Hands On Lab: Unit 10

MICS-252, Fall 2024

Incident Response, Linux

Prepared by: Karl-Johan Westhoff

email: kjwesthoff@berkeley.edu

UC Berkeley School of Information

MICS Course 252 Fall 2024 (Kristy Westphal)

1 Introduction

We are given a snapshot of some files from a Linux host which has been compromised. Linux version 7.1 is mentioned, assuming this means the Red Hat Distribution 7.1, code named "Seawolf", released in 2001 [1]

2 Look at the files

It looks like we were issued the contents of Linux system 'var' folder (or parts thereof). In Linux the /var folder contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files [2]. Folder content is shown in Figure 1

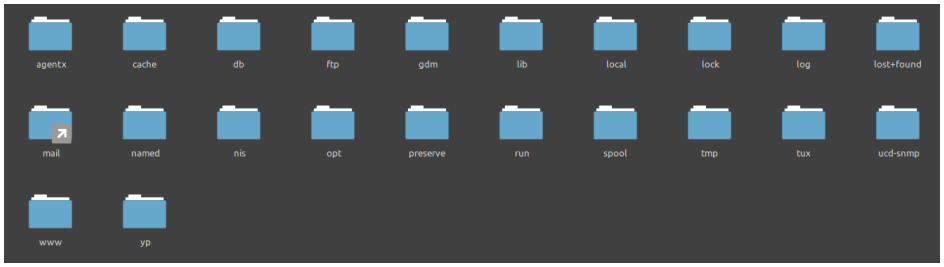

Figure 1: *We are given the 'var' folder from a compromised Linux host*

2.1 /tmp folder

An interesting folders is the /tmp folder holding temporary files. The /tmp folder could contain artifacts from someone trying privilege escalation or achieve persistence. Sometimes un-careful design of by user privileged sudo executable files may contain relative links to files which are then executed with sudo privilege, and a /tmp folder is a good place to execute these. But the /tmp folder is empty.

2.2 /www folder

The /www folder indicates a web server was installed on the compromised host, this could be an attack vector. Sure enough poking around the html files installed by various modules in the web app, the following was enumerated:

- Apache version 1.3b5 is used (html/manual/mod/mod_perl.html)
- mod_ssl version 2.8 is used to manage the server via SSL v2/v3 and TLS v1 implemented using openSSL (html/manual/mod/mod_ssl/index.html)
- uses s php-nuke Version from 2003 to handle database transactions to a mySQL database

The modules and versions of the installed web server are outdated and expose multiple vulnerabilities: XSS, remote code execution, SQL injection [3].

It is not stated if the compromised Linux host functioned as a webserver, or even if the web server was operating at the time of breach. Will look for evidence in the further analyses.

2.3 /log folder

The log folder has the following sub folders:

- httpd: Apache webserver log files, contains time stamped access and error log entries
- news: no content
- sa: may hold "system activity" logs, has a limited number of binary files and some HH:MM:SS time stamped entries
- samba: Indicates that a SMB file service (Interface to Windows files sharing protocol) has been running. The logs contain some time stamped "authentication failed" entries and connection reset entries. These could be correlated with other events based on timestamp
- squid: logs for a caching proxy used with the web server (indicating that the compromised host was deployed as a web server - you would not need caching for a test/small number of user application) Holds time stamped logs showing failed http requests etc.
- vbox: empty

The log folder holds a number of species of log files, split into multiple files, some of which are empty:

- boot.log.x (1-25): Logging startup, could show artifacts of malicious persistence on the host
- cron.x (1-25): Logs scheduled jobs, definitely a place to look for persistence attempts

- dmesg: Kernel stuff, may capture things during boot before other services come online
- lastlog: Binary
- maillog (1-25): Logs for mail-server (apparently the thing also served mail..)
- messages (1-25): Messages from various services (syslog, ftp etc)
- mysqld.log(1-25): Not much here, most of the files are empty
- pacct (1-6): Binary files, containing information on user's command execution
- secure (1-25): Authentication events (logins, ssh etc.)
- spooler (1-25): Empty
- tmplog: Looks like a mashup of different files
- wtmp (1): Binary
- xferlog (1-24): Empty

3 The log files

Just looking at the files, the "secure" logs, containing login attempts and the "cron" logs containing persistently scheduled jobs look the most interesting. The logs show some authentication attempts that failed.
I decided to explore all the files, looking for keywords indicating failed login attempts, tool of choice: bash and grep (see Appendix B) for some notes on the process of choosing tools..

```
grep -Ei 'fail*|illegal|unauth*|refused' {messages*,secure*,boot*,dmesg*,cron*}
```

Gives a long list of failed login attempts, further reducing this to get a list of unique users and ip addresses repeatedly trying to login, piping (`|`) the previous result to a regex searching for IP addresses:

3.1 Suspicious IP's

```
grep -E -o '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)'
```

And sorting by unique occurrences and counting the attempts:

```
 | sort | uniq -c | sort -nr
```

Gives a list of 'candidates' with a high number of failed attempts to log in Figure 2 (full list shown in Appendix A):

```
kj@kj-ThinkPad-T460p:~/Documents/Courses/MICS/252 SecOps/Assingments/HandsOnLab10/Compromi
zedHostFiles/var/log$ grep -Ei 'fail*|illegal|unauth*|refused' {messages*,secure*,boot*,dm
esg*,cron*} | grep -E -o '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01
]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-
9]?)' | sort | uniq -c | sort -nr
    368 172.184.233.242
    133 80.213.62.170
     58 63.251.144.88
     38 220.117.203.9
     35 131.234.157.10
     23 194.47.249.18
     22 65.39.171.249
     20 82.67.216.46
```

Figure 2: *IP's sorted by failed login attempts, complete list in Appendix A*

"172.184.233.242" has 368 failed attempts and seems to be associated with the domain "acb8e9f2.ipt.aol.com", grepping for that gives a long list of failed FTP logons. Trying to see if it was successful at some point

A bunch of ftp sessions in the "secure" logs show as started, the logs show File:Date Time Host (ns1) service(xinetd)[pid,704] protocol(ftp), xinetd pid and source IP

```
secure.17:Jun 16 23:21:58 ns1 xinetd[704]: START: ftp pid=16450 from=172.184.233.242
secure.17:Jun 16 23:21:59 ns1 xinetd[704]: START: ftp pid=16451 from=172.184.233.242
secure.17:Jun 16 23:21:59 ns1 xinetd[704]: START: ftp pid=16452 from=172.184.233.242
secure.17:Jun 16 23:21:59 ns1 xinetd[704]: START: ftp pid=16453 from=172.184.233.242
secure.17:Jun 16 23:21:59 ns1 xinetd[704]: START: ftp pid=16454 from=172.184.233.242
```

Figure 3: *FTP processes attempted starts*

They all seem to be short duration, 45-65s (matching START and EXIT for each pid) seconds indicating they were refused by the server, so nothing conclusive, other that a lot of FTP logon attempts from the source.

"80.213.62.170" has 133 failed logins trying to access an SSH service, but is unsuccessful, looks like a failed credential spraying attempt.

"Remaining candidates" The rest of the ip's with failed logon attempts in Figure 2 are related to failed ssh logons.

3.2 grep'ing for ssh

grep and filtering out all the rejected logon attempts:

```
grep -Ei 'sshd' {messages*,secure*,boot*,dmesg*,cron*}
| grep -E -v -i 'fail*|clos*|disconn*|reverse*|did*|illegal'
```

Shows that some IP's are scanning the ssh server using "SSH-1.0-SSH_Version_Mapper . Don't panic." again this could indicate someone poking around. Checking for successful logons next:

```
grep -Ei 'sshd' {messages*,secure*,boot*,dmesg*,cron*} | grep -E -i accepted
```

Shows that only the user 'test' has been able to log in, and making a unique list of the IP's, see Figure 4 narrows it down to a list that can be compared to the sources with many failed logons:

```
kj@kj-ThinkPad-T460p:~/Documents/Courses/MICS/252 SecOps/Assingments/HandsOnLab10/CompromizedHostFiles/var/log$ grep -Ei 'sshd' {messages*,secure*,boot*,dme
sg*,cron*} | grep -E -i accepted | grep -E -o '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)' | sort | uniq -c | sort -nr
     20 140.115.238.250
     10 61.34.6.105
     10 218.236.84.50
     10 218.106.100.101
     10 217.64.96.195
     10 211.169.202.21
     10 210.92.30.90
     10 210.58.80.133
      8 195.114.90.12
      3 211.75.221.105
      2 80.17.37.11
      2 68.121.176.36
      2 203.193.14.26
      1 212.155.111.21
```

Figure 4: *List of IP's that have successfully logged in*

The successful logins, which also have failed attempts are:

```
218.106.100.101 (10 failed attepmts)
212.155.111.21 (4 failed attempts, looks like a forgotten password)
```

218.106.x... has 10 failed logon attempts, will look into what else that host has been up to. The IP has 10 failed attempts on September 27, but manages to log in on October 7. All attempts are in quick succession spanning a few seconds, looks automated see Figure 5

```
kj@kj-ThinkPad-T460p:~/Documents/Courses/MICS/252 SecOps/Assingments/HandsOnLab10/CompromizedHostFiles/var/log$ grep -Ei '218.106.100.101' {messages*,secure
*,boot*,dmesg*,cron*}
secure.1:Oct  7 15:42:16 ns1 sshd[10965]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:16 ns1 sshd[10965]: Accepted password for test from 218.106.100.101 port 41101 ssh2
secure.1:Oct  7 15:42:16 ns1 sshd[10966]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:16 ns1 sshd[10966]: Accepted password for test from 218.106.100.101 port 41120 ssh2
secure.1:Oct  7 15:42:17 ns1 sshd[10967]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:17 ns1 sshd[10967]: Accepted password for test from 218.106.100.101 port 41158 ssh2
secure.1:Oct  7 15:42:17 ns1 sshd[10968]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:17 ns1 sshd[10968]: Accepted password for test from 218.106.100.101 port 41163 ssh2
secure.1:Oct  7 15:42:18 ns1 sshd[10969]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:18 ns1 sshd[10969]: Accepted password for test from 218.106.100.101 port 41169 ssh2
secure.1:Oct  7 15:42:18 ns1 sshd[10970]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:18 ns1 sshd[10970]: Accepted password for test from 218.106.100.101 port 41174 ssh2
secure.1:Oct  7 15:42:18 ns1 sshd[10971]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:18 ns1 sshd[10971]: Accepted password for test from 218.106.100.101 port 41191 ssh2
secure.1:Oct  7 15:42:19 ns1 sshd[10973]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:19 ns1 sshd[10973]: Accepted password for test from 218.106.100.101 port 41208 ssh2
secure.1:Oct  7 15:42:19 ns1 sshd[10972]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:19 ns1 sshd[10972]: Accepted password for test from 218.106.100.101 port 41206 ssh2
secure.1:Oct  7 15:42:19 ns1 sshd[10974]: Could not reverse map address 218.106.100.101.
secure.1:Oct  7 15:42:19 ns1 sshd[10974]: Accepted password for test from 218.106.100.101 port 41212 ssh2
secure.2:Sep 27 01:23:18 ns1 sshd[5436]: Could not reverse map address 218.106.100.101.
secure.2:Sep 27 01:23:18 ns1 sshd[5436]: Failed password for illegal user test from 218.106.100.101 port 36746 ssh2
secure.2:Sep 27 01:23:18 ns1 sshd[5436]: Connection closed by 218.106.100.101
secure.2:Sep 27 01:23:18 ns1 sshd[5437]: Could not reverse map address 218.106.100.101.
```

Figure 5: *218.106.100.101 successful logons, multiple ssh processes in quick succession*

4 Conclusion

The ssh logons from 218.106.100.101 look suspicious, the cron logs however, did not show signs of persistence. Next step would be to look at the pacct logs which are in binary (will require a tool for reading). For now I think 218.106.100.101, posing a user 'test' is the culprit. Determining if the Apache web server was the initial attack vector can be determined by looking for reverse shell commands being executed, this may be found in the pacct logs, i was unable to find logs for the Apache server and grepping for reverse shell execution (bash -i >) did not reveal anything. So for now 218.106.100.101 should be investigated further..

References

[1] Wikipedia contributors. *Red Hat Linux — Wikipedia, The Free Encyclopedia*. `https://en.wikipedia.org/w/index.php?title=Red_Hat_Linux&oldid=1240829700`. [Online; accessed 27-October-2024]. 2024.

[2] The Linux foundation. *Chapter 5. The /var Hierarchy Linux foundation*. `https://refspecs.linuxfoundation.org/FHS_3.0/fhs/ch05.html`. [Online; accessed 27-October-2024].

[3] Apache Project. *Apache HTTP Server 1.3 vulnerabilities*. `https://httpd.apache.org/security/vulnerabilities_13.htmlnote`. [Online; accessed 27-October-2024].

Appendices

A IP's with failed logins

```
grep -Ei 'fail*|illegal|unauth*|refused' {messages*,secure*,boot*,dmesg*,cron*}
| grep -E -o '(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)
\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)'
| sort | uniq -c | sort -nr
```

```
    368 172.184.233.242
    133 80.213.62.170
     58 63.251.144.88
     38 220.117.203.9
     35 131.234.157.10
     23 194.47.249.18
     22 65.39.171.249
     20 82.67.216.46
     20 69.19.154.44
     20 64.246.58.96
     20 61.211.237.34
     20 219.94.51.51
     20 218.104.55.15
     20 212.40.162.195
     20 211.180.157.251
     20 211.174.186.145
     19 65.194.200.129
     19 210.0.186.83
     17 217.172.188.217
     11 61.41.235.53
     11 211.114.173.193
     10 80.237.77.94
```

```
10 70.240.3.138
10 67.18.172.10
10 64.140.42.69
10 61.166.155.162
10 219.113.213.21
10 218.30.21.236
10 218.158.126.247
10 218.106.100.101
10 213.232.127.249
10 210.114.220.147
10 196.40.45.116
10 196.35.68.95
10 163.180.21.201
10 153.104.6.221
 9 61.166.6.60
 9 224.0.1.1
 9 220.92.31.135
 9 203.72.251.14
 9 195.78.43.183
 9 193.96.238.131
 6 69.0.134.72
 6 62.179.22.250
 6 203.71.62.9
 4 212.155.111.21
 4 211.240.65.3
 2 61.221.77.82
 2 220.168.17.55
 2 216.234.56.11
 1 65.104.169.18
 1 62.204.197.193
 1 62.166.92.73
 1 212.118.103.2
 1 211.117.191.70
```

```
1 210.83.203.34
1 210.112.147.22
1 142.207.115.6
```

B A bit on my process working on this

My initial idea was to use this assignment to explore a new log analysis tool, so I installed "The Sleuth Kit" and Autopsy. That took quite some time and the results were inconclusive...

Then I thought 'pandas' in python using 'glob' would be a good fit, easy graphics to see what is going on etc.. But pandas is slow going and not ideal for working with many different kinds of log files.. it was taking too long I was running out of time.

So I decided to poke around the data using grep and awk. It proved to be, bay far the quickest and also the most intuitive