Penetration Testing Report

Name : Low Hong Jun

Student Code : S13

Penetration Testing

Project Structure

1. Getting the User Input

1.1 Get from the user a network to scan.

1.2 Get from the user a name for the output directory.

1.3 Allow the user to choose 'Basic' or 'Full'.

1.3.1 Basic: scans the network for TCP and UDP, including the service version and weak passwords.

1.3.2 Full: include Nmap Scripting Engine (NSE), weak passwords, and vulnerability analysis.

1.4 Make sure the input is valid.

2. Weak Credentials

2.1 Look for weak passwords used in the network for login services.

2.1.1 Have a built-in password.lst to check for weak passwords.

2.1.2 Allow the user to supply their own password list.

2.2 Login services to check include: SSH, RDP, FTP, and TELNET.

3. Mapping Vulnerabilities

3.1 Mapping vulnerabilities should only take place if Full was chosen.

3.2 Display potential vulnerabilities via NSE and Searchsploit.

4. Log Results

4.1 During each stage, display the stage in the terminal.

4.2 At the end, show the user the found information.

4.3 Allow the user to search inside the results.

4.4 Allow to save all results into a Zip file.

5. Creativity

This project mainly uses bash for its script execution

1.1 Get from the user a network to scan.

```
    # Get the user input for IP address

echo -e
echo -n "Please enter an IP address : "; read ip
echo -e

    # Use a regular expression to check if the input is a valid IP address

if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then

    # Use an array to store the octets of the IP address
    IFS='.' read -ra octets <<< "$ip"

    # Check if each octet is between 0 and 255

    for octet in "${octets[@]}";

    do

        if [ $octet -lt 0 ] || [ $octet -gt 255 ];

            then
            echo "$ip is not a valid IP address. Please re-enter your target IP address."

            exit

        fi

    done

    echo "The IP address $ip is valid and will be use for scanning the target..."

else

    echo "$ip is not a valid IP address. Please re-enter your target IP address."

    exit

fi
```

Getting input from user

If user input follows this pattern then proceed to next step

Store the ip address octet using "." as a field separator

Checking if the octets contains intergers between 0 and 255

If octet is lesser than 0, greater than 255, it is not valid input, exit script

If between 0-255, it will be a valid input and will echo message to user

Continuation from first IF statement, If IP does not follow the octet pattern, it will also be invalid and exit

Script 1.1

This piece of code is retrieve the user input and do validation to make sure it is a valid network address. It does this by making sure it follow the ip addresss octet pattern, e.g. 192.168.xxx.xxx.

If this is valid, it will then store each octet using the "." period as a field separator and store it in a array. The script will then loop through the array and check if the numbers in the octet is lesser than 0 and greater 255.

If the condition is valid, the ip address will be used. If not, the script will exit.

```
Please enter an IP address : 192.168.155.131

The IP address 192.168.155.131 is valid and will be use for scanning the target...
```

Terminal 1.1a

Valid input, script will print message that ip address is ready to be use for scanning.

```
Please enter an IP address : 12345.12.144.222

12345.12.144.222 is not a valid IP address. Please re-enter your target IP address.
```

Terminal 1.1b

Invalid, as it does not fulfill the condition where the octet should be between 0-255

```
Please enter an IP address : fffff

fffff is not a valid IP address. Please re-enter your target IP address.
```

Terminal 1.1c

Invalid, as it does not fulfill the condition where the octet follows a pattern

1.2 Get from the user a name for the output directory

```
fi

echo -e


#1.2 Get from the user a name for the output directory.


echo "Please enter the directory you wish the results to be stored in..."
read dstdir

if [ -z "$dstdir" ]; then

    echo -n "You left your previous input blank, Please enter the directory you wish the results to be stored in..."; read dstdir

    if [ -z "$dstdir" ]; then

        echo "No input detected, automatically creating folder in current directory to stored the results..."
        mkdir -p Results
        sleep 3
        echo "Results folder created..."
        dstdir=Results
    fi

else

    mkdir -p $dstdir

fi

echo -e


#1.3 Allow the user to choose 'Basic' or 'Full'.


echo "Please choose if you wish to do a Basic or Full vulnerability scan..."

while [[ $scanchoice != "basic" && $scanchoice != "full" ]]; do

```

Read and store user input in the $dstsir variable

Read the $dstdir, if it is empty, prompt user input again

If still no user input, script will automatically create folder to save results

If user input detected, make output directory according to user input

This portion of the script, we are trying to get the user destination directory to store the results from all the scans the script will be doing.

The read command will get the input from the user and store it in a variable call dstdir. The script will then check if the dstdir variable is empty using -z

If the input is empty for 2 consecutive rows, a results folder will be created at the current file path.

If input is detected, the folder will be name and created according to the user input in to $dstdir.

```
Please enter the directory you wish the results to be stored in...
/home/kali/Desktop/Results
Results folder created...
```

Terminal 1.2a

User input creates a folder named Results according to the file path they provide

```
Please enter the directory you wish the results to be stored in...

You left your previous input blank, Please enter the directory you wish the results to be stored in...
No input detected, automatically creating folder in current directory to stored the results...
Results folder created...
```

Terminal 1.2b

User given no input and the script detected it and prompt the user again. After the 2nd time the user did not give a input, the script automatically creates a Results folder at the current file path the user is at.

1.3 Allow the user to choose 'Basic' or 'Full'.

```
#1.3 Allow the user to choose 'Basic' or 'Full'.

echo "Please choose if you wish to do a Basic or Full vulnerability scan..."

while [[ $scanchoice != "basic" && $scanchoice != "full" ]]; do

echo -n "Your choice : "; read scanchoice

scanchoice=$(echo "$scanchoice" | tr '[:upper:]' '[:lower:]')

done

echo -e
```

While loop will only accept basic and full as valid inputs, if not it will loop endlessly

Read user input and store it in the $scanchoice variable, choose between basic or full

Translate upper cases to lower case so that it will still be a valid input even if in upper case

This part of the script reads the user input and only accept basic and full as valid input due to the while loop condition. If this condition is false, not “basic” or “full”, the loop will endlessly ask the user for input till the condition is fulfill. This is to ensure the user choose one of the mode to scans.

A translate command is include to translate possible user input as upper case to lower case, so that the while loop condition can be fulfill. Without this, the while loop might take it as a false condition instead.

```
Please choose if you wish to do a Basic or Full vulnerability scan...
Your choice : full

Running Full Vulnerability Scan...
```

Terminal 1.3a

Full scan input by user, full scan script will start executing all relevant scans

```
Please choose if you wish to do a Basic or Full vulnerability scan...
Your choice : basic

Running Basic Vulnerability Scan...
```

Terminal 1.3b

Basic scan input by user, basic scan script will start executing all basic scans and output message to inform user of scan chosen.

1.3.1 Basic: scans the network for TCP and UDP, including the service version and weak passwords.

BASIC SCAN

```
if [ $scanchoice == "basic" ]; then

    mkdir -p $dstdir/Scans
    echo "Running Basic Vulnerability Scan..."
    echo -e
    echo "Running Nmap Scanning Tool..."
    nmap -sV $ip -p- >> $dstdir/Scans/TCPresults.lst
    cat $dstdir/Scans/TCPresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Nmap Scan completed..."
    echo -e
    echo "Running Masscan... Permission might be required"
    sudo masscan $ip -pU -p- --rate 500 >> $dstdir/Scans/UDPresults.lst
    cat $dstdir/Scans/UDPresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Masscan completed..."
    echo -e
```

If user input was “basic”

Nmap service scan output appended to an All_Results.lst file for later use

Masscan UDP scan output appended to All_Results.lst file for later use

FULL SCAN

```
else

    # Run FULL vulnerability scan command here

    mkdir -p $dstdir/Scans
    echo "Running Full Vulnerability Scan..."
    echo -e
    echo "Running Nmap Scanning Tool..."
    nmap -sV $ip -p- >> $dstdir/Scans/TCPresults.lst
    cat $dstdir/Scans/TCPresults.lst >> $dstdir/Scans/All_Results.lst
    echo -e
    echo "Running Vulscan on open ports..."
    nmap -sV --script=vulscan/vulscan.nse $ip >> $dstdir/Scans/Vulscanresults.lst
    cat $dstdir/Scans/Vulscanresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Nmap Scan completed..."
    echo -e
    echo "Running Masscan... Permission might be required"
    sudo masscan $ip -pU -p- --rate 500 >> $dstdir/Scans/UDPresults.lst
    cat $dstdir/Scans/UDPresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Masscan completed..."
    echo -e
```

Nmap service scan output appended to an All_Results.lst file for later use

Extra Vulnerability scan for full mode, this will find related CVE according the service version

Masscan UDP scan output appended to All_Results.lst file for later use

In the Basic and Full Scans code, the structure is around the same except for a extra vulnerability scan being executed in the Full Scan mode.

The user input will be register in $scanchoice and determine if a basic or full scan will be run. This is make possible with a IF ELSE statement separating the 2 choices.

A Nmap service scan, UDP Masscan will be performed for the basic portion.

While a Nmap service scan, UDP Masscan, and a Vulscan will be performed for the full portion.

```
Running Basic Vulnerability Scan...

Running Nmap Scanning Tool...
Nmap Scan completed...

Running Masscan... Permission might be required
[sudo] password for kali:
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2024-01-02 12:35:55 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Masscan completed...
```

Terminal 1.3.1a

Nmap and Masscan executing when basic vulnerability script is being run.

```
Running Full Vulnerability Scan...

Running Nmap Scanning Tool...

Running Vulscan on open ports...
Nmap Scan completed...

Running Masscan... Permission might be required
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2024-01-02 15:55:07 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Masscan completed...
```

Terminal 1.3.1b

Nmap, Vulscan and Masscan are executed as part of the Full Scan script.

2.1 Look for weak passwords used in the network for login services.

```
# Weak Credentials on login services

echo "Proceeding to Weak Credentials Checks, do you wish to upload your own password or user lists? *Yes/No*"

while [[ $uploadchoice != "yes" && $uploadchoice != "no" ]]; do

echo -n "Your choice : "; read uploadchoice

uploadchoice=$(echo "$uploadchoice" | tr '[:upper:]' '[:lower:]')

done

echo -e

if [ $uploadchoice == "yes" ]; then

    echo "Please specify if you will upload the USER, PASS or BOTH lists to be use for the credential checks..."
    echo "Default files will be provided if only one options is chosen..."

    while [[ $listname != "user" && $listname != "pass" && $listname != "both" ]]; do

    echo -n "Your choice : "; read listname

    listname=$(echo "$listname" | tr '[:upper:]' '[:lower:]')

    done

    echo -e
```

Read user input, check if the user wants to upload their own user, password list

Translate all UPPER case to LOWER case so while loop condition will be still true even if UPPER case is used

If user input yes, script will prompt user if they want to upload both user and password list or either one, user input will be store in $listname variable

Translate UPPER to LOWER

The next part of the script (Basic and Full Scan is the same code), it will ask the user if they want to upload their own user and password list. The while loop will only allow the script to continue if either of its condition is true. “YES and NO”.

The script will then prompt the user again to check if the user wants to upload both the user and password list or just either one of them.

After this portion, it will bring us to the next part of the script.

```
Proceeding to Weak Credentials Checks, do you wish to upload your own password or user lists? *Yes/No*
Your choice : no

Please specify if you would like to use the provided default list of USER, PASS or BOTH...
Choosing to use only one of the default list will required user to enter the needed credentials for the other required field...
Your choice : both
```

Terminal 2.1a

User input and terminal output of the script above

2.1.1 Have a built-in password.lst to check for weak passwords.

```
else

    echo "Please specify if you would like to use the provided default list of USER, PASS or BOTH..."
    echo "Choosing to use only one of the default list will required user to enter the needed credentials for the other required field..."

    while [[ $listname2 != "user" && $listname2 != "pass" && $listname2 != "both" ]]; do

    echo -n "Your choice : "; read listname2

    listname2=$(echo "$listname2" | tr '[:upper:]' '[:lower:]')

    done

    echo -e

    if [[ $listname2 == "both" ]]; then

    echo "Retrieving default user and password list..."

    sleep 5

    echo -e

    wget -q -O $dstdir/user.lst https://raw.githubusercontent.com/danielmiessler/SecLists/master/Usernames/top-usernames-shortlist.txt
    wget -q -O $dstdir/pass.lst https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10-million-password-list-top-100.txt

    userfile=$dstdir/user.lst
    passfile=$dstdir/pass.lst

    echo "User and Password list saved in $dstdir"
```

Mentioned on the previous diagram

If user input is “both”, this IF statement is fulfill and will be carried out

WGET command to download the default lists into the destination file previously input by user

Newly downloaded files assigned new variables

Continuation from the script, if the user input is “both”, the script will proceed to download the default user and password list from the link provided in the script. The flag -q turn off verbose for the wget process so the terminal is neat and tidy.

The newly download files will then be assigned their variables, $userfile and $passfile and be use for the bruteforcing process later.

```
Please specify if you would like to use the provided default list of USER, PASS or BOTH...
Choosing to use only one of the default list will required user to enter the needed credentials for the other required field...
Your choice : both

Retrieving default user and password list...

User and Password list saved in /home/kali/Desktop/Results
```

Terminal 2.1.1a

When the option “both” is chosen, the script will download the default list of user and password for the bruteforcing process later

2.1.2 Allow the user to supply their own password list.

```
elif [[ $listname == "pass" ]]; then

    echo "Please provide the relative path of the PASS file you will be using..."
    read passfile

    echo -e
    echo "Retrieving default user list..."
    echo-e

    sleep 5

    wget -q -O $dstdir/user.lst https://raw.githubusercontent.com/danielmiessler/SecLists/master/Usernames/top-usernames-shor

    userfile=$dstdir/user.lst

    echo "Password list saved in $dstdir"

if [[ $listname == "user" ]]; then

    echo "Please provide the relative path of the USER file you will be using..."
    read userfile
    echo -e

    echo "Retrieving default password list..."
    echo -e

    sleep 5

    wget -q -O $dstdir/pass.lst https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Common-Credentials/10-milli

    passfile=$dstdir/pass.lst

    echo "Password list saved in $dstdir"
```

If input “pass” this part will run due to ELIF statement

Read user input on password list path

Download default user list using wget. -q turn the download into non verbose mode

Dedicate the newly download list its variable

If input “user” this part will run due to ELIF statement

Read user input on user list path

Download default pass list using wget. -q turn the download into non verbose mode

Dedicate the newly download list its variable

```
else

    echo "Please provide the relative path of the USER file you will be using..."
    read userfile
    echo -e
    echo "Please provide the relative path of the PASS file you will be using..."
    read passfile

fi
```

User will indicate both the user and password list relative path to upload to the script

Continuation from the previous part of the script. If the user input is “user” for $listname, it will then get the user to upload the user file and download the default password file using wget.

Vice versa for if the user input is “pass”.

If the user input is “both”, the script will then prompt the user to upload both the user and password file.

```
Please specify if you will upload the USER, PASS or BOTH lists to be use for the credential checks...
Default files will be provided if only one options is chosen...
Your choice : pass

Please provide the relative path of the PASS file you will be using...
/home/kali/Desktop/Results/pass.lst

Retrieving default user list...
Password list saved in Results
```

Terminal 2.1.2a

In this example, the user input that they want to upload their password file and provided the path for it. The default user file was then automatically downloaded by the script.

```
Please specify if you will upload the USER, PASS or BOTH lists to be use for the credential checks...
Default files will be provided if only one options is chosen...
Your choice : user

Please provide the relative path of the USER file you will be using...
/home/kali/Desktop/Results/user.lst

Retrieving default password list...

Password list saved in Results
```

Terminal 2.1.2b

Vice versa, if the user input user, they will provide the relative path for the user file and the default password list will be provided.

2.2 Login services to check include: SSH, RDP, FTP, and TELNET.

```
echo "Checking for Weak Credentials..."
echo -e
echo "Bruteforcing on SSH... This may take awhile..."
hydra -L $userfile -P $passfile $ip ssh -t 4 -u 2>/dev/null >> $dstdir/Bruteforce/SSHresults.lst
cat $dstdir/Bruteforce/SSHresults.lst >> $dstdir/Scans/All_Results.lst
echo "Completed..."
sleep 5
echo -e
echo "Bruteforcing on RDP... This may take awhile..."
hydra -L $userfile -P $passfile $ip rdp -t 4 -u 2>/dev/null >> $dstdir/Bruteforce/RDPresults.lst
cat $dstdir/Bruteforce/RDPresults.lst >> $dstdir/Scans/All_Results.lst
echo "Completed..."
sleep 5
echo -e
echo "Bruteforcing on FTP... This may take awhile..."
hydra -L $userfile -P $passfile $ip ftp -t 4 -W 10 -u 2>/dev/null >> $dstdir/Bruteforce/FTPresults.lst
cat $dstdir/Bruteforce/FTPresults.lst >> $dstdir/Scans/All_Results.lst
echo "Completed..."
sleep 5
echo -e
echo "Bruteforcing on Telnet... This may take awhile..."
nmap -p 23 --script telnet-brute --script-args userdb=$userfile,passdb=$passfile,telnet-brute.timeout=8s $ip >> $dstdir/Bruteforce/TELNETresults.lst
cat $dstdir/Bruteforce/TELNETresults.lst >> $dstdir/Scans/All_Results.lst
echo "Completed..."
sleep 5
echo -e
echo "Results for weak credentials are saved in folder..."
```

Non verbose process

SSH bruteforce, hydra with 4 task each time so client does not try to disconnect us. -L for user file and -P for password file

RDP bruteforce, hydra with 4 task each time so client does not try to disconnect us. -L for user file and -P for password file

FTP bruteforce, hydra with 4 task each time and wait time of 10 sec so client does not try to disconnect us. -L for user file and -P for password file

Telnet bruteforce, NSE script using nmap with user and password file indicated by userdb and passdb

This is the bruteforcing portion of the script. It will mainly bruteforce 4 main services, namely SSH, RDP, FTP and TELNET.

Tools used are hydra and NSE scripts.

In this part, we will just execute hydra accordingly using the user and password file previously provided. While we send all the process output to 2>/dev/null so that it does not show in the terminal and make the look neater.

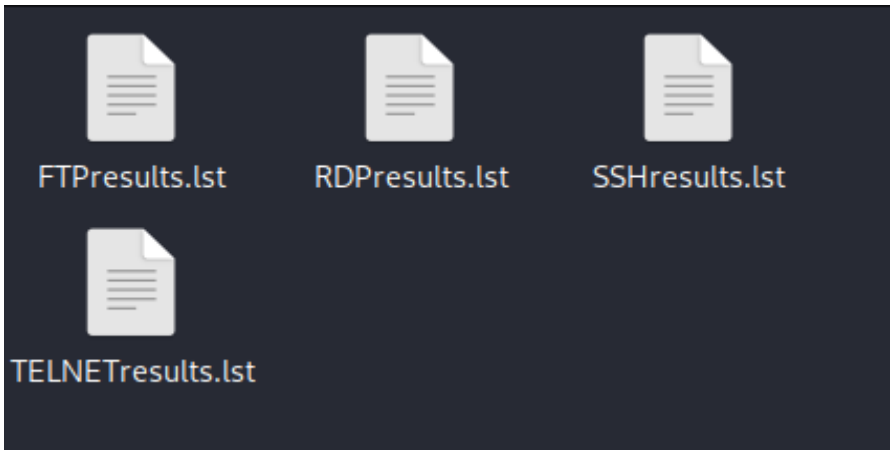
All the output results of the bruteforcing will then be append to the All_Results.lst in addition to each of their own files.

For TELNET, as hydra is rather unreliable regarding it. We used a NSE script to do the bruteforcing instead.

```
Checking for Weak Credentials...

Bruteforcing on SSH... This may take awhile...
Completed...

Bruteforcing on RDP... This may take awhile...
Completed...

Bruteforcing on FTP... This may take awhile...
Completed...

Bruteforcing on Telnet... This may take awhile...
Completed...

Results for weak credentials are saved in folder...
```

Terminal 2.2a

Bruteforcing process on terminal. The actual process is running on non verbose, so that it will not flood the terminal.



Terminal 2.2b

All bruteforcing results are saved in the folder user specify previously.

3.1 Mapping vulnerabilities should only take place if Full was chosen.

```
else
    # Run FULL vulnerability scan command here

    mkdir -p $dstdir/Scans
    echo "Running Full Vulnerability Scan..."
    echo -e
    echo "Running Nmap Scanning Tool..."
    nmap -sV $ip -p- >> $dstdir/Scans/TCPresults.lst
    cat $dstdir/Scans/TCPresults.lst >> $dstdir/Scans/All_Results.lst
    echo -e
    echo "Running Vulscan on open ports..."
    nmap -sV --script=vulscan/vulscan.nse $ip >> $dstdir/Scans/Vulscanresults.lst
    cat $dstdir/Scans/Vulscanresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Nmap Scan completed..."
    echo -e
    echo "Running Masscan... Permission might be required"
    sudo masscan $ip -pU -p- --rate 500 >> $dstdir/Scans/UDPresults.lst
    cat $dstdir/Scans/UDPresults.lst >> $dstdir/Scans/All_Results.lst
    echo "Masscan completed..."
    echo -e
```

Vulnerability scan will only take place in full scan mode when user input “full” on the previous while loop

3.2 Display potential vulnerabilities via NSE and Searchsploit.

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-02 10:54 EST
Nmap scan report for msf (192.168.155.131)
Host is up (0.00083s latency).
Not shown: 978 closed tcp ports (conn-refused)
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
| vulscan: VulDB - https://vuldb.com:
| No findings
|
| MITRE CVE - https://cve.mitre.org:
| No findings
|
| SecurityFocus - https://www.securityfocus.com/bid/:
| No findings
|
| IBM X-Force - https://exchange.xforce.ibmcloud.com:
| No findings
|
| Exploit-DB - https://www.exploit-db.com:
| No findings
|
| OpenVAS (Nessus) - http://www.openvas.org:
| No findings
|
```

The vulnerability scan will only be executed if the user selected “FULL” scan. The output of the vulnerability scan will be stored with the Vulscan results file and the All_Results.lst file.

4.3 Allow the user to search inside the results.

```
echo "Do you wish to do a search using the terminal? *Yes/No"
echo "Both options will open the final result file for viewing..."

while [[ $listname3 != "yes" && $listname3 != "no" ]]; do

        echo -n "Your choice : "; read listname3

        listname3=$(echo "$listname3" | tr '[:upper:]' '[:lower:]')

        done

echo -e

    if [[ $listname3 == "yes" ]]; then

        open $dstdir/Scans/All_Results.lst

        while true; do

        # Ask the user for input

        read -p "Enter a pattern to grep: " pattern
        echo -e

        # Use grep to find all lines that match the pattern

        grep -E "$pattern" $dstdir/Scans/All_Results.lst
        echo -e

        # Check if the user wants to quit

        read -p "Press Q to quit or any other key to continue: " choice
        echo -e

        if [[ "$choice" == "Q" ]] || [[ "$choice" == "q" ]]; then

            break

        fi
```

Opening the All_Results file for the user to view

Get user input on the word to filter

Using the user input, the script execute the grep command to filter out results

Escape sequence, so that user can exit the search query

In this section of code, the script will prompt the user if they want to do a search query on the results the script saved so far using a while loop.

If “yes”, the function grep will filter results in the All_Results.lst file that saved all results that was done in the scans and return it to the user. The user can keep searching for related results and exit by entering Q.

If “no” is input, the script will simply open up the All_Results.lst file to show the user all the results gathered so far.

```
Do you wish to do a search using the terminal? *Yes/No
Both options will open the final result file for viewing...
Your choice : yes
Enter a pattern to grep: telnet

23/tcp    open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp open  telnet
| telnet-brute:
23/tcp    open  telnet      Linux telnetd
23/tcp    open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp open  telnet
| telnet-brute:
23/tcp open  telnet
| telnet-brute:
23/tcp    open  telnet      Linux telnetd
23/tcp    open  telnet      Linux telnetd
23/tcp    open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp   open  telnet      Linux telnetd
23/tcp open  telnet
| telnet-brute:
23/tcp open  telnet
| telnet-brute:
23/tcp open  telnet
| telnet-brute:

Press Q to quit or any other key to continue:
```

Terminal 4.3a

User get prompt on whether to do search. Search query will then prompt for the word to filter.

In this example, we entered telnet, and the script returns all related results to us.

4.4 Allow to save all results into a Zip file.

```
    else

    open $dstdir/Scans/All_Results.lst

    fi

zip -r $dstdir $dstdir

echo -e
echo "All results zipped into zip folder..."

echo "End of Vulnerability scans..."
```

In this section of code, if the script will simply save all results, folders and file that was specify by the user in $dstdir, and zip to it.

```
  adding: Results/ (stored 0%)
  adding: Results/pass.lst (deflated 43%)
  adding: Results/user.lst (deflated 24%)
  adding: Results/Bruteforce/ (stored 0%)
  adding: Results/Bruteforce/SSHresults.lst (deflated 74%)
  adding: Results/Bruteforce/FTPresults.lst (deflated 81%)
  adding: Results/Bruteforce/RDPresults.lst (deflated 77%)
  adding: Results/Bruteforce/TELNETresults.lst (deflated 70%)
  adding: Results/Scans/ (stored 0%)
  adding: Results/Scans/Vulscanresults.lst (deflated 95%)
  adding: Results/Scans/UDPresults.lst (deflated 95%)
  adding: Results/Scans/All_Results.lst (deflated 94%)
  adding: Results/Scans/TCPresults.lst (deflated 82%)

All results zipped into zip folder...
End of Vulnerability scans...
```

Terminal 4.4a

The terminal output showing the files being zip into the zip folder

END OF REPORT