Project 02 Summary and Writeup

Interview

Kevin: What does your company do?

Herb: We run virtual machines for companies.

Kevin: Do you have any resident linux forensics experts? Or anyone else who may have already run a preliminary analysis?

Herb: No.

Kevin: So why was I called?

Herb: A customer reported sketchy stuff on their VM.

Kevin: What kind of behavior was sketchy?

Herb: They said they saw some suspicious processes, but I don't think there's anything wrong with the system.

Kevin: When did this first start happening?

Herb: It was a while ago. We took the machine offline over a year ago, but now there's a government audit that's requiring us to investigate the system.

Kevin: Any idea why that audit is required?

Herb: No idea at all.

Kevin: Let's talk a bit more about the system. What type of OS does it run?

Herb: It's an Ubuntu 14, X86 architecture.

Kevin: Who has accounts on the system?

Herb: Just IT and the customer.

Kevin: Account names on the system?

Herb: Um… I don't know. I know the customer's name was Tim.

Kevin: Any other information you think I need to know?

Herb: Umm… the timestamps might have been altered with. I don't know why or how but timestamps are off.

Kevin: Awesome. Thanks Herb.

Herb: You're welcome!

Forensics Investigation

Scenario 1: A **rootkit** was installed on the system

Dead analysis revealed the existence of a possible rootkit called "Lampiao". I first noticed lampiao.sh when I ran the "linux_pslist" command. Running the "strings" command against lampiao.sh revealed it killed whatever was running on port 80 and piped the contents of the lampiao.txt file to port 80 on the local host. The

contents of lampiao.sh and lampiao.txt can be found at etc/cangaco/lampiao.txt and etc/cangaco/lampiao.sh. No other indicators of malicious behavior were found.

I will say that this is not a very sophisticated rootkit. It does not escalate privileges, access logs, or do any functions similar to other rootkits we've analyzed. Nevertheless, it does kill processes on port 80 and pipes the lampiao.txt file to port 80, so it is a rootkit in the sense that it changes how the system processes. You can see in the log/upstart/lampiao.log directory (enumerated in MySQL using "select * from files order by accessdate desc, accesstime desc") that processes are being killed on port 80, signal errors are being redirected to stdout and it is catting the contents of /etc/cangaco/lampiao.txt.

Scenario 2: A **keylogger** was installed on the system

I had originally ruled out evidence of a keylogger when the commands "linux_keyboard_notifiers" and "linux_check_tty.txt" returned nothing. But then I found PID 361 /run/logkeys.upload.pid in the linux_enumerate_files.txt, which indicates that a keylogger is running on the system. In addition, the linux_bash.txt file indicates that the user was trying to test the keylogger's functionality by echoing certain phrases and then catting the var/log/logkeys.log file to verify the keylogging.

Scenario 3: A **reverse shell** was installed on the system

The "linux_bash" command and the gethistories.sh command revealed that something called "revshell7777.bin" was moved to the /usr/sbin directory. It was given a new name: "updatephp". Updatephp is seen running under PID 24913 according to linux_pslist and linux_bash_env. According to "bash_histories.txt", the revshell7777.bin was also made executable in both sbin and the cron directories. Running "strings updatephp | grep "7777"" reveals that there is information being sent from port 7777 on the local host. This indicates that information is being taken from the compromised system and being exfiltrated (this is applicable to ***Scenario 8*** as well).

Scenario 4: **ext4 extended attributes** are being used to exfiltrate information, or re-compromise the system on startup

For this, all I was able to find was evidence from the linux_bash command of a line /etc/cron.daily/update-php. It seems that the attacker had primed the obfuscated revshell7777 malware to run everyday. This is actually a smart obfuscation because it runs the malware everyday, but disguises it as something that wouldn't raise suspicion. This would mean that the system would be constantly reinfected each day, even if the process running the malware had been killed the day before. While this is not something that occurs on startup, it is essentially achieving the same

Scenario 5: A **new account** was created on the system **with root privileges**

There is no obvious indication that a user was created according to the bash history. Nevertheless, the interview with Herb indicated that only two accounts existed on the system: root and user. Herb said the customer's name was Tim. Interestingly, according to etc/passwd (and its enumeration in MySQL as seen with users.txt), there are two user accounts on the system: Tiago and Benji. This indicates that another user was added at some point in time. Tiago most likely was the original user because their uid is 1000, while Benji is 1001. In addition, the get-logins.sh script revealed that Thiago logged in years before the first appearance of Benji in January 2020. This makes me suspect that Benji was created by whoever infiltrated the system.

Scenario 6: An **existing system account** was **modified to allow interactive logins**

According to users.txt, the user libuuid allows interactive logins. This account normally has the "no login" tag so it is I wasn't able to load the users into the MySQL database because libuuid was malformed, so I had to remove it from the etc/passwd file to get the data to load into MySQL. Furthermore the account "proxy" is another suspicious user because it points to bin/bash when that's not its default directory to point to. This means someone could execute remote commands from that account as well.

Scenario 7: The system's **network was hijacked** to route traffic through a monitored interface

Yes. There is evidence of traffic monitoring on the system. The command "linux_netstat" revealed two suspicious activities where traffic is being monitored.

TCP 192.168.56.102 :10002 192.168.56.1 :35551 ESTABLISHED SnwjT/13311
TCP 192.168.56.102 :50610 192.168.56.1 : 4433 ESTABLISHED SnwjT/13311
UNIX 105849 SnwjT/13311

TCP 192.168.56.102 :10002 192.168.56.1 :35551 ESTABLISHED PiMPz/4544
TCP 192.168.56.102 :50600 192.168.56.1 : 4433 ESTABLISHED PiMPz/4544
UNIX 105849 PiMPz/4544

Further evidence and examination of these files in the /tmp directory did not reveal their functionality, but they are not native to the system and thus were installed by whoever compromised the system. Exactly what their true purpose is cannot be divined from the analysis. Nevertheless, the proc_map command revealed that both of these are both executable and writable, which means an attacker can use them to great effect.

Scenario 8: **Software that supports file exfiltration or attack obfuscation** was installed on the target machine after it was compromised

In the interview, Herb had mentioned that the timestamps may have been tampered with or changed. The only evidence of this I see is in the timeline I created from the create-timeline.sh script. In that file I discovered that Freedesktop may be running to change system timestamps in the timeline. Now, Freedesktop is not inherently a malicious program and it is standard with Linux distros. Nevertheless, it could be used maliciously to change timestamps, which is what I believe I'm seeing in the timeline.

In addition, as I mentioned above, there is evidence from strings updatephp that files may be exfiltrated from the system. If I run "strings updatephp | grep "file"", I can see several functions in the binary that suggest file exfiltration may be occurring. There is even "Transferred a partial file", "elf_file_get_em", "sendfile.c", and many, many more. I was unable to run dynamic analysis on updatephp. I kept getting a 126 error code when I tried to run it in gdp. So I can't verify what these functions do exactly, but the names seem self-explanatory enough to merit genuine concern.

Scenario 9: **Suspicious volumes** are mounted on the system that are used by the attacker

No. The “linux_mount” command revealed nothing suspicious was mounted to the system. This would make sense because the workstation was used to host a webserver, so most exploitation would most likely occur in the network interface. In addition, a manual exploration of the mounted volumes revealed no suspicious activity.

Scenario 10: The system's **kernel contains trojaned modules**

No evidence of trojaned modules was found according to the “linux_check_modules” command. In addition, it seems unlikely that the attacker would exploit the kernel because it was a webserver. I imagine this why I’ve seen more activity in the network rather than on the system itself.

Scenario 11: **Existing executable** files on the system have been **replaced with malware versions**

Yes. The updatephp binary in the usr/sbin directory is malware. I first learned this because when I tried to send it via email, it was flagged as malware by Gmail and Office. This is not surprising given we saw the revshell7777.bin file moved to that directory and renamed updatephp. This was clearly a way for the attacker to obfuscate the binary. I did a static analysis of the binary using objdump (full command shown in the commands section below), and there is a lot going on. There are dozens of functions and it looks like this is a very sophisticated piece of malware. Included in the functions are commands to read files, directories, and other information on the system.

Beyond this obvious malicious file, I compared some common md5hashes to team cyru using whois -h team.cymru.com “hash” and did not find any hashes on their database. Now, this does not guarantee that no binaries have been replaced with malware, but there is no current evidence to suggest any malware exists on the system other than the revshell7777 binary and the PiMPz and SnwjT processes running on the ports.

Scenario 12: **A dictionary attack was performed** against the system so that the attacker could determine a user's password

The failed logins retrieved from the get-logins.sh script reveal some failed login attempts, but not enough to suggest a dictionary attack was performed against the system. There are only 10 failed logins, so unless the user had an incredibly weak password, then it's safe to assume that no dictionary attack was performed. In addition, no evidence of a password hacking program - like John the Ripper or another brute force program - was found in the bash history or in the process lists. It seems the system was compromised another way.

Scenario 13: **A shared credential attack was performed** to elevate the privileges of a process on the system

The only evidence I found of any shared credentials was between two processes according to linux_check_creds. According to the output, the processes 696 and 26 were sharing credentials, but I didn't learn any additional information using linux_proc_maps on each process. It may be evidence of a shared credential attack, but I can't say for certain that it was used by an attacker.

Scenario 14: A **network card** on the system has been **put into promiscuous mode** for packet sniffing

Neither of the interfaces are in promiscuous mode according to the "linux_ifconfig" command. I did not see any sign of promiscuous mode in "linux_arp" either. It is more likely that the suspicious processes found running on the network in ***Scenario 7*** are up to no good.

Summary of Findings

The attacker gained access when they exploited a vulnerability in the system to install a keylogger. The attacker most likely retrieved the user password from this keylogger and used it to access the system remotely with root privileges. I do not suspect a dictionary attack was used at this time.

The attacker most likely created the account "benji", and used benji to escalate the privileges of other accounts in the system.

In addition, the attacker installed a rootkit called "lampiao". This rootkit, as mentioned above, seems to kill processes running on port 80 and pipes the contents of the lampiao.txt file to that port before sleeping for 10 seconds.

There is evidence that other directories were installed on the system to collect network data and perhaps other information. PiMPz and SnwjT are currently redirecting network traffic based on the information found from the analysis. It should be assumed that these processes were collecting information on the system or the database server and passing it to the IP address shown in the linux_netstat and linux_arp commands.

Recommendations

Urgent: Disconnect the system from your network.

Urgent: Either dispose of the compromised system or recover it via OS reinstallation.

Recommended: See if other systems have been compromised and follow the above steps.

Recommended: Institute new policies to regularly check for compromised machines.

Recommended: Implement new policy to investigate customer complaints instead of dismissing them

Recommended: Periodic review of all system logs, including those used by customers, such as MySQL.

Recommended: Install intrusion detection system on the webserver; also require customers to install these intrusion detection systems as part of the TOS for your company

Commands Run for Project 02:

Memory Analysis

p2vol linux_pslist | tee linux_pslist.txt
p2vol linux_psaux | tee linux_psaux.txt
p2vol linux_pstree | tee linux_pstree.txt
p2vol linux_psenv | tee linux_psenv.txt (ran also for processes
p2vol linux_psxview | tee linux_psxview.txt
p2vol linux_proc_maps | tee linux_proc_maps.txt
p2vol linux_bash | tee linux_bash.txt
p2vol linux_bash_env | tee linux_bash_env.txt
p2vol linux_bash_hash | tee linux_bash_hash.txt
p2vol linux_check_afinfo | tee linux_check_afinfo.txt
p2vol linux_check_creds | tee linux_check_creds.txt
p2vol llinux_check_fop | tee linux_check_fop.txt
p2vol linux_check_idt | tee linux_check_idt.txt
p2vol linux_check_inline_kernel | tee linux_check_inline_kernel.txt
p2vol linux_check_modules | tee linux_check_modules.txt
p2vol linux_check_syscall | tee linux_check_syscall.txt
p2vol linux_check_tty | tee linux_check_tty.txt
p2vol linux_keyboard_notifiers | tee linux_keyboard_notifiers.txt
p2vol linux_ifconfig | tee linux_ifconfig.txt
p2vol linux_netstat | tee linux_netstat.txt
p2vol linux_netfilter | tee linux_netfilter.txt
p2vol linux_arp | tee linux_arp.txt
p2vol linux_list_raw | tee linux_list_raw.txt
p2vol linux_mount | tee linux_mount.txt
p2vol linux_enumerate_files | tee linux_enumerate_files.txt
p2vol linux_malfind | tee linux_malfind.txt

Analyzing Mounted Images

sudo ./getmacs.sh /media/part0 > p2.csv

mysql -u root -p create case-p2

mysql -u root -p
connect case-p2
sudo cp c2.csv /var/lib/mysql-files

{added tables and loaded files as shown in Lecture 11}

sudo cp /media/part0/etc/passwd /var/lib/mysql-files/p2-passwd
sudo cp /media/part0/etc/group /var/lib/mysql-files/p2-group

tee users.txt
select * from users order by uid;
tee distinct_users.txt
Select distinct uid from users;

./create-timeline.sh case-p2
./print-timeline.sh case-p2 > timeline.txt

sudo ./get-histories.sh /media/part0 case-p2
tee mysql_bash_histories.txt
select * from histories order by recno;

sudo ./get-logfiles.sh /media/part0 case-p2
tee logfiles.txt
select logentry from logs where logfilename like '%sys%' order by recno;

sudo ./get-logins.sh /media/part0 case-p2
tee logins.txt
select * from logins order by start;
Tee login_failed.txt
select * from login_fails order by start;

Malware Analysis

objdump --disassemble -M intel /media/part0/usr/sbin/updatephp | tee objdump_updatephp.txt
strings updatephp
strings lampiao.sh
strings updatephp | grep "7777"