed and tried to setup a bot on this machine.

Through the “last” command we found out that someone was logged in as root and a few other user named ale and alee which never existed. After checking the /etc/passwd we came to know that three new users were created - ale, alee & cri. The interesting thing was that he made the users ale and alee with identity of root. Their uid and gid were 0.

Here is the commands he used(from .bash_history).

```
[shell]$ /usr/sbin/useraddd -u 0 -g 0 -o ale
[shell]$ /usr/sbin/useraddd -u 0 -g 0 -o alee
[shell]$ /usr/sbin/useraddd cri
```

The history also revealed that he scouted for some information regarding the system at first. These were his first commands.

```
[shell]$ uname -a
[shell]$ cat /proc/cpuinfo
```

Later he deleted the home directories of the created users.

Then we tried to check if the user has tried to communicate with any remote server. We checked for listening ports and existing connections and found that a single application(qmail-clean) was acting as both a backdoor(port 3303) and was connected to a remote IRC server(ircu.atw.hu).

```
[shell]$ netstat -A inet -l  # for listening ports
[shell]$ netstat -A inet   # for existing connections
```

As a result, we tried to locate the file, but it didn't return any result.

```
[shell]$ locate qmail
[shell]$ #nothing
```

Then we checked for qmail-clean was accessing.

```
[shell]$ lsof -p <pid>
```

This told us that the application was accessing files in /tmp/.BASH folder. After opening it we realized that there was a zipped folder in it named "psyRomana.tgz".

After analysing this software a bit we realized that it provided a backdoor to the attacker with probably a separate shell to escape any monitoring on bash etc.

Things Learned

We started our work on honeypots by going through the literature available on this topic - almost all papers available on honeynet.org, sections from the book "virtual honeypots: from botnet tracking to intrusion detection". It helped us understand the earlier and most recent methods of honeypot deployment. This is how we gathered all the possible scenarios and decided to go with

While working with Sebek we got a good understanding of the internals of hypervisor based monitoring techniques.

Once we started with the deployment part, configuring systems was a challenge too and help us learned a lot of new things.

- Bridge configuration in linux and integrating it with KVM.
- This was the first time we used KVM and qemu.
- We tweaked a lot in the code of ssh-server and now have a good understanding of its code base.
- Logs handing in linux - syslog & rsyslog.

When the deployment was over and we made the system live, we came across a lot of new

things.

- While making various changes to the ssh server we realized that many attackers were creating a new session or every password attempt. This way increasing or decreasing the value of max authtries had no affect on them. We even saw a growth rate in the attacks by increasing the value of max unauthorized sessions.
- The attackers weren't brute forcing blindly. They were using the most common set of passwords used by novice users and admins. The data shows that most of the attacks were on common accounts - root, test, guest etc.
- Once we finally got an attacker in the machine, he did just the right amount of work on native bash to come up with a program which provides its own backdoor. This way we weren't able to log any more events.
- While studying their software, we got a good understanding of the workings of botnets. The application acts both a server and a client. It gathers commands from a remote IRC server and lets you directly access the system through a backdoor.
- With we ssh server patched to accept any password for a valid user, many of the attackers became suspicious and just exited as soon as they logged in. Since, then we didn't hear from those IPs again. Probably a black listing of some kind. We shouldn't have made it so obvious but it did helped us learn a lot about the attackers tactics.

Future work

As part of this project, our main focus was on setting up and gathering information through a high-interaction linux based honeypot running a ssh server. This was a very focused and narrow scenario. Things that we can further work on are:

- Instead of going for a complete OS, we can go for low interaction honeypots which are usually application(service) emulators which gather all sorts of information and attacks done on the service. e.g. - HoneyC etc.
- Along with Linux we an setup Windows and study the attack patterns for this operating system, e.g. gebek.
- We can install servers with known vulnerabilities and can monitor attacks targeting such vulnerabilities.
- We can dedicate some resources and set up machines to become a part of distributed honeynets, e.g.- Distributed web honeypots etc.

If the work is carried on, we'll begin with analyzing the mechanisms used by bots to communicate to each other, their commands, purpose/goal etc.

We already witnessed a botnet attack on the ssh server. Analyzing the code could give us vital insight into the internal workings of botnets.

Conclusion

This project reveals how machines are constantly under attack on a public network. We can of course secure out machine by keeping the software up to date but in order to come up with better strategies to fight attackers, we need to learn their moves and neutralize them. Gather and analysing such information is possible through honeypots.

This project targeted ssh server attacks but there are lot of windows in to a system which gives a malicious user more surface to attack. By setting up honeypots studying those domains could help gain more insight into the attackers methodologies and help in the development of strategies to prevent them.