

Methodology

Common methods of analysis:

- 1.) What port is the malware listening on?
- 2.) What is the Process ID (PID) of the malware? This answer cannot be verified, but it's required for the next question(s).
- 3.) What is the name of the binary or executable associated with the network listener?
- 4.) What is the full path of the previously identified binary?
- 5.) Run the ps command against the previously identified malicious Process ID (PID). What is the full command line invocation of this binary?
- 6.) List the full process tree for the malicious process. What is the name of the parent process that spawned the malware?
- 7.) What is the SHA-256 hash value of the malware? Note: This is not asking about the challenge.elf
- 8.) Extract all printable **string** characters from the binary. Review the results for any indicators such as risky objects, descriptions, comments, banners, or usage information. Based on these findings, what utility or program can you identify?

9.) Examine the system-wide cron jobs configured on the local system. Identify any entries that may be malicious, indicating persistence mechanisms or continued compromise by the attacker. What is the full file path of the script that is executed?

10.) Use a service like crontab.guru to interpret the cron schedule. What is the specific day when this cron entry runs?

11.) Now, list your own user account's crontab entries. What IP address does the attacker's beacon reach out to?

My methodology :

1.) First network connection check, like port

2.) Next, find the network listen file name and extract malware file hashes and find the malware execution file location (use the proc directory)

3.) Analyze main malware extract **strings using floss and strings tool** and extract and check file reputation finally execute sandbox or internet sandbox

4.) Next analysis that extracts strings and takes a verdict on what to do that malware like nc, cmd use or crontab, and comes output using AI analysis, and confirms what to do with the malware

5.) Next, check all files for malware execution places because that malware-related file chance EX: crontab log file, dropper malware files

6.) Process analysis finds process IDs (parent and child) and analyzes the proc directory for specific malware processes

7.) Sometimes malware creates a dropper payload, then that dropper executes after deleting the dropper payloads, but runs in still background that case, analysis (proc directory, this virtual file system is created by the kernel), that specific malware process

8.) Crontab analysis case analysis,s, wide level check, user level,l, and check all crontab directories like shell code, and finally check for malicious files there

9.) If the crontab finds the case analysis date and extracts that date, use crontab.guru

10.) Always check the logs manual and automate used tools, grep (manual) , logwatch (automate)

Important Commands for Linux analysis:

Network analysis:

```
$ sudo netstat -tnp ( or ) netstat -atnp
```

```
$ sudo netstat -tulnp
```

```
$ sudo ss -tnp
```

```
$ sudo lsof -i -P
```

```
$ sudo lsof -p
```

Process analysis :

What process runs the system:

```
$ sudo ps
```

```
$ sudo ps -u <username>
```

```
$ sudo ps -AFH | less
```

```
$ sudo ps -p < parent or child process_id > -F
```

```
$ sudo ps --ppid <parent id 3764 > -F
```

Find the parent-child process using pstree:

```
$pstree ⇒ show hierarchical view
```

```
$pstree -p -s <process_id> ⇒ more suitable specific process
```

The dynamic update running process uses top :

\$sudo top ⇒ show all processes and dynamically update all new processes

\$sudo top -u ms17 ⇒ extract specific uses

\$sudo top -u ms17 -c ⇒ show command of execute

\$sudo top -u ms17 -c -o -TIME+ ⇒ Capture recently process

proc directory find all processes:

\$cd /proc/3894 ⇒ show all processes and dynamically update all new processes

some import file to proc reverse shell directory:

\$cmd ⇒ cmd execute file name

\$cwd ⇒ malware execution location

\$cat environ | tr '\0' '\n' ⇒ It contains the environment variables of the running process (in string format).

Example: PATH, USER, HOME, LANG, SHELL, SECRET_KEY, etc.

but our case important username=tcm , and home=/home/tcm

\$exe ⇒ ls -al exe (show what file and malware execute location)

Delete the file but run background in proc dir analysis:

Note that even malware delete case process runs background because that runs a virtual file (proc directory), which means it runs in memory.

```
$ps -AHF | grep "notmalware"
```

⇒ Create a backup malware file

⇒ and DELETE the malware file

```
$ps -AHF | grep "notmalware" | grep -v grep
```

\$ lsof -p <malware process id> = (in that case, malware runs background, even malware file is deleted)

\$ lsof +L1 ⇒ This command views all deleted files, even those running in the background

If you check go proc directory and see that still available

Crontab analysis:

\$ cat /etc/crontab ⇒ we can malicious cmds or files

\$ \$ ls -al /var/spool/cron/crontabs/ ⇒ which users use crontab

\$ sudo cat /var/spool/cron/crontabs/tcm ⇒ Read user file

Instead, another analysis cmds:

```
$ sudo crontab -u tcm -l
```

Analysis crontab main directory

```
$ ls -al /etc | grep cron
```

⇒ Check all directories like

cron.d

cron.daily

cron.hourly

cron.yearly



cron.weekly



cron.monthly

Log analysis:

Manual Logs analysis:

Key Linux Log Files to Analyze (Security-Relevant)

 Log File	 Purpose
<code>/var/log/auth.log</code> (Debian)	Authentication events (login, sudo, SSH, failures, etc.)
<code>/var/log/secure</code> (RHEL/CentOS)	Same as <code>auth.log</code> (for RedHat-based systems)

 Log File	 Purpose
<code>/var/log/syslog</code>	System-wide logs, including services
<code>/var/log/messages</code>	Kernel + system logs (RHEL-based)
<code>/var/log/cron</code>	Cron job execution logs
<code>/var/log/wtmp</code>	Binary log of user logins/logouts – use <code>last</code>
<code>/var/log/btmp</code>	Failed login attempts – use <code>lastb</code>
<code>/var/log/dmesg</code>	Kernel ring buffer (boot, hardware, drivers)
<code>/var/log/bash_history</code>	Shell command history (if not cleared by attacker)
<code>/var/log/audit/audit.log</code>	SELinux/auditd logs (if auditd is enabled)
<code>/var/log/httpd/access_log</code>	Web server logs (Apache/Nginx) – good for web attack tracing

Manual Log Analysis Techniques:

View SSH login attempts

```
$ grep 'sshd' /var/log/auth.log
```

View failed login attempts

```
$ grep 'Failed password' /var/log/auth.log
```

View successful logins

```
$ grep 'Accepted password' /var/log/auth.log
```

Check for new user creation


```
$ grep 'useradd' /var/log/auth.log
```

```
# Review Command Execution (if bash history available)
```

```
$ cat ~/.bash_history
```

Auto Logs analysis:

```
$ sudo logwatch --detail High --service All --range today --format text
```

```
$ sudo logwatch --detail High --range today > /tmp/log_report.txt
```

File analysis:

```
$ floss test.elf
```

or

```
$ strings test.elf
```