PENETRATION TESTING (VULNER)

Yap Ching Siong
CFC2407

Contents

Overview of the script

This script is created to automate discovery of live hosts on LAN, scan for open ports and detect possible vulnerabilities on the found live hosts.

User can exclude IP/device from the scan by inputting the IP address when asked.

User need to provide user name list, provide or generate a password list for the next action to check for weak password through brute force using hydra.

After files are provided, the script will perform brute force on the first log in service found for each device, currently it is limited to ftp, ssh and telnet. The brute force result will be printed on terminal and saved to reports.

Throughout the execution of the script, results will be saved to reports.

All reports are saved in /Vulner folder.

A general statistic will be printed on terminal after the brute force attempt.

At the end of script execution, directory of the reports is presented to user. User is also given a choice to view full report or report of individual IP/device.

The installTools() function will install Nmap, hydra and cupp.

The createDir() function will create directories results if they are not already created.

The networkScan() function will find out LAN range, IP address of default gateway and local machine. Then perform a scan to find out all live hosts on LAN. This is also where user can decide which IP address to exclude from enumeration, vulnerabilities scan and brute force.

The enumDevice() function will execute port and vulnerabilities scan using Nmap.

The defUserList() function will request user name list from user and check whether it exists.

The defPassList() function will request password list from user and check whether it exists, or generate password list.

The passwdCheck() function will perform brute force using hydra on the first log in service on each enumerated IP/device.

The statistics() function will present information such as Date/Time, Duration, number of live hosts found, number of devices included and excluded from enumeration.

The reportSum() function will compile Report Summary file for user.

The viewReport() function will display the directory where the reports are located and give user option to view either report summary or a specific IP/device's report.

Detail code explanation will be in the following sections.

installTools function

The inst function will execute the following actions:

1. apt-get update
2. nmap tool installation
3. hydra tool installation
4. cupp tool installation

```
function installTools()
{
echo "************************************"
echo "Starting Tools Installation & Updates"
echo "************************************"
sleep 2
sudo apt-get update -y
sudo apt-get install nmap -y
sudo apt-get install hydra -y
sudo apt-get install cupp -y
echo "************************************"
echo "Tools Installation & Update Completed"
echo "************************************"
echo -e "\n\n\n"
sleep 5
}
```

Declare function name installTools

echo to print information on terminal to notify user tools installation will be starting and user
should provide inputs when required.

echo "****" --prints *** and is for cosmetic purpose to highlight the message within.

echo prints "Starting Tools Installation & Updates" to notify user of current status.

sleep 2 --Pause for 2 seconds before continuing the script

function installTools()

{

*echo "************************************"*

echo "Starting Tools Installation & Updates"

*echo "************************************"*

sleep 2

```
┌──(kali㉿kali)-[~]
└─$ bash Vulner.sh
************************************
Starting Tools Installation & Updates
************************************
```

sudo = superuser doer, used in front of a command to execute normal user command with root.
privileges without changing to root user. sudo is used because some commands require elevated
permission.

apt-get update, downloads versions of updated package information or their dependencies to
local machine.

-y flag tells apt-get to assume all answers to prompt is yes.

sudo apt-get update -y

```
[sudo] password for kali:
Get:1 http://mirror.aktkn.sg/kali kali-rolling InRelease [30.6 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 Packages [19.2 MB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 Contents (deb) [44.2 MB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/contrib amd64 Packages [114 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/contrib amd64 Contents (deb) [173 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/non-free amd64 Packages [239 kB]
Get:7 http://mirror.aktkn.sg/kali kali-rolling/non-free amd64 Contents (deb) [906 kB]
Fetched 64.8 MB in 11s (6,024 kB/s)
Reading package lists ... Done
```

Install nmap tool.

sudo apt-get install nmap -y

```
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libc-bin libc-dev-bin libc-devtools libc-l10n libc6 libc6-dev libc6-i386 locales nmap-common
Suggested packages:
  glibc-doc libnss-nis libnss-nisplus ncat ndiff zenmap
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-devtools libc-l10n libc6 libc6-dev libc6-i386 locales nmap nmap-common
10 upgraded, 0 newly installed, 0 to remove and 1699 not upgraded.
Need to get 18.5 MB of archives.
After this operation, 4,191 kB disk space will be freed.
Get:1 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-l10n all 2.36-6 [672 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-devtools amd64 2.36-6 [50.7 kB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-dev-bin amd64 2.36-6 [43.1 kB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6-dev amd64 2.36-6 [1,897 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6-i386 amd64 2.36-6 [2,453 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/main amd64 locales all 2.36-6 [3,901 kB]
Get:7 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6 amd64 2.36-6 [2,745 kB]
Get:8 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-bin amd64 2.36-6 [604 kB]
Get:9 http://http.kali.org/kali kali-rolling/non-free amd64 nmap amd64 7.93+dfsg1-0kali2 [2,009 kB]
Get:10 http://http.kali.org/kali kali-rolling/non-free amd64 nmap-common all 7.93+dfsg1-0kali2 [4,155 kB]
Fetched 18.5 MB in 2s (7,758 kB/s)
Preconfiguring packages ...
(Reading database ... 338510 files and directories currently installed.)
Preparing to unpack .../0-libc-l10n_2.36-6_all.deb ...
Unpacking libc-l10n (2.36-6) over (2.33-8) ...
Preparing to unpack .../1-libc-devtools_2.36-6_amd64.deb ...
Unpacking libc-devtools (2.36-6) over (2.33-8) ...
Preparing to unpack .../2-libc-dev-bin_2.36-6_amd64.deb ...
Unpacking libc-dev-bin (2.36-6) over (2.33-8) ...
Preparing to unpack .../3-libc6-dev_2.36-6_amd64.deb ...
Unpacking libc6-dev:amd64 (2.36-6) over (2.33-8) ...
Preparing to unpack .../4-libc6-i386_2.36-6_amd64.deb ...
Unpacking libc6-i386 (2.36-6) over (2.33-8) ...
Preparing to unpack .../5-locales_2.36-6_all.deb ...
Unpacking locales (2.36-6) over (2.33-8) ...
Preparing to unpack .../6-libc6_2.36-6_amd64.deb ...
Checking for services that may need to be restarted ...
Checking init scripts ...
```

```
Restarting services possibly affected by the upgrade:
  ssh: restarting ... done.
  cron: restarting ... done.

Services restarted successfully.
(Reading database ... 338503 files and directories currently installed.)
Preparing to unpack .../libc-bin_2.36-6_amd64.deb ...
Unpacking libc-bin (2.36-6) over (2.33-8) ...
Setting up libc-bin (2.36-6) ...
(Reading database ... 338503 files and directories currently installed.)
Preparing to unpack .../nmap_7.93+dfsg1-0kali2_amd64.deb ...
Unpacking nmap (7.93+dfsg1-0kali2) over (7.92+dfsg2-1kali1) ...
Preparing to unpack .../nmap-common_7.93+dfsg1-0kali2_all.deb ...
Unpacking nmap-common (7.93+dfsg1-0kali2) over (7.92+dfsg2-1kali1) ...
Setting up libc-l10n (2.36-6) ...
Setting up locales (2.36-6) ...
Installing new version of config file /etc/locale.alias ...
Generating locales (this might take a while)...
  en_US.UTF-8... done
Generation complete.
Setting up nmap-common (7.93+dfsg1-0kali2) ...
Setting up libc6-i386 (2.36-6) ...
Setting up libc-dev-bin (2.36-6) ...
Setting up libc-devtools (2.36-6) ...
Setting up nmap (7.93+dfsg1-0kali2) ...
Setting up libc6-dev:amd64 (2.36-6) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for kali-menu (2022.3.1) ...
Processing triggers for libc-bin (2.36-6) ...
```

Install hydra tool.

sudo apt-get install hydra -y

```
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  hydra-gtk icu-devtools libbson-1.0-0 libicu-dev libicu72 libmongoc-1.0-0
Suggested packages:
  icu-doc
The following NEW packages will be installed:
  libicu72
The following packages will be upgraded:
  hydra hydra-gtk icu-devtools libbson-1.0-0 libicu-dev libmongoc-1.0-0
6 upgraded, 1 newly installed, 0 to remove and 1693 not upgraded.
Need to get 20.6 MB of archives.
After this operation, 38.0 MB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 libbson-1.0-0 amd64 1.23.1-1+b1 [76.3 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libicu72 amd64 72.1-3 [9,376 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 libmongoc-1.0-0 amd64 1.23.1-1+b1 [304 kB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/main amd64 hydra amd64 9.4-1 [275 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/main amd64 hydra-gtk amd64 9.4-1 [42.6 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libicu-dev amd64 72.1-3 [10.3 MB]
Get:7 http://mirror.aktkn.sg/kali kali-rolling/main amd64 icu-devtools amd64 72.1-3 [206 kB]
Fetched 20.6 MB in 2s (8,502 kB/s)
(Reading database ... 338503 files and directories currently installed.)
```

```
Preparing to unpack .../0-libbson-1.0-0_1.23.1-1+b1_amd64.deb ...
Unpacking libbson-1.0-0 (1.23.1-1+b1) over (1.22.0-1) ...
Selecting previously unselected package libicu72:amd64.
Preparing to unpack .../1-libicu72_72.1-3_amd64.deb ...
Unpacking libicu72:amd64 (72.1-3) ...
Preparing to unpack .../2-libmongoc-1.0-0_1.23.1-1+b1_amd64.deb ...
Unpacking libmongoc-1.0-0 (1.23.1-1+b1) over (1.22.0-1) ...
Preparing to unpack .../3-hydra_9.4-1_amd64.deb ...
Unpacking hydra (9.4-1) over (9.3-3+b1) ...
Preparing to unpack .../4-hydra-gtk_9.4-1_amd64.deb ...
Unpacking hydra-gtk (9.4-1) over (9.3-3+b1) ...
dpkg: considering deconfiguration of libicu-dev:amd64, which would be broken by installation of icu-devtools ...
dpkg: yes, will deconfigure libicu-dev:amd64 (broken by icu-devtools)
Preparing to unpack .../5-icu-devtools_72.1-3_amd64.deb ...
De-configuring libicu-dev:amd64 (71.1-3), to allow installation of icu-devtools (72.1-3) ...
Unpacking icu-devtools (72.1-3) over (71.1-3) ...
Preparing to unpack .../6-libicu-dev_72.1-3_amd64.deb ...
Unpacking libicu-dev:amd64 (72.1-3) over (71.1-3) ...
Setting up libicu72:amd64 (72.1-3) ...
Setting up libbson-1.0-0 (1.23.1-1+b1) ...
Setting up icu-devtools (72.1-3) ...
Setting up libicu-dev:amd64 (72.1-3) ...
Setting up libmongoc-1.0-0 (1.23.1-1+b1) ...
Setting up hydra (9.4-1) ...
Setting up hydra-gtk (9.4-1) ...
Processing triggers for mailcap (3.70+nmu1) ...
Processing triggers for kali-menu (2022.3.1) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for libc-bin (2.36-6) ...
Processing triggers for man-db (2.10.2-1) ...
```

Install cupp tool.

sudo apt-get install cupp -y

```
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  cupp
0 upgraded, 1 newly installed, 0 to remove and 1693 not upgraded.
Need to get 13.3 kB of archives.
After this operation, 60.4 kB of additional disk space will be used.
Get:1 http://http.kali.org/kali kali-rolling/main amd64 cupp all 0.0+20190501.git986658-6 [13.3 kB]
Fetched 13.3 kB in 1s (20.6 kB/s)
Selecting previously unselected package cupp.
(Reading database ... 338521 files and directories currently installed.)
Preparing to unpack .../cupp_0.0+20190501.git986658-6_all.deb ...
Unpacking cupp (0.0+20190501.git986658-6) ...
Setting up cupp (0.0+20190501.git986658-6) ...
Processing triggers for kali-menu (2022.3.1) ...
Processing triggers for man-db (2.10.2-1) ...
```

Prints "Tools Installation & Update Completed" to notify user of current status.

*echo "************************************"*

echo "Tools Installation & Update Completed"

*echo "************************************"*

```
************************************
Tools Installation & Update Completed
************************************
```

-e flag to enable echo to interpret backslash, for this case echo interprets “\n” as a insert a newline
instead of printing “\n” on the terminal.

Inserts four new lines as spacing before the next function, for cosmetic purpose.

sleep 5 to wait for 5 seconds before proceed to next line of code follow by } to close the function.

echo -e "\n\n\n"

sleep 5

}

createDir function

The createDir function will create directory to store folders, temp files and reports created by this script.

In this case, Vulner folder will be created in present working directory.

```
function createDir()
{
wDir=$(pwd)
if [ ! -d "$wDir/Vulner" ]
then
        mkdir $wDir/Vulner
fi
}
```

Declare createDir as the function name.

Check the present working directory and store path string in wDir variable. This variable will be
used often throughout this script for accessing directories to save logs and results.

function createDir()

{

wDir=$(pwd)

-d flag to check for existence of folder.

! in if condition to check that the folder does not exist.

then mkdir to create Vulner folder in present working directory .

if [! -d "$wDir/Vulner"]

then

mkdir $wDir/Vulner

fi

}

```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  userpass.lst  Videos  Vulner  Vulner.sh
```

networkScan function

The networkScan function creates a folder when excuting the script to store temp files and results for each script execution, the folder is name as Result follow by date and time of script execution.

Then it retrieves the LAN network range, for nmap scan for all live hosts in the LAN.

It will also retrieve the IP address of the current machine IP, default gateway and add to list of IP that will not be further enumerated.

```
function networkScan()
{
cd Vulner
echo "****************************"
echo "Checking Local Network Range"
echo "****************************"
echo
sleep 2
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
start=$(date +%s)
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
mkdir $wDir/Vulner/Result$fileID
cd Result$fileID
networkRange=$(ip address | grep inet | grep eth0 | awk '{print $2}')
echo "Your Local Area Network Range is" $networkRange
echo
echo "*********************************"
echo "Local Network Range Scan Completed"
echo "*********************************"
echo -e "\n\n\n"
sleep 5
echo "*************************"
echo "Scanning LAN For Live Hosts"
echo "*************************"
echo
sleep 2
nmap -sn $networkRange | grep report | awk '{print $NF}' | sort -t . -k1,1n -k2,2n -k3,3n -k4,4n >> LiveHost.lst
echo "List of Live Hosts Found: "
cat LiveHost.lst
echo
echo "****************************"
echo "Live Host Scanning Completed"
echo "****************************"
echo -e "\n\n\n"
sleep 5
}
```

```
# Define network Scan as the function name.
# Change directory into Vulner folder.
# echo to print “Checking Local Network Range” to notify user current status.
# echo to insert spacing for cosmetic purpose.
# sleep 2 to wait for 2 seconds before executing next line of code.
function networkScan()
{
cd Vulner
echo "****************************"
echo "Checking Local Network Range"
echo "****************************"
echo
sleep 2
```

```
****************************
Checking Local Network Range
****************************
```

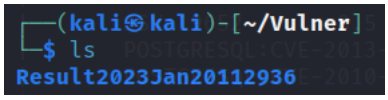
```
# date to print out current date and time.
# awk '{print $NF,$2,$3,$4}' is to print out specific information in a specific format; Year, Month,
#  Day, Time.
# Finally store this value in dateTime variable. This variable will be further processed for naming the
#  Result folder for every script execution for unique identification.
dateTime=$(date | awk '{print $NF,$2,$3,$4}')

# store unix time in seconds into start variable, will be used later for duration calculation.
start=$(date +%s)

# echo $dateTime, use the value in dateTime variable and perform the following action.
# tr -d [:space:] | tr -d [:punct:], remove space and punctuation from the $dateTime value
# Then store the new value in fileID variable.
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
```

Create folder using above fileID variable to store temp files and folders.

Change directory into the newly created folder.

mkdir $wDir/Vulner/Result$fileID

cd Result$fileID



ip address | -- show protocol of interfaces on current device and pass the result to next action.

grep inet | -- search for keyword ‘inet’ and pass the results to next action.

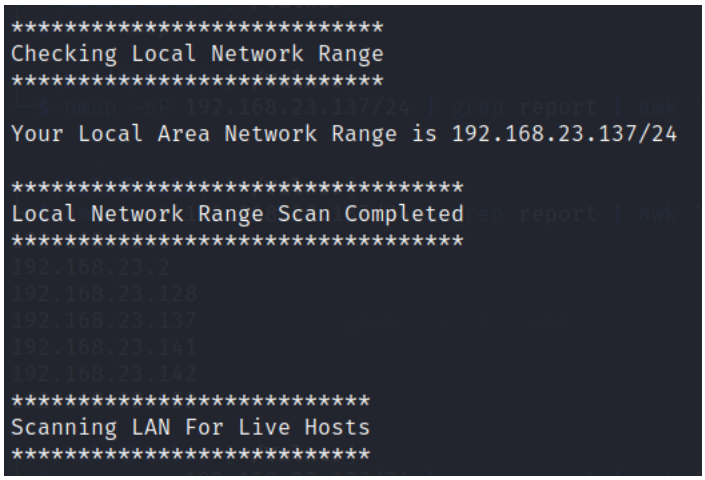
grep eth0 | -- search for keyword ‘eth0’ and pass the results to next action.

awk '{print $2}' -- print out the second word from the left.

Finally the value is stored in networkRange variable, this is the LAN network range information.

networkRange=$(ip address | grep inet | grep eth0 | awk '{print $2}')

```
# Print the LAN network range on Terminal.
# Print "Local Network Range Scan Completed" status on Terminal to notify user of current status.
# Print "Scanning LAN For Live Hosts" status on Terminal to notify user of current status.
# All echoes are for cosmetic purpose for printing on Terminal.
echo "Your Local Area Network Range is" $networkRange
echo
echo "**********************************"
echo "Local Network Range Scan Completed"
echo "**********************************"
echo -e "\n\n\n"
sleep 5
echo "****************************"
echo "Scanning LAN For Live Hosts"
echo "****************************"
echo
sleep 2
```



nmap -sP $networkRange | -- execute nmap to perform a ping sweep of LAN range to discover all
live hosts on the LAN, then pass the result to next action.

grep report | -- Search for keyword ‘report’ then pass the result to next action.

awk '{print $NF}' | -- print the last in the found statement from previous action, then pass the
result to the next action.

sort -- sort the found result from previous action.

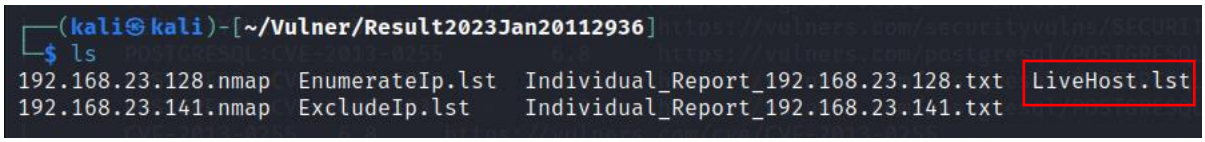
-t . -- using . as the separator for the sort command.

-k -- sort via a location (for our case it defines the column).

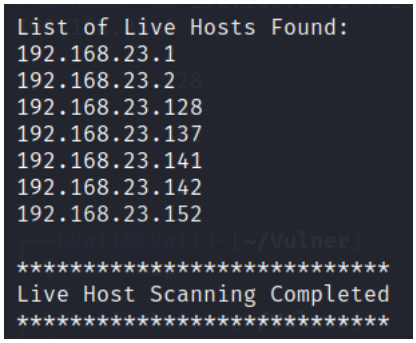
-n -- numeric sort.

sort -t . -k1,1n -k2,2n -k3,3n -k4,4n >> LiveHost.lst – sort the found ip addresses by numeric ascending order prioritise in the order, first octet, second octet, third octet, fourth octet and append the result in LiveHost.lst file.

nmap -sP $networkRange | grep report | awk '{print $NF}' | sort -t . -k1,1n -k2,2n -k3,3n -k4,4n >> LiveHost.lst



```
# Print on terminal text "List of Live Hosts Found: " following by the list of live hosts.
# Print "Live Host Scanning Completed" to notify user of current status.
echo "List of Live Hosts Found: "
cat LiveHost.lst
echo
echo "**************************"
echo "Live Host Scanning Completed"
echo "**************************"
echo -e "\n\n\n"
sleep 5
}
```

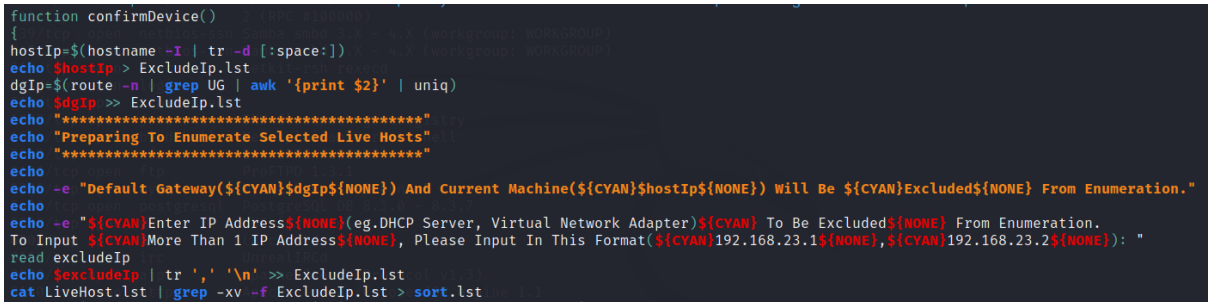


confirmDevice function

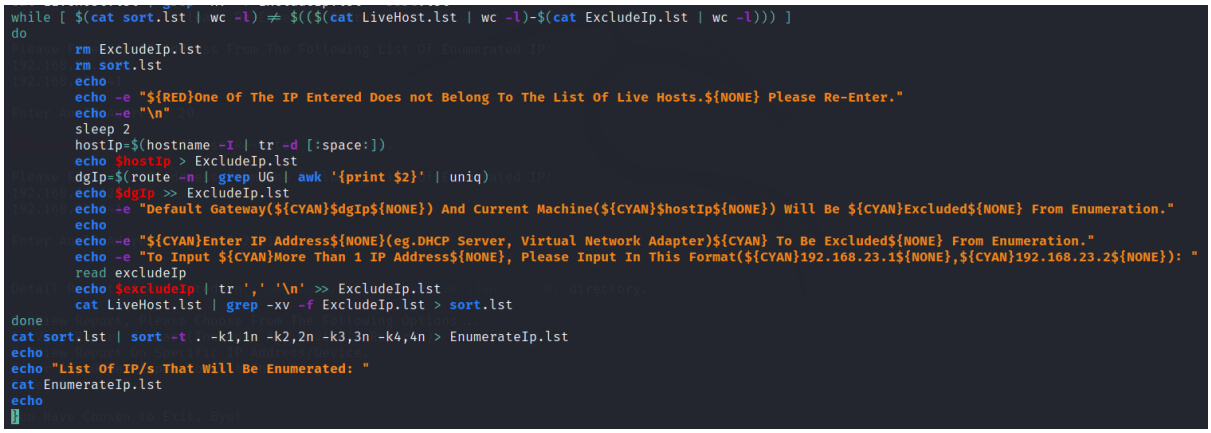
The confirmDevice function informs user the default gateway and current machine IP will be excluded from enumeration, and wants user to input which other IP should be excluded. It will then check the IP that user input to ensure it exists in LAN.

If IP does not exists in LAN, it will request user to input IP again.

If IP exists, it will list our the IP/devices that will be enumerated in the next function.



```
function confirmDevice()
{
hostIp=$(hostname -I | tr -d [:space:])
echo $hostIp > ExcludeIp.lst
dgIp=$(route -n | grep UG | awk '{print $2}' | uniq)
echo $dgIp >> ExcludeIp.lst
echo "****************************************"
echo "Preparing To Enumerate Selected Live Hosts"
echo "****************************************"
echo
echo -e "Default Gateway(${CYAN}$dgIp${NONE}) And Current Machine(${CYAN}$hostIp${NONE}) Will Be ${CYAN}Excluded${NONE} From Enumeration."
echo
echo -e "${CYAN}Enter IP Address${NONE}(eg.DHCP Server, Virtual Network Adapter)${CYAN} To Be Excluded${NONE} From Enumeration.
To Input ${CYAN}More Than 1 IP Address${NONE}, Please Input In This Format(${CYAN}192.168.23.1${NONE},${CYAN}192.168.23.2${NONE}): "
read excludeIp
echo $excludeIp | tr ',' '\n' >> ExcludeIp.lst
cat LiveHost.lst | grep -xv -f ExcludeIp.lst > sort.lst
```



```
while [ $(cat sort.lst | wc -l) -ne $(($(cat LiveHost.lst | wc -l)-$(cat ExcludeIp.lst | wc -l))) ]
do
        rm ExcludeIp.lst
        rm sort.lst
        echo
        echo -e "${RED}One Of The IP Entered Does not Belong To The List Of Live Hosts.${NONE} Please Re-Enter."
        echo -e "\n"
        sleep 2
        hostIp=$(hostname -I | tr -d [:space:])
        echo $hostIp > ExcludeIp.lst
        dgIp=$(route -n | grep UG | awk '{print $2}' | uniq)
        echo $dgIp >> ExcludeIp.lst
        echo -e "Default Gateway(${CYAN}$dgIp${NONE}) And Current Machine(${CYAN}$hostIp${NONE}) Will Be ${CYAN}Excluded${NONE} From Enumeration."
        echo
        echo -e "${CYAN}Enter IP Address${NONE}(eg.DHCP Server, Virtual Network Adapter)${CYAN} To Be Excluded${NONE} From Enumeration."
        echo -e "To Input ${CYAN}More Than 1 IP Address${NONE}, Please Input In This Format(${CYAN}192.168.23.1${NONE},${CYAN}192.168.23.2${NONE}): "
        read excludeIp
        echo $excludeIp | tr ',' '\n' >> ExcludeIp.lst
        cat LiveHost.lst | grep -xv -f ExcludeIp.lst > sort.lst
done
cat sort.lst | sort -t . -k1,1n -k2,2n -k3,3n -k4,4n > EnumerateIp.lst
echo
echo "List Of IP/s That Will Be Enumerated: "
cat EnumerateIp.lst
echo
}
```

Declare function name confirmDevice.

function confirmDevice()

{

hostname -I | -- show the ip address of current machine and pass the result to next action.

tr -d [:space:] -- delete spaces using translate command.

Finally store the value into hostIp variable, this is the ip address of current device.

hostIp=$(hostname -I | tr -d [:space:])

```
# Append the IP address of current device to ExcludeIp.lst file.
echo $hostIp > ExcludeIp.lst

# route -n | -- show routing table in numerical address and pass the result to next action.
# grep UG | -- search for keyword ‘UG’ and pass the results to next action.
# awk '{print $2}' -- print out the second word from the left.
# uniq -- omit repeated lines
# Finally store the value into dgIp variable, this is the default gateway of current device.
dgIp=$(route -n | grep UG | awk '{print $2}' | uniq)

# Append the default gateway of current device to ExcludeIp.lst file.
echo $dgIp >> ExcludeIp.lst

# Print "Preparing To Enumerate Selected Live Hosts" on Terminal to notify user of current status.
echo "****************************************"
echo "Preparing To Enumerate Selected Live Hosts"
echo "****************************************"
echo

# echo -e to interpret backslash, which is present in ${CYAN} and ${NONE} terminal variables.
# ${} is used for terminal variables so the code in the {} is executed in current environment.
# ${} variables are defined in the last section of the script.
# ${CYAN} and ${NONE} are used to colour the letters that are placed in between both variables.
# Prints information that the default gateway and current machine IP will be excluded from
# enumeration to the user.
echo -e "Default Gateway(${CYAN}$dgIp${NONE}) And Current Machine(${CYAN}$hostIp${NONE})
Will Be ${CYAN}Excluded${NONE} From Enumeration."
echo
```

Ask user to input IP address to be excluded from enumeration and prints information on the
format to input multiple IP.

Input by user is store in excludeIp variable.

echo -e "${CYAN}Enter IP Address${NONE}(eg.DHCP Server, Virtual Network Adapter)${CYAN} To Be Excluded${NONE} From Enumeration.

To Input ${CYAN}More Than 1 IP Address${NONE}, Please Input In This Format(${CYAN}192.168.23.1${NONE},${CYAN}192.168.23.2${NONE}): "

read excludeIp

```
Default Gateway(192.168.23.2) And Current Machine(192.168.23.137) Will Be Excluded From Enumeration.

Enter IP Address(eg.DHCP Server, Virtual Network Adapter) To Be Excluded From Enumeration.
To Input More Than 1 IP Address, Please Input In This Format(192.168.23.1,192.168.23.2):
192.168.23.150,192.168.23.1,192.168.23.142
```

Change the , to a newline in excludeIp variable and save to ExcludeIp.lst file.

echo $excludeIp | tr ',' '\n' >> ExcludeIp.lst

grep -xv to find lines that does not match.

-f ExcludeIp.lst to use ExcludeIp.lst file for the search.

Read LiveHost.lst file, find lines that does not match values in ExcludeIp.lst and save to sort.lst file.

cat LiveHost.lst | grep -xv -f ExcludeIp.lst >> sort.lst

Start of while loop to check whether user input IP exists in LAN.

$(cat sort.lst | wc -l) to compute the number of IP in sort.lst file. If IP does not exists in LAN, there
will be more IP than expected stored in sort.lst file.

$(($(cat LiveHost.lst | wc -l)-$(cat ExcludeIp.lst | wc -l))) to compute the number of IP difference by
subtracting the number of IP in ExcludeIp.lst from LiveHost.lst files.

!= --not equal condition, to perform while loop execution when the condition is met.

while [$(cat sort.lst | wc -l) != $(($(cat LiveHost.lst | wc -l)-$(cat ExcludeIp.lst | wc -l)))]

do

delete ExcludeIp.lst, for re-compilation.

rm ExcludeIp.lst

delete sort.lst, for re-compilation.

rm sort.lst

echo

Prints "One Of The IP Entered Does not Belong To The List Of Live Hosts.Please Re-Enter." to
inform user there is a mistake input by user.

echo -e "\n" --Two lines spacing.

sleep 2 --Pause for two seconds before proceed to next the line of code.

echo "One Of The IP Entered Does not Belong To The List Of Live Hosts.Please Re-Enter."

echo -e "\n"

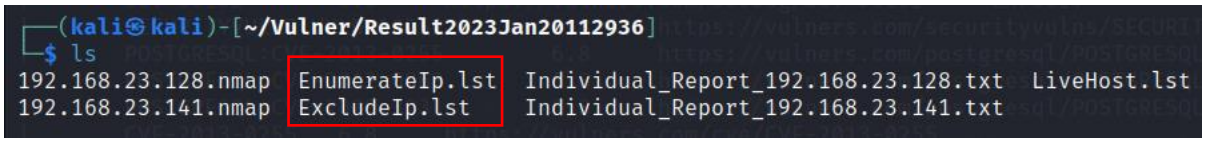
sleep 2

```
Default Gateway(192.168.23.2) And Current Machine(192.168.23.137) Will Be Excluded From Enumeration.

Enter IP Address(eg.DHCP Server, Virtual Network Adapter) To Be Excluded From Enumeration.
To Input More Than 1 IP Address, Please Input In This Format(192.168.23.1,192.168.23.2):
10.10.10.10

One Of The IP Entered Does not Belong To The List Of Live Hosts. Please Re-Enter.

Default Gateway(192.168.23.2) And Current Machine(192.168.23.137) Will Be Excluded From Enumeration.

Enter IP Address(eg.DHCP Server, Virtual Network Adapter) To Be Excluded From Enumeration.
To Input More Than 1 IP Address, Please Input In This Format(192.168.23.1,192.168.23.2):
```

```
# Repeats most of the code from the start of the confirmDevice function.

# Checks for Host IP and default gateway IP.

# Append hostIp, dgIP, and user input IP to ExcludeIp.lst file.

# Comparing ExcludeIp.lst and LiveHostIp.lst, then append the uncommon IP to sort.lst file for
#  enumeration in the next step.

        hostIp=$(hostname -I | tr -d [:space:])

        echo $hostIp > ExcludeIp.lst

        dgIp=$(route -n | grep UG | awk '{print $2}' | uniq)

        echo $dgIp >> ExcludeIp.lst

        echo -e "Default Gateway(${CYAN}$dgIp${NONE}) And Current
Machine(${CYAN}$hostIp${NONE}) Will Be ${CYAN}Excluded${NONE} From Enumeration."

        echo

        echo -e "${CYAN}Enter IP Address${NONE}(eg.DHCP Server, Virtual Network
Adapter)${CYAN} To Be Excluded${NONE} From Enumeration."

        echo -e "To Input ${CYAN}More Than 1 IP Address${NONE}, Please Input In This
Format(${CYAN}192.168.23.1${NONE},${CYAN}192.168.23.2${NONE}): "

        read excludeIp

        echo $excludeIp | tr ',' '\n' >> ExcludeIp.lst

        cat LiveHost.lst | grep -xv -f ExcludeIp.lst > sort.lst

done
```

Read sort.lst file and sort using . as separator in numeric order, starting from first octet to the
fourth octet, then save the results in EnumerateIp.lst file.

cat sort.lst | sort -t . -k1,1n -k2,2n -k3,3n -k4,4n > EnumerateIp.lst



Display to user values in EnumerateIp.lst the IP will be enumerated.

echo

echo "List Of IP/s That Will Be Enumerated: "

cat EnumerateIp.lst

echo

}

```
List Of IP/s That Will Be Enumerated:
192.168.23.128
192.168.23.141
```

enumDevice function

Nmap will use the list of IP to scan for open ports, service/version info, OS version and possible vulnerabilities via script scanning.

The scanned results will be saved into files for each IP/device.

```
function enumDevice ()
{
echo "*****************************************"
echo "Enumerating Live Hosts, Please Be Patient"
echo "*****************************************"
echo
```

```
for x in $(cat EnumerateIp.lst)
do
        sudo nmap $x -sV -O -oN $x.nmap
        echo
        echo " ___ _   _ ____ _____     _____ ____  _   _   _    _     " >> Individual_Report_$x.txt
        echo "|_ _| \ | |  _ \_ _\ \   / /_ _|  _ \| | | | / \  | |    " >> Individual_Report_$x.txt
        echo " | ||  \| | | | | | \ \ / / | || | | | | | |/ _ \ | |    " >> Individual_Report_$x.txt
        echo " | || |\  | |_| | |  \ V /  | || |_| | |_| / ___ \| |___ " >> Individual_Report_$x.txt
        echo "|___|_| \_|____/___|  \_/  |___|____/ \___/_/   \_\_____|" >> Individual_Report_$x.txt
        echo >> Individual_Report_$x.txt
        echo " ____  _____ ____   ___  ____ _____ " >> Individual_Report_$x.txt
        echo "|  _ \| ____|  _ \ / _ \|  _ \_   _|" >> Individual_Report_$x.txt
        echo "| |_) |  _| | |_) | | | | |_) || |  " >> Individual_Report_$x.txt
        echo "|  _ <| |___|  __/| |_| |  _ < | |  " >> Individual_Report_$x.txt
        echo "|_| \_\_____|_|    \___/|_| \_\|_|  " >> Individual_Report_$x.txt
        echo -e "\n" >> Individual_Report_$x.txt
        echo -e "${BOLD}Program Started On:${NONE} $dateTime" >> Individual_Report_$x.txt
        echo -e "${BOLD}Device IP:${NONE} $x" >> Individual_Report_$x.txt
        echo -ne "${BOLD}MAC Address:${NONE}" >> Individual_Report_$x.txt
        cat $x.nmap | grep -i "mac address" | awk -Fs: '{print $2}' >> Individual_Report_$x.txt
        echo -ne "${BOLD}OS Details:${NONE}" >> Individual_Report_$x.txt
        cat $x.nmap | grep -i "os details" | awk -Fs: '{print $2}' >> Individual_Report_$x.txt
        echo -e "\n${BOLD}Nmap Enumeration & Vulnerabilities Scan Results:${NONE}" >> Individual_Report_$x.txt
        cat $x.nmap >> Individual_Report_$x.txt
done
echo "********************"
echo "Enumeration Completed"
echo "********************"
echo -e "\n\n\n"
sleep 5
}
```

Declare function name enumDevice.

Print "Preparing To Enumerate Selected Live Hosts" on Terminal to notify user of current status.

function enumDevice()

{

Print “Enumerating Live Hosts, Please Be Patient”, update user the current status.

*echo "***"*

echo "Enumerating Live Hosts, Please Be Patient"

*echo "***"*

```
*****************************************
Enumerating Live Hosts, Please Be Patient
*****************************************
```

use for loop for to perform repeated action for each IP in EnumerateIp.lst file.

perform nmap scan for most common 1000 ports on the IP in EnumerateIp.lst.

use –script=vuln to scan for possible vulnerabilities on open ports.

-sV to scan for service and it's version.

-O to scan for OS version.

-oN $x.nmap to save results as normal output to file named ip address.

for x in $(cat EnumerateIp.lst)

do

sudo nmap $x --script=vuln -sV -O -oN $x.nmap

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-01-20 23:30 EST
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.128
Host is up (0.00022s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.1p1 Debian 1 (protocol 2.0)
MAC Address: 00:0C:29:AC:F7:78 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 37.11 seconds
```

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-01-20 23:31 EST
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.141
Host is up (0.00050s latency).
Not shown: 978 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2011-2523  BID:48539
|       vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|     Disclosure date: 2011-07-03
|     Exploit results:
|       Shell command: id
|       Results: uid=0(root) gid=0(root)
|     References:
|       https://www.securityfocus.com/bid/48539
|       https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
|       http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:4.7p1:
|       SECURITYVULNS:VULN:8166 7.5     https://vulners.com/securityvulns/SECURITYVULNS:VULN:8166
|       CVE-2010-4478   7.5     https://vulners.com/cve/CVE-2010-4478
|       CVE-2008-1657   6.5     https://vulners.com/cve/CVE-2008-1657
|       SSV:60656       5.0     https://vulners.com/seebug/SSV:60656    *EXPLOIT*
|       CVE-2010-5107   5.0     https://vulners.com/cve/CVE-2010-5107
|       CVE-2012-0814   3.5     https://vulners.com/cve/CVE-2012-0814
|       CVE-2011-5000   3.5     https://vulners.com/cve/CVE-2011-5000
|       CVE-2008-5161   2.6     https://vulners.com/cve/CVE-2008-5161
```

```
# Compile the Individual report for each IP/device enumerated.

# Print “INDIVIDUAL REPORT” header in the report.

        echo

        echo "___ _   _ ____ _____    _____ ____ _  _  _   _" >> Individual_Report_$x.txt

        echo "|_ _| \ | |  _ \_ _\ \  //_ _|  _ \| | | |/ \  | |" >> Individual_Report_$x.txt

        echo " | ||  \| | | | | | \ \/ / | || | | | | | |/ _ \ | |" >> Individual_Report_$x.txt

        echo " | || |\  | |_| | |  \ V /  | || |_| | |_| / ___ \| |" >> Individual_Report_$x.txt

        echo "|___|_| \_|____/___|  \_/  |___|____/ \___/_/   \_\_____|" >>
Individual_Report_$x.txt

        echo >> Individual_Report_$x.txt

        echo " ____  _____ ____   ___  ____ _____" >> Individual_Report_$x.txt

        echo "|  _ \| ____|  _ \ / _ \|  _ \_   _|" >> Individual_Report_$x.txt

        echo "| |_) |  _| | |_) | | | | |_) || |" >> Individual_Report_$x.txt

        echo "|  _ <| |___|  __/| |_| |  _ < | |" >> Individual_Report_$x.txt

        echo "|_| \_\_____|_|    \___/|_| \_\|_|" >> Individual_Report_$x.txt

        echo -e "\n" >> Individual_Report_$x.txt


# ${BOLD} and ${NONE} to bold the words that are placed in between the two variables then store in
#  the individual IP/device report.

        echo -e "${BOLD}Program Started On:${NONE} $dateTime" >> Individual_Report_$x.txt

        echo -e "${BOLD}Device IP:${NONE} $x" >> Individual_Report_$x.txt


#  echo -ne --do not output trailing new line when appending to individual IP/device report, so that
#  the following appended text will continue on the same line.

# grep -i – search and ignore case sensitivity on the term “mac address” on the store nmap result
# file then pass to next action.

# awk -Fs: '{print $2}' -- using s: as separator, print out the second word and append to individual
#  IP/device report.

        echo -ne "${BOLD}MAC Address:${NONE}" >> Individual_Report_$x.txt

        cat $x.nmap | grep -i "mac address" | awk -Fs: '{print $2}' >> Individual_Report_$x.txt
```

echo -ne --do not output trailing new line when appending to individual IP/device report, so that
the following appended text will continue on the same line.

grep -i – search and ignore case sensitivity on the term “os details” on the store nmap result file
then pass to next action.

awk -Fs: '{print $2}' -- using s: as separator, print out the second word and append to individual
IP/device report.

echo -ne "${BOLD}OS Details:${NONE}" >> Individual_Report_$x.txt

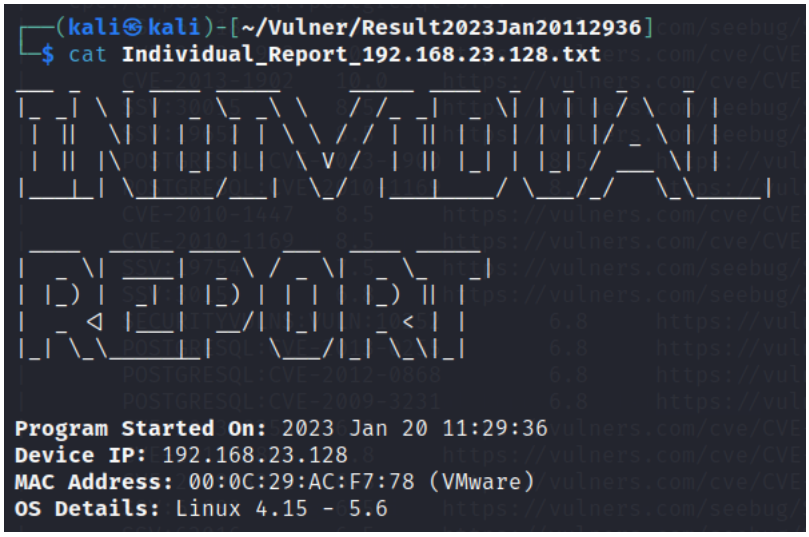
cat $x.nmap | grep -i "os details" | awk -Fs: '{print $2}' >> Individual_Report_$x.txt

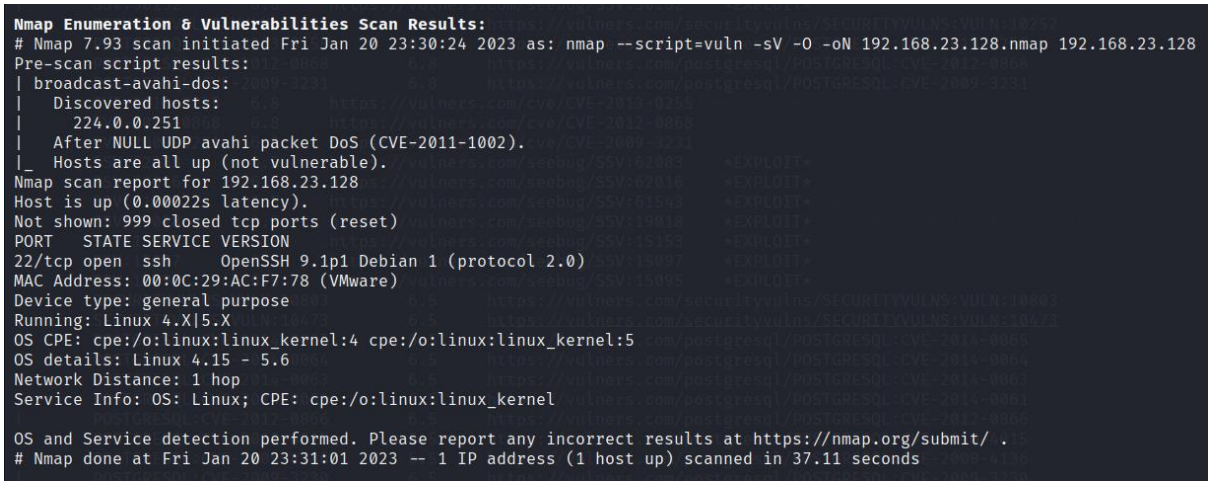
Print “Nmap Enumeration & Vulnerabilities Scan Results:” to individual IP/device report follow by
the nmap scan results of that IP/device.

echo -e "\n${BOLD}Nmap Enumeration & Vulnerabilities Scan Results:${NONE}" >> Individual_Report_$x.txt

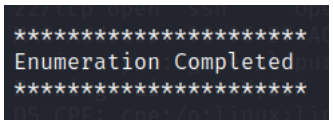
cat $x.nmap >> Individual_Report_$x.txt

done





```
#  Print out "Enumeration Completed" on Terminal to notify user of current status.
echo
echo "**********************"
echo "Enumeration Completed"
echo "**********************"
echo -e "\n\n\n"
sleep 5
}
```



defUserList function

The defUserlist function requires user to enter the file path of user name list to use for the weak password check action/brute force.

After user entered the file path of the username list, the script will check whether the file exists in system before proceeding to next step. If file does not exist, this function will be called again.

```
function defUserlist()
{
echo "************************************"
echo "Preparing To Check For Weak Passwords"
echo "************************************"
echo
read -p "Please Enter File Path Of The User List: " userlistPath
echo
if [ ! -f "$userlistPath" ];
then
        echo "File Does Not Exist. Please Re-Enter."
        echo
        defUserlist
fi
}
```

Define the function name defUserList.

Print "Preparing To Check For Weak Passwords" to notify user of current status.

Requests user the enter file path of the user list to use for brute force and store user's input into
userlistPath variable.

function defUserList()

{

*echo "************************************"*

echo "Preparing To Check For Weak Passwords"

*echo "************************************"*

echo

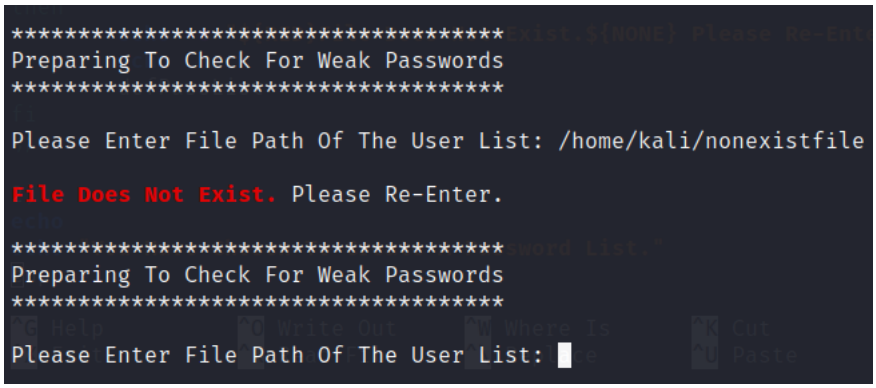
read -p "Please Enter File Path Of The User List: " userlistPath

echo

```
************************************
Preparing To Check For Weak Passwords
************************************

Please Enter File Path Of The User List: /home/kali/userpass.lst
```

```
#  # -f flag to check for existence of file.
# ! in if condition to check that the file does not exist.
# if file does not exists, "File Does Not Exist. Please Re-Enter." will be printed and this same function
#  will be repeated.
if [ ! -f "$userlistPath" ];
then
        echo -e "${RED}File Does Not Exist.${NONE} Please Re-Enter."
        echo
        defUserList
fi
}
```

Result if user inputs file that does not exists or wrong file path.



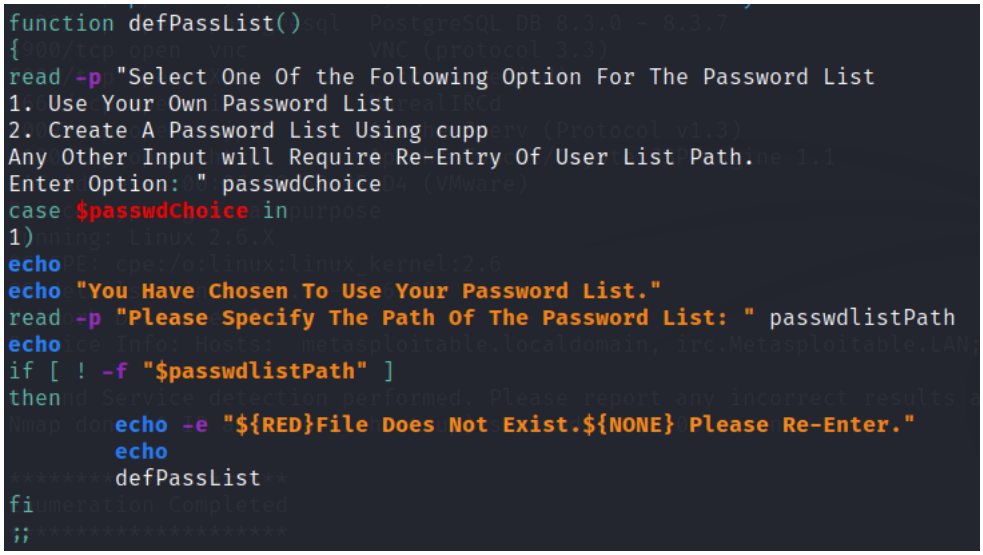
defPassList function

The defPassList function gives user the option to use an existing password list or generate a new password list for weak password check/brute force.

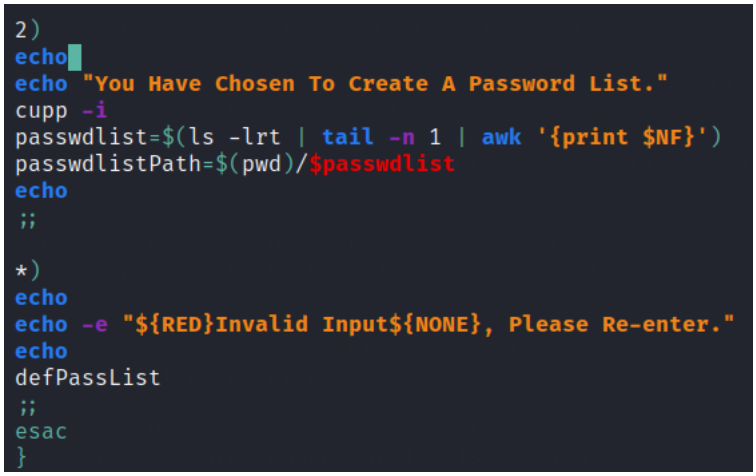
If user chose option 1 to use and existing password list, the file path of the password list is required. There will be a check to ensure the file exists before proceeding to the next action. If the file does not exist, this function will be repeated.

If user chose option 2 to generate as password list, the cupp command will be executed for password list generation and will proceed to the next action.

If user input any other options, “Invalid Input” will be printed and this function will be repeated.



```
function defPassList()
{
read -p "Select One Of the Following Option For The Password List
1. Use Your Own Password List
2. Create A Password List Using cupp
Any Other Input will Require Re-Entry Of User List Path.
Enter Option: " passwdChoice
case $passwdChoice in
1)
echo
echo "You Have Chosen To Use Your Password List."
read -p "Please Specify The Path Of The Password List: " passwdlistPath
echo
if [ ! -f "$passwdlistPath" ]
then
        echo -e "${RED}File Does Not Exist.${NONE} Please Re-Enter."
        echo
        defPassList
fi
;;
```



```
2)
echo
echo "You Have Chosen To Create A Password List."
cupp -i
passwdlist=$(ls -lrt | tail -n 1 | awk '{print $NF}')
passwdlistPath=$(pwd)/$passwdlist
echo
;;

*)
echo
echo -e "${RED}Invalid Input${NONE}, Please Re-enter."
echo
defPassList
;;
esac
}
```

Define function name defPassList.

Print "Select One Of the Following Option For The Password List

1. Use Your Own Password List

2. Create A Password List Using cupp

Any Other Input will Require Re-Entry Of User List Path.

Enter Option: " and waits for user input to be stored in passwdChoice variable.

function defPassList()

{

read -p "Select One Of the Following Option For The Password List

1. Use Your Own Password List

2. Create A Password List Using cupp

Any Other Input will Require Re-Entry Of User List Path.

Enter Option: " passwdChoice

```
Select One Of the Following Option For The Password List
1. Use Your Own Password List
2. Create A Password List Using cupp
Any Other Input will Require Re-Entry Of User List Path.
Enter Option: 1
```

Start of case statement, using passwdChoice variable as the selector.

if passwdChoice variable is '1' "You Have Chosen To Use Your Password List." "Please Specify The
Path Of The Password List: " will be printed on the terminal.

User input will be stored into passwdlistPath variable.

case $passwdChoice in

1)

echo

echo "You Have Chosen To Use Your Password List."

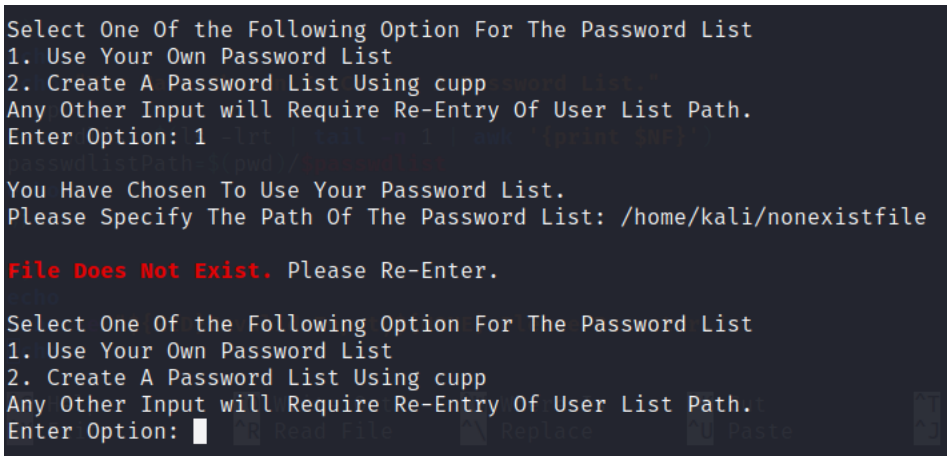
read -p "Please Specify The Path Of The Password List: " passwdlistPath

echo

```
Select One Of the Following Option For The Password List
1. Use Your Own Password List
2. Create A Password List Using cupp
Any Other Input will Require Re-Entry Of User List Path.
Enter Option: 1

You Have Chosen To Use Your Password List.
Please Specify The Path Of The Password List: /home/kali/userpass.lst
```

```
#  -f flag to check for existence of file.

# ! in if condition to check that the file does not exist.

# if file does not exists, "File Does Not Exist. Please Re-Enter." will be printed and this same function
#  will be repeated.

if [ ! -f "$passwdlistPath" ];

then

        echo "File Does Not Exist. Please Re-Enter."

        echo

        defPassList

fi

;;
```

Result if user inputs password file that does not exists or wrong file path.



if passwdChoice variable is ‘2’ "You Have Chosen To Create A Password List." will be printed on the
terminal.

Cupp interactive command will be executed, a password list will be generated based on the
answers input by user.

ls -lrt | --sort files in current directory in list format, by time oldest first and pass the result to next
action.

tail -n 1 | --grep the last line and pass the result to next action.

awk '{print $NF}'--print the last word which is the filename and store in passwdlist variable.

print present working directory + passwdlist variable and store in passwdlistPath variable.

2)

echo

echo "You Have Chosen To Create A Password List."

cupp -i

passwdlist=$(ls -lrt | tail -n 1 | awk '{print $NF}')

passwdlistPath=$(pwd)/$passwdlist

echo

;;

```
Select One Of the Following Option For The Password List
1. Use Your Own Password List
2. Create A Password List Using cupp
Any Other Input will Require Re-Entry Of User List Path.
Enter Option: 2

You Have Chosen To Create A Password List.

  cupp.py!                  # Common
     \                      # User
      \                     # Passwords
       \  (oo)____          # Profiler
          (__)    )\
             ||--||  *      [ Muris Kurgas | j0rgan@remote-exploit.org ]
                            [ Mebus | https://github.com/Mebus/]


[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: 1
> Surname:
> Nickname:
> Birthdate (DDMMYYYY):
```

```
> Partners) name:
> Partners) nickname:
> Partners) birthdate (DDMMYYYY):


> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):

> Pet's name:
> Company name:


> Do you want to add some key words about the victim? Y/[N]:
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]:

[+] Now making a dictionary ...
[+] Sorting list and removing duplicates ...
[+] Saving dictionary to 1.txt, counting 13 words.
[+] Now load your pistolero with 1.txt and shoot! Good luck!

****************************************
Brute Force In Progress. Please Be Patient.
Results Will Be Printed.
****************************************
```

if passwdChoice variable neither ‘1’ nor ‘2’ " Invalid Input" in red will be printed follow by "Please
Re-enter."

defPassList function be will executed again.

*)

echo

echo -e "${RED}Invalid Input${NONE}, Please Re-enter."

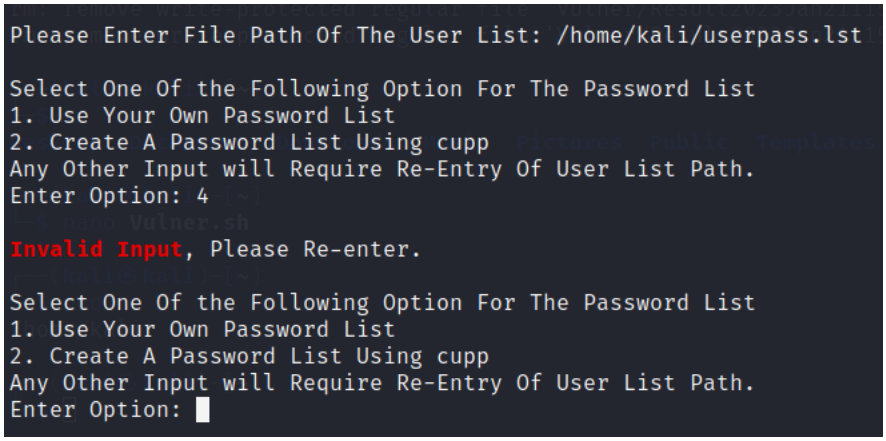
echo

defPassList

;;

esac

}



passwdCheck function

The passwdCheck function performs weak password check/brute force on the IP in EnumeratedIP.lst file using hydra.It uses user name list and password list decided by user from the defUserlist and defPassList functions.

The service and ports of each IP is retrieved from the nmap scan results saved during the enumDevice function.

After brute force execution, the result will be printed on terminal and saved in the individual report for each IP/device.

```
function passwdCheck()
{
echo "****************************************"
echo "Brute Force In Progress. Please Be Patient."
echo "Results Will Be Printed.                "
echo "****************************************"
for x in $(cat EnumerateIp.lst)
do
        hydraIp=$(cat $x.nmap | grep 'scan report' | awk '{print  $NF}')
        hydraPort=$(cat $x.nmap | grep open | grep -E 'ftp|ssh|telnet' | head -n 1 | awk -F/ '{print $1}')
        hydraService=$(cat $x.nmap | grep open | grep -E 'ftp|ssh|telnet' | head -n 1 | awk '{print $3}')
        echo -e "\n\n${BOLD}Brute Force Results:${NONE} " >> Individual_Report_$x.txt
        hydra $hydraIp -s $hydraPort $hydraService -L $userlistPath -P $passwdlistPath -o hydra_$hydraIp.txt >> Individual_Report_$x.txt
        if [ $(cat hydra_$hydraIp.txt | grep login | wc -c) == 0 ]
        then
                echo >> Individual_Report_$x.txt
                echo -e "${GREEN}No Valid Passwords Found for $hydraIp${NONE}"
                echo -e "${GREEN}No Valid Passwords Found for $hydraIp${NONE}" >> Individual_Report_$x.txt
                echo -e "\n" >> Individual_Report_$x.txt
        else
                echo >> Individual_Report_$x.txt
                echo -e "${RED}Valid Passwords Found for $hydraIp${NONE}"
                echo -e "${RED}Valid Passwords Found for $hydraIp${NONE}" >> Individual_Report_$x.txt
                echo -e "\n" >> Individual_Report_$x.txt
        fi
done
end=$(date +%s)
echo
echo "************************"
echo "Password Check Completed"
echo "************************"
echo -e "\n\n\n\n"
sleep 5
}
```

Define function name passwdCheck.

Prints "Brute Force In Progress. Please Be Patient. "Results Will Be Printed." on terminal to notify
user of current status.

function passwdCheck()

{

*echo "**"*

echo "Brute Force In Progress. Please Be Patient."

echo "Results Will Be Printed.   "

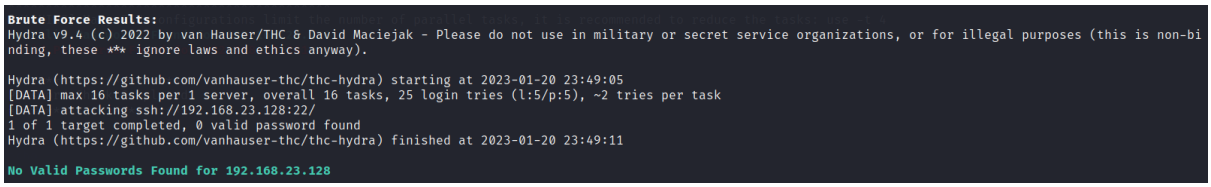
*echo "**"*

```
****************************************
Brute Force In Progress. Please Be Patient.
Results Will Be Printed.
****************************************
```

```
# Define function name passwdCheck.

# For loop to perform repeated actions on each IP address in EnumerateIp.lst.

for x in $(cat EnumerateIp.lst)

do


# cat $x.nmap --print the nmap scan result of IP address then pass to next action.

# grep 'scan report' | --find the word ‘scan report’ and pass the line containing the word to next
#  action.

# awk '{print  $NF}' -- to print the last word of the line, which is the IP address and store to hydraIp
#  variable.

        hydraIp=$(cat $x.nmap | grep 'scan report' | awk '{print  $NF}')


# cat $x.nmap --print the nmap scan result of IP address then pass to next action.

# grep 'open' | --find the word ‘open’ and pass the line containing the word to next action.

# grep -E 'ftp|ssh|telnet' | --use interpret patterns as extended regular expression to use search for
#  multiple words(ftp,ssh,telnet) then pass the found results to next action.

# head -n 1 | --retrieve the first result/service found then pass to next action.

#  awk -F/ '{print $1}' --using / as separator, print the first word and store in variable hydraPort.

        hydraPort=$(cat $x.nmap | grep open | grep -E 'ftp|ssh|telnet' | head -n 1 | awk -F/ '{print
$1}')


# cat $x.nmap --print the nmap scan result of IP address then pass to next action.

# grep 'open' | --find the word ‘open’ and pass the line containing the word to next action.

# grep -E 'ftp|ssh|telnet' | --use interpret patterns as extended regular expression to use search for
#  multiple words(ftp,ssh,telnet) then pass the found results to next action.

# head -n 1 | --retrieve the first result/service found then pass to next action.

#  awk -F/ '{print $3}' --using / as separator, print the third word and store in variable hydraService.

        hydraService=$(cat $x.nmap | grep open | grep -E 'ftp|ssh|telnet' | head -n 1 | awk '{print
$3}')
```

Append “Brute Force Results:” in bold fonts to Individual Report of each IP/device.

echo -e "\n\n${BOLD}Brute Force Results:${NONE} " >> Individual_Report_$x.txt



Perform hydra using IP stored in hydraIP variable, -s to specify the port stored in hydraPort variable, service stored in hydraService variable.

-L to specify the user name list stored in userlistPath variable.

-P to specify the password list stored in passwdlistPath variable.

-o to save the result to hydra_ file and append the same result to individual report of each
ip/device.

hydra $hydraIp -s $hydraPort $hydraService -L $userlistPath -P $passwdlistPath -o hydra_$hydraIp.txt >> Individual_Report_$x.txt

if statement to check whether weak password is present in any IP.

cat hydra_$hydraIp.txt | --open hydra result and pass it to next action.

grep login | --search for the term login then pass it to next action.

wc -c --perform a character count to check whether count is 0

if count is 0, print “No Valid Passwords Found for” for that IP in green colour fonts, also append the
result to that IP’s individual report file.

if count is not 0, print “Valid Passwords Found for” for that IP in red colour fonts, also append the
result to that IP’s individual report file.

if [$(cat hydra_$hydraIp.txt | grep login | wc -c) == 0]

then

echo >> Individual_Report_$x.txt

echo -e "${GREEN}No Valid Passwords Found for $hydraIp${NONE}"

echo -e "${GREEN}No Valid Passwords Found for $hydraIp${NONE}" >> Individual_Report_$x.txt

echo -e "\n" >> Individual_Report_$x.txt

else

echo >> Individual_Report_$x.txt

echo -e "${RED}Valid Passwords Found for $hydraIp${NONE}"

```
        echo -e "${RED}Valid Passwords Found for $hydraIp${NONE}" >> Individual_Report_$x.txt

        echo -e "\n" >> Individual_Report_$x.txt

    fi

done
```

```
No Valid Passwords Found for 192.168.23.128
Valid Passwords Found for 192.168.23.141
```

```
Brute Force Results:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bi
nding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-01-20 23:49:05
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking ssh://192.168.23.128:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-01-20 23:49:11

No Valid Passwords Found for 192.168.23.128
```

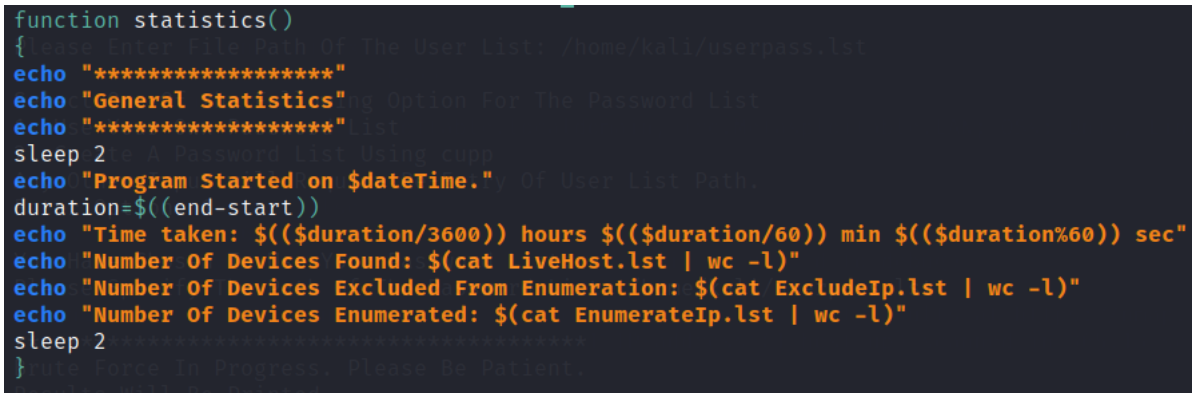
```
Brute Force Results:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bi
nding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-01-20 23:49:11
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking ftp://192.168.23.141:21/
[21][ftp] host: 192.168.23.141   login: service   password: service
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-01-20 23:49:19

Valid Passwords Found for 192.168.23.141
```

```
# store unix time in seconds into end variable, will be used later for duration calculation.

end=$(date +%s)


# Print "Password Check Completed" on terminal to update user of current status.

echo

echo "************************"

echo "Password Check Completed"

echo "************************"

echo -e "\n\n\n"

sleep 5

}
```

```
No Valid Passwords Found for 192.168.23.128
Valid Passwords Found for 192.168.23.141

************************
Password Check Completed
************************
```

statistics function

The statistics function will print out general statistics of date and time of script execution, duration starting from script execution to the end of hydra brute force, the number of devices found on LAN, the number of devices excluded from enumeration and the number of devices enumerated.



Define function name statistics.

Print "General Statistics" to notify user of the information printed.

function statistics()

{

*echo "******************"*

echo "General Statistics"

*echo "******************"*

sleep 2

Print "Program Started on” together with dateTime variable to notify user the date and time this
script is executed.

echo "Program Started on $dateTime."

Store the difference between end(from passwdCheck function) and start(from networkScan
function) variable and store the result in duration variable.

Print the time taken, using the duration variable divide by 3600 for the hours, duration variable
divide by 60 for the minutes and the remainder of duration variable divide by 60 for the seconds.

duration=$((end-start))

echo "Time taken: $(($duration/3600)) hours $(($duration/60)) min $(($duration%60)) sec"

Print "Number Of Devices Found:|" together with the number of lines found in LiveHost.lst file to
notify user the number of devices found on LAN.

echo "Number Of Devices Found: $(cat LiveHost.lst | wc -l)"

Print "Number Of Devices Excluded From Enumeration: " together with the number of lines found
in ExcludeIp.lst file to notify user the number devices excluded from enumeration.

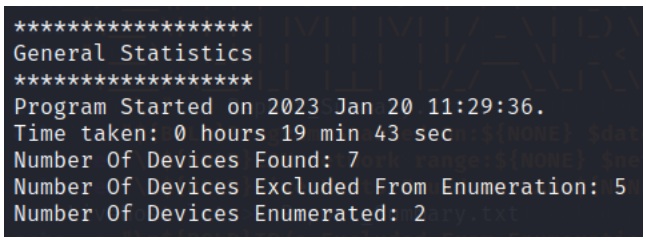
echo "Number Of Devices Excluded From Enumeration: $(cat ExcludeIp.lst | wc -l)"

Print "Number Of Devices Enumerated: " together with the number of lines found in
EnumerateIp.lst file to notify user the number devices enumerated.

echo "Number Of Devices Enumerated: $(cat EnumerateIp.lst | wc -l)"

sleep 2

}



reportSum function

The reportSum function compiles the report summary and appends individual IP/device reports into the report summary.

```
function reportSum()
{
touch Report_Summary.txt
echo " ____  _____ ____   ___  ____ _____ " >> Report_Summary.txt
echo "|  _ \| ____|  _ \ / _ \|  _ \_   _|" >> Report_Summary.txt
echo "| |_) |  _| | |_) | | | | |_) || |" >> Report_Summary.txt
echo "|  _ <| |___|  __/| |_| |  _ < | |" >> Report_Summary.txt
echo "|_| \_\_____|_|    \___/|_| \_\|_|" >> Report_Summary.txt
echo >> Report_Summary.txt
echo " ____  _   _ __  __ __  __    _    ____  __   __">> Report_Summary.txt
echo "/ ___|| | | |  \/  |  \/  |  / \  |  _ \ \ / /" >> Report_Summary.txt
echo "\___ \| | | | |\/| | |\/| | / _ \ | |_) \ V / " >> Report_Summary.txt
echo " ___) | |_| | |  | | |  | |/ ___ \|  _ < | |  " >> Report_Summary.txt
echo "|____/ \___/|_|  |_|_|  |_/_/   \_\_| \_\|_|  " >> Report_Summary.txt
echo -e "\n" >> Report_Summary.txt
echo -e "${BOLD}Program Started On:${NONE} $dateTime" >> Report_Summary.txt
echo -e "\n${BOLD}LAN network range:${NONE} $networkRange" >> Report_Summary.txt
echo -e "\n${BOLD}Live Hosts Found on LAN:${NONE} " >> Report_Summary.txt
cat LiveHost.lst >> Report_Summary.txt
echo -e "\n${BOLD}IP/s Excluded From Enumeration:${NONE} " >> Report_Summary.txt
cat ExcludeIp.lst  >> Report_Summary.txt
echo -e "\n${BOLD}Enumerated IP/s:${NONE} " >> Report_Summary.txt
cat EnumerateIp.lst >> Report_Summary.txt
for x in $(cat EnumerateIp.lst)
do
        echo -e "\n" >> Report_Summary.txt
        echo -e "\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++=" >> Report_Summary.txt
        cat Individual_Report_$x.txt >> Report_Summary.txt
done
}
```

Define function name reportSum.

touch -- create a file named Report_Summary.txt.

Append heading “Report Summary” into Report_Summary.txt file.

echo -e "\n" --append 2 new line into Report_Summary.txt file after heading for cosmetic purpose.

function reportSum()

{

touch Report_Summary.txt

echo " ____ _____ ____   ___ ____ _____ " >> Report_Summary.txt

echo "| _ \| ____| _ \ / _ \| _ _   _|" >> Report_Summary.txt

echo "| |_) | _| | |_) | | | | |_) || |" >> Report_Summary.txt

echo "| _ <| |___| __/| |_| | _ < | |" >> Report_Summary.txt

echo "|_| ______|_| ___/|_| _\|_|" >> Report_Summary.txt

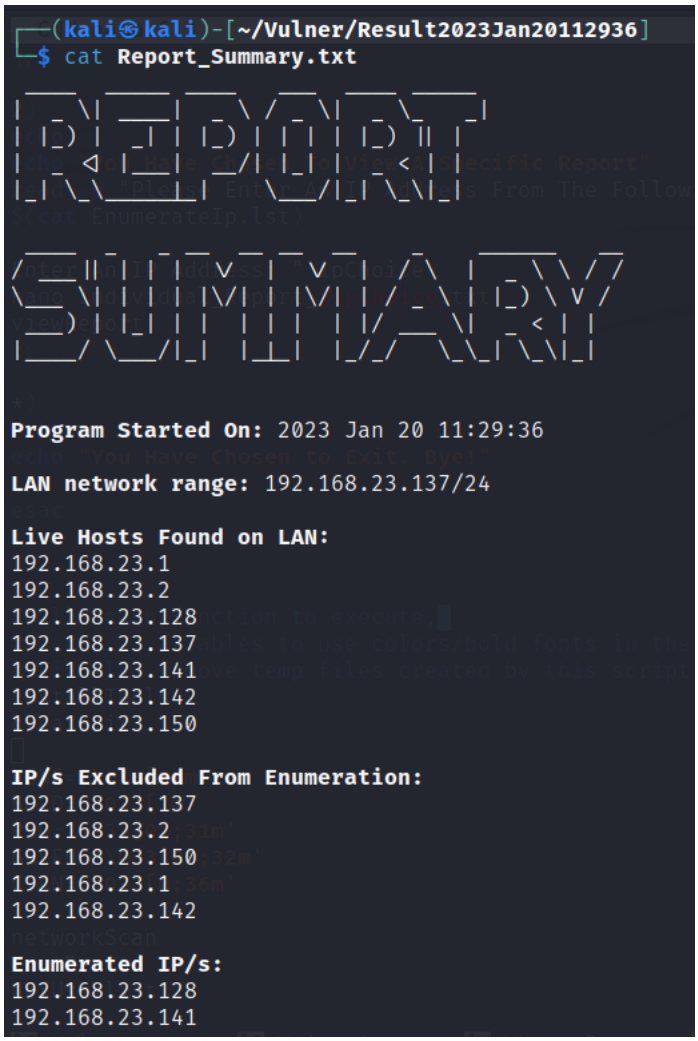
echo >> Report_Summary.txt

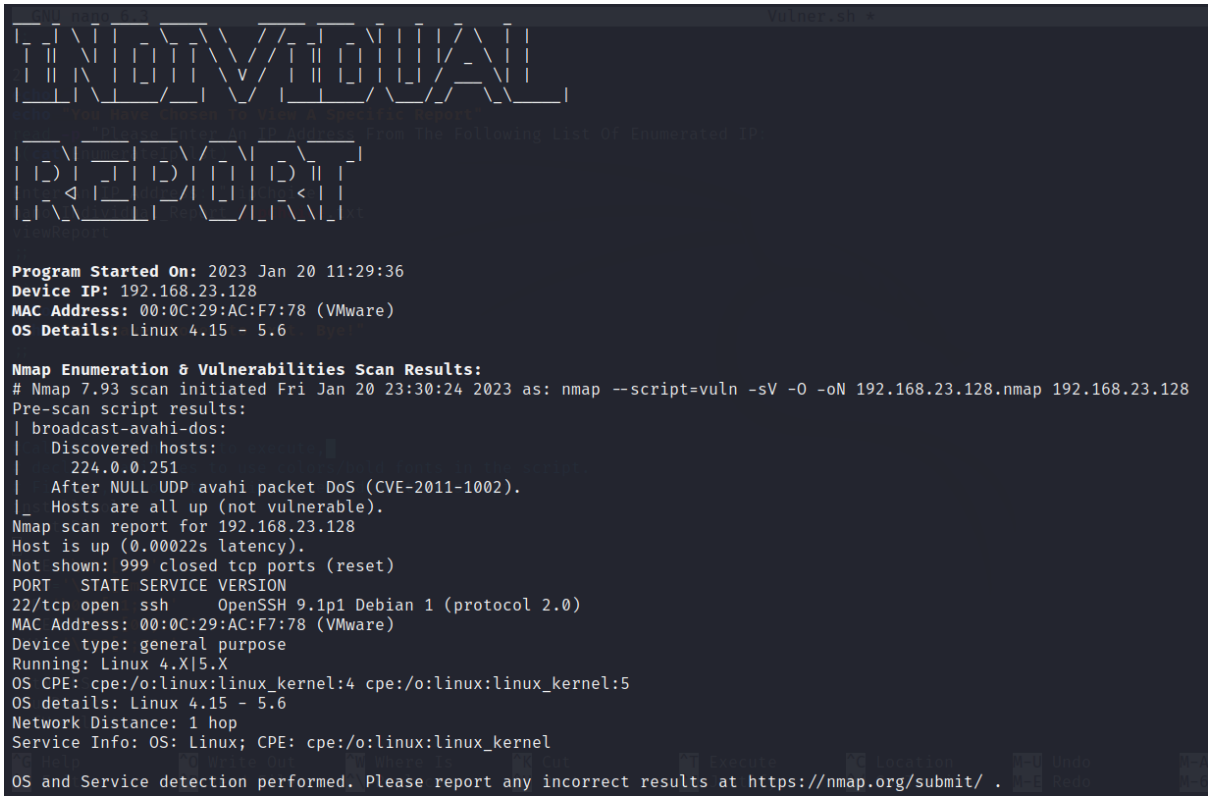
echo " ____ _   _ __ __ __ __   _   ______   __">> Report_Summary.txt

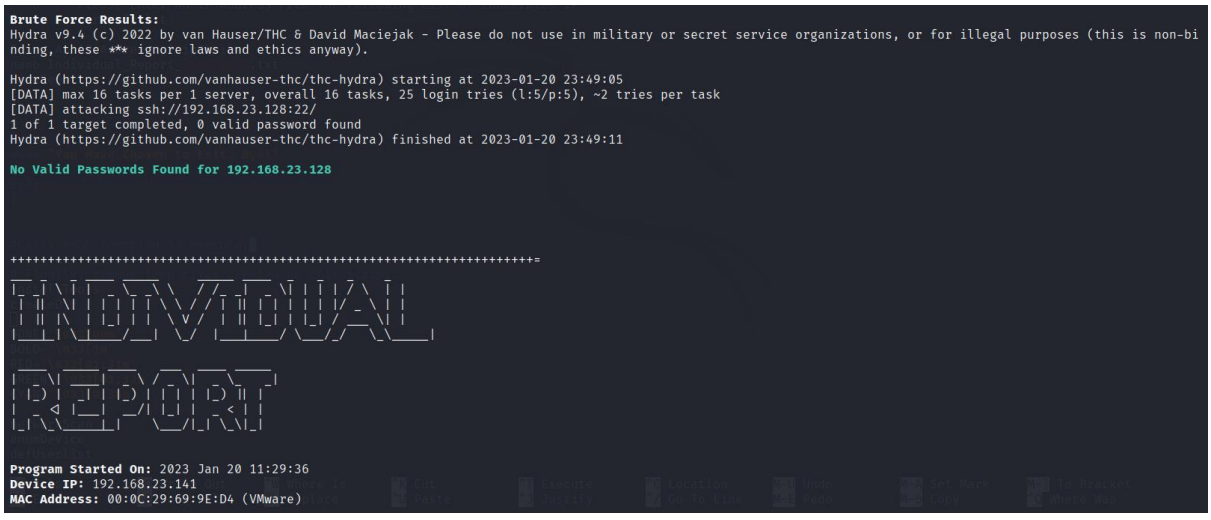
echo "/ ___|| | | | \/ | \/ | / \ | _ \ \ / /" >> Report_Summary.txt

```
echo "\___ \| | | | |V| | |V| | / _ \ | |_) \ V / " >> Report_Summary.txt

echo " ___) | |_| | |  | | |  | |/ ___ \|  _ < | |  " >> Report_Summary.txt

echo "|____/ \___/|_|  |_|_|  |_/_/   \_\_| \_\|_|  " >> Report_Summary.txt

echo -e "\n" >> Report_Summary.txt


# Append "Program Started On: " in bold with dateTime variable into Report_Summary.txt.

echo -e "${BOLD}Program Started On:${NONE} $dateTime" >> Report_Summary.txt


# Append "LAN network range: " in bold with networkRange variable into Report_Summary.txt with
#  an empty line above.

echo -e "\n${BOLD}LAN Network Range:${NONE} $networkRange" >> Report_Summary.txt


# Append "Live Hosts Found on LAN " in bold with LiveHost.lst values on the same line into
#  Report_Summary.txt with an empty line above.

echo -e "\n${BOLD}Live Hosts Found On LAN:${NONE} " >> Report_Summary.txt

cat LiveHost.lst >> Report_Summary.txt


# Append "IP/s Excluded From Enumeration " in bold with ExcludeIp.lst values on the same line into
#  Report_Summary.txt with an empty line above.

echo -e "\n${BOLD}IP/s Excluded From Enumeration:${NONE} " >> Report_Summary.txt

cat ExcludeIp.lst  >> Report_Summary.txt


# Append "Enumerated IP/s " in bold with EnumerateIp.lst values on the same line into
#  Report_Summary.txt with an empty line above.

echo -e "\n${BOLD}Enumerated IP/s:${NONE} " >> Report_Summary.txt

cat EnumerateIp.lst >> Report_Summary.txt
```

```
# For loop to execute following commands for each IP in EnumerateIp.lst file.

# echo -e "\n" --insert 2 new lines to Report_Summary.txt.

# echo -e "\n++++" --append ++ to Report_Summary.txt with new line above.

# cat Individual_Report_$x.txt >> Report_Summary.txt --appends individual report to
#  Report_Summary.txt.

for x in $(cat EnumerateIp.lst)

do

        echo -e "\n" >> Report_Summary.txt

        echo -e
"\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++" >>
Report_Summary.txt

        cat Individual_Report_$x.txt >> Report_Summary.txt

done

}
```




Program Started On: 2023 Jan 20 11:29:36
Device IP: 192.168.23.128
MAC Address: 00:0C:29:AC:F7:78 (VMware)
OS Details: Linux 4.15 - 5.6
Nmap Enumeration & Vulnerabilities Scan Results:
Nmap 7.93 scan initiated Fri Jan 20 23:30:24 2023 as: nmap --script=vuln -sV -O -oN 192.168.23.128.nmap 192.168.23.128
Pre-scan script results:
| broadcast-avahi-dos:
| Discovered hosts:
| 224.0.0.251
| After NULL UDP avahi packet DoS (CVE-2011-1002).
|_ Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.128
Host is up (0.00022s latency).
Not shown: 999 closed tcp ports (reset)
PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 9.1p1 Debian 1 (protocol 2.0)
MAC Address: 00:0C:29:AC:F7:78 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

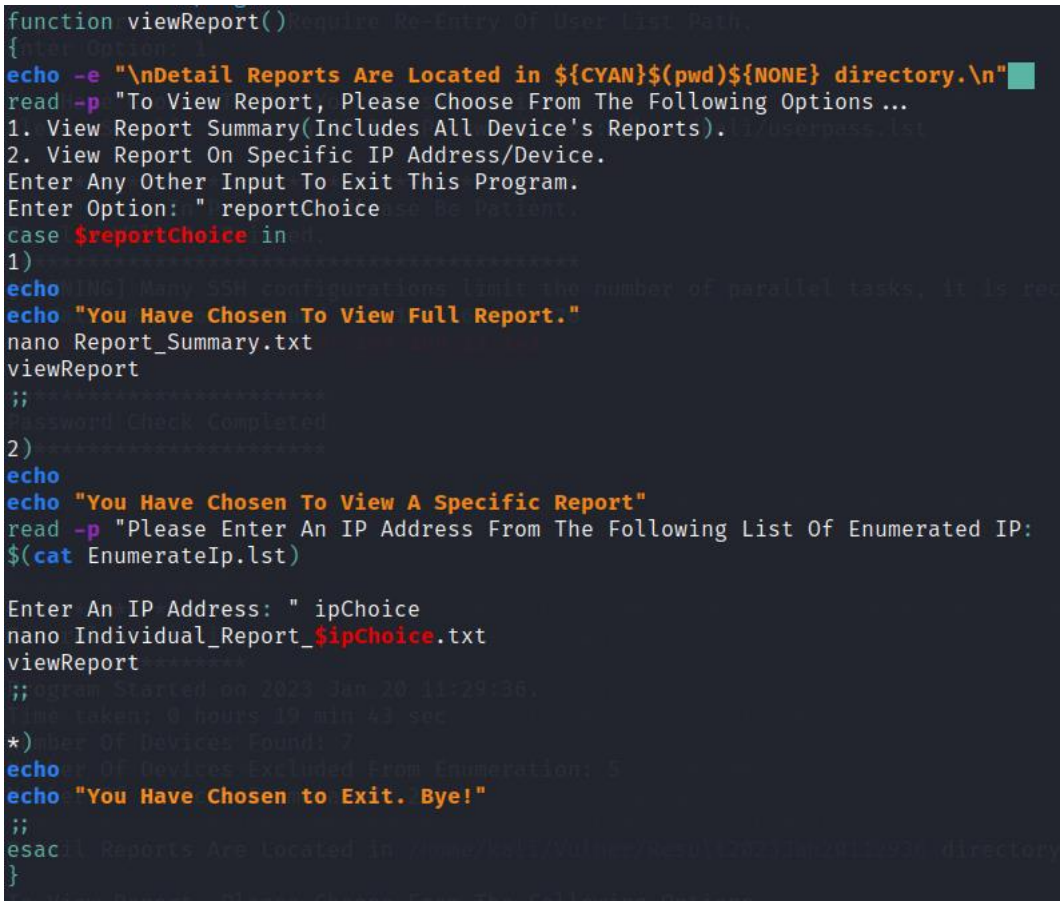

Brute Force Results:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-01-20 23:49:05
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking ssh://192.168.23.128:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-01-20 23:49:11
No Valid Passwords Found for 192.168.23.128
Program Started On: 2023 Jan 20 11:29:36
Device IP: 192.168.23.141
MAC Address: 00:0C:29:69:9E:D4 (VMware)

viewReport function

The viewReport function prints the directory of where the reports are stored and gives user a choice to view the summary report(includes individual reports) or specific IP/device report.

If user chose to view specific IP/device report, they will need to input the IP address.

Finally, if any other option is input, the program will end.



```
function viewReport()
{
echo -e "\nDetail Reports Are Located in ${CYAN}$(pwd)${NONE} directory.\n"
read -p "To View Report, Please Choose From The Following Options ...
1. View Report Summary(Includes All Device's Reports).
2. View Report On Specific IP Address/Device.
Enter Any Other Input To Exit This Program.
Enter Option: " reportChoice
case $reportChoice in
1)
echo
echo "You Have Chosen To View Full Report."
nano Report_Summary.txt
viewReport
;;
2)
echo
echo "You Have Chosen To View A Specific Report"
read -p "Please Enter An IP Address From The Following List Of Enumerated IP:
$(cat EnumerateIp.lst)

Enter An IP Address: " ipChoice
nano Individual_Report_$ipChoice.txt
viewReport
;;
*)
echo
echo "You Have Chosen to Exit. Bye!"
;;
esac
}
```

```
# Define function name viewReport.

# Prints "Detail Reports Are Located in" with directory name and empty line before and after this
#  sentence.

# Prints "To View Report, Please Choose From The Following Options...

#  1. View Report Summary(Includes All Device's Reports).

#  2. View Report On Specific IP Address/Device.

#  Enter Any Other Input To Exit This Program.

#  Enter Option: " then expects an input from user to store in reportChoice variable.

function viewReport()

{

echo -e "\nDetail Reports Are Located in ${CYAN}$(pwd)${NONE} directory.\n"

read -p "To View Report, Please Choose From The Following Options...

1. View Report Summary(Includes All Device's Reports).

2. View Report On Specific IP Address/Device.

Enter Any Other Input To Exit This Program.

Enter Option: " reportChoice
```

```
Detail Reports Are Located in /home/kali/Vulner/Result2023Jan20112936 directory.

To View Report, Please Choose From The Following Options...
1. View Report Summary(Includes All Device's Reports).
2. View Report On Specific IP Address/Device.
Enter Any Other Input To Exit This Program.
Enter Option:
```

```
# Start of case statement, reads the reportChoice variable.
# If reportChoice variable is “1” it will print the Report_Summary.txt in terminal.
# Then, the viewReport function will be executed again.
case $reportChoice in
1)
echo
echo "You Have Chosen To View Full Report."
cat Report_Summary.txt
viewReport
;;
```

```
Detail Reports Are Located in /home/kali/Vulner/Result2023Jan23120846 directory.

To View Report, Please Choose From The Following Options ...
1. View Report Summary(Includes All Device's Reports).
2. View Report On Specific IP Address/Device.
Enter Any Other Input To Exit This Program.
Enter Option: 1

You Have Chosen To View Full Report.
 ____  _____ ____   ___  ____ _____
|  _ \| ____|  _ \ / _ \|  _ \_   _|
| |_) |  _| | |_) | | | | |_) || |
|  _ <| |___|  __/| |_| |  _ < | |
|_| \_\_____|_|    \___/|_| \_\|_|

 ____  _   _ __  __ __  __    _    ____  __
/ ___|| | | |  \/  |  \/  |  / \  |  _ \ \ / /
\___ \| | | | |\/| | |\/| | / _ \ | |_) \ V /
 ___) | |_| | |  | | |  | |/ ___ \|  _ < | |
|____/ \___/|_|  |_|_|  |_/_/   \_\_| \_\|_|


Program Started On: 2023 Jan 23 12:08:46

LAN Network Range: 192.168.23.137/24

Live Hosts Found On LAN:
192.168.23.1
192.168.23.2
192.168.23.128
192.168.23.137
192.168.23.141
192.168.23.142
192.168.23.150

IP/s Excluded From Enumeration:
192.168.23.137
192.168.23.2
192.168.23.1
192.168.23.150
```

```
Enumerated IP/s:
192.168.23.128
192.168.23.141



+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ___ _   _ ____ _____     _____ ____  _   _   _    _     
|_ _| \ | |  _ \_ _\ \   / /_ _|  _ \| | | | / \  | |    
 | ||  \| | | | | | \ \ / / | || | | | | | |/ _ \ | |    
 | || |\  | |_| | |  \ V /  | || |_| | |_| / ___ \| |___ 
|___|_| \_|____/___|  \_/  |___|____/ \___/_/   \_\_____|

 ____  _____ ____   ___  ____ _____ 
|  _ \| ____|  _ \ / _ \|  _ \_   _|
| |_) |  _| | |_) | | | | |_) || |  
|  _ <| |___|  __/| |_| |  _ < | |  
|_| \_\_____|_|    \___/|_| \_\|_|  


Program Started On: 2023 Jan 23 12:08:46
Device IP: 192.168.23.128
MAC Address: 00:0C:29:AC:F7:78 (VMware)
OS Details: Linux 4.15 - 5.6

Nmap Enumeration & Vulnerabilities Scan Results:
# Nmap 7.93 scan initiated Mon Jan 23 12:09:07 2023 as: nmap --script=vuln -sV -O -oN 192.168.23.128.nmap 192.168.23.128
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.128
Host is up (0.00020s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.1p1 Debian 1 (protocol 2.0)
MAC Address: 00:0C:29:AC:F7:78 (VMware)
Device type: general purpose
```

```
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Mon Jan 23 12:09:45 2023 -- 1 IP address (1 host up) scanned in 37.97 seconds


Brute Force Results:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bi
nding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-01-23 12:27:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking ssh://192.168.23.128:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-01-23 12:27:13

No Valid Passwords Found for 192.168.23.128




+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ___ _   _ ____ _____     _____ ____  _   _   _    _     
|_ _| \ | |  _ \_ _\ \   / /_ _|  _ \| | | | / \  | |    
 | ||  \| | | | | | \ \ / / | || | | | | | |/ _ \ | |    
 | || |\  | |_| | |  \ V /  | || |_| | |_| / ___ \| |___ 
|___|_| \_|____/___|  \_/  |___|____/ \___/_/   \_\_____|

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

```
Program Started On: 2023 Jan 23 12:08:46
Device IP: 192.168.23.141
MAC Address: 00:0C:29:69:9E:D4 (VMware)
OS Details: Linux 2.6.9 - 2.6.33

Nmap Enumeration & Vulnerabilities Scan Results:
# Nmap 7.93 scan initiated Mon Jan 23 12:09:46 2023 as: nmap --script=vuln -sV -O -oN 192.168.23.141.nmap 192.168.23.141
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.141
Host is up (0.00041s latency).
Not shown: 977 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2011-2523  BID:48539
```

If reportChoice variable is “2”, "You Have Chosen To View A Specific Report" will be printed.

Follow by "Please Enter An IP Address From The Following List Of Enumerated IP:” with the values
in EnumerateIp.lst file. User input will be stored in ipChoice variable.

2)

echo

echo "You Have Chosen To View A Specific Report"

read -p "Please Enter An IP Address From The Following List Of Enumerated IP:

$(cat EnumerateIp.lst)

Enter An IP Address: " ipChoice

echo

```
You Have Chosen To View A Specific Report
Please Enter An IP Address From The Following List Of Enumerated IP:
192.168.23.128
192.168.23.141

Enter An IP Address: 10
```

```
# -f flag to check for existence of file.

# ! in if condition to check that the file does not exist.

# While file does not exists, "File Does Not Exist. Please Re-Enter." will be printed and "Please Enter
#  An IP Address From The Following List Of Enumerated IP:" with the values in EnumerateIp.lst file.
#  User input will be stored in ipChoice variable.

# While file exists, the file will be printed on terminal, and viewReport function will be called.

while [ ! -f "Individual_Report_$ipChoice.txt" ];

do

        echo -e "${RED}File Does Not Exist.${NONE} Please Re-Enter IP Address."

        echo

        read -p "Please Enter An IP Address From The Following List Of Enumerated IP:

$(cat EnumerateIp.lst)


Enter An IP Address: " ipChoice

        echo

done

cat Individual_Report_$ipChoice.txt

viewReport

;;
```

```
You Have Chosen To View A Specific Report
Please Enter An IP Address From The Following List Of Enumerated IP:
192.168.23.128
192.168.23.141

Enter An IP Address: 10
File Does Not Exist. Please Re-Enter IP Address.

Please Enter An IP Address From The Following List Of Enumerated IP:
192.168.23.128
192.168.23.141
```

When correct IP is entered to view Individual Report.

```
Detail Reports Are Located in /home/kali/Vulner/Result2023Jan23120846 directory.

To View Report, Please Choose From The Following Options ...
1. View Report Summary(Includes All Device's Reports).
2. View Report On Specific IP Address/Device.
Enter Any Other Input To Exit This Program.
Enter Option: 2

You Have Chosen To View A Specific Report
Please Enter An IP Address From The Following List Of Enumerated IP:
192.168.23.128
192.168.23.141

Enter An IP Address: 192.168.23.128

 ___ _   _ ____ _____     _____ ____ _   _  _   _
|_ _| \ | |  _ \_ _\ \   / /_ _|  _ \| | | |/ \  | |
 | ||  \| | | | | | \ \ / / | || |_) | | | |/ _ \ | |
 | || |\  | |_| | |  \ V /  | ||  _ <| |_| / ___ \| |
|___|_| \_|____/___|  \_/  |___|_| \_\\___/_/   \_\____|

 ____  _____ ____   ___  ____ _____
|  _ \| ____|  _ \ / _ \|  _ \_   _|
| |_) |  _| | |_) | | | | |_) || |
|  _ <| |___|  __/| |_| |  _ < | |
|_| \_\_____|_|    \___/|_| \_\|_|


Program Started On: 2023 Jan 23 12:08:46
Device IP: 192.168.23.128
MAC Address: 00:0C:29:AC:F7:78 (VMware)
OS Details: Linux 4.15 - 5.6

Nmap Enumeration & Vulnerabilities Scan Results:
# Nmap 7.93 scan initiated Mon Jan 23 12:09:07 2023 as: nmap --script=vuln -sV -O -oN 192.168.23.128.nmap 192.168.23.128
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|     224.0.0.251
|   After NULL UDP avahi packet DoS (CVE-2011-1002).
```

```
|_  Hosts are all up (not vulnerable).
Nmap scan report for 192.168.23.128
Host is up (0.00020s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.1p1 Debian 1 (protocol 2.0)
MAC Address: 00:0C:29:AC:F7:78 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Mon Jan 23 12:09:45 2023 -- 1 IP address (1 host up) scanned in 37.97 seconds


Brute Force Results:
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bi
nding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-01-23 12:27:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 25 login tries (l:5/p:5), ~2 tries per task
[DATA] attacking ssh://192.168.23.128:22/
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-01-23 12:27:13

No Valid Passwords Found for 192.168.23.128



Detail Reports Are Located in /home/kali/Vulner/Result2023Jan23120846 directory.
```

If reportChoice variable is not “1” or “2”, "You Have Chosen to Exit. Bye!" will be printed and the script will end.

*)

echo

echo "You Have Chosen to Exit. Bye!"

;;

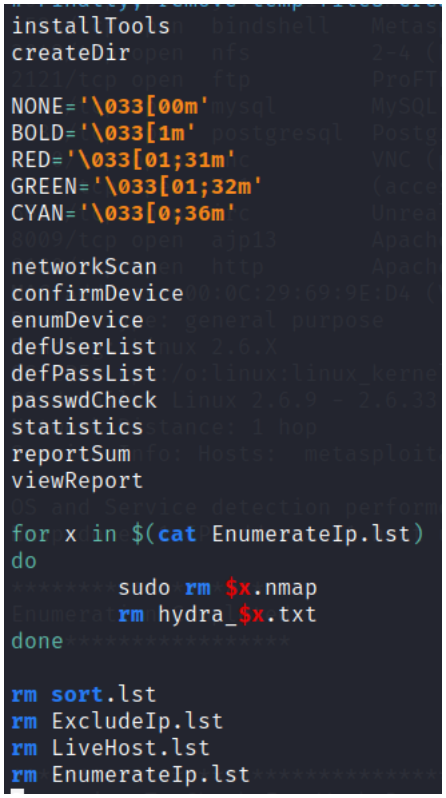
esac

}

```
Detail Reports Are Located in /home/kali/Vulner/Result2023Jan21123800 directory.

To View Report, Please Choose From The Following Options ...
1. View Report Summary(Includes All Device's Reports).
2. View Report On Specific IP Address/Device.
Enter Any Other Input To Exit This Program.
Enter Option: 5

You Have Chosen to Exit. Bye!
```

Global scope code

These codes are used to call the functions, declare variables and delete files towards the end of the script.



```
installTools
createDir

NONE='\033[00m'
BOLD='\033[1m'
RED='\033[01;31m'
GREEN='\033[01;32m'
CYAN='\033[0;36m'
networkScan
confirmDevice
enumDevice
defUserList
defPassList
passwdCheck
statistics
reportSum
viewReport
for x in $(cat EnumerateIp.lst)
do
        sudo rm $x.nmap
        rm hydra_$x.txt
done

rm sort.lst
rm ExcludeIp.lst
rm LiveHost.lst
rm EnumerateIp.lst
```

Call the installTools function.

installTools

Call the createDir function.

createDir

Declare each global variable according to their colour code.

NONE='\033[00m'

BOLD='\033[1m'

RED='\033[01;31m'

GREEN='\033[01;32m'

CYAN='\033[0;36m'

Call the networkScan function.

networkScan

Call the enumDevice function.

enumDevice

Call the defUserlist function.

defUserlist

Call the defPassList function.

defPassList

Call the passwdCheck function.

passwdCheck

Call the statistics function.

statistics

Call the reportSum function.

reportSum

Call the viewReport function.

viewReport

```
# Create a For loop by using the IP that was enumerated.
# Delete nmap result temp file for every IP that was enumerated.
# Delete hydra result temp file for every IP that was enumerated.
for x in $(cat EnumerateIp.lst)
do
        sudo rm $x.nmap
        rm hydra_$x.txt
done

# Delete temp file ExcludeIp.lst
rm ExcludeIp.lst

# Delete temp file LiveHost.lst
rm LiveHost.lst

# Delete temp file EnumerateIp.lst
rm EnumerateIp.lst
```

Reference

Linux ip Command Examples

https://www.cyberciti.biz/faq/linux-ip-command-examples-usage-syntax/

How to Echo Newline in Bash

https://linuxhint.com/echo-newline-bash/

Linux / UNIX Shell: Sort IP Address

https://www.cyberciti.biz/faq/unix-linux-shell-script-sorting-ip-addresses/

Sorting multiple columns, with the second column being sorted by numerical order [duplicate]

https://unix.stackexchange.com/questions/419424/sorting-multiple-columns-with-the-second-column-being-sorted-by-numerical-order

How to change the output color of echo in Linux

https://stackoverflow.com/questions/5947742/how-to-change-the-output-color-of-echo-in-linux

End