Penetration Testing Project: VULNER

Penetration Testing

STUDENT: MUHAMMAD FEROZ (S10)

Content Page

Overview

This script performs security checks and penetration testing in a given network. It determines the number of hosts and checks services for vulnerabilities that are not secured. The script goes further to check on possible exploits that could be used against these services. Outputs are generated for reference and analysis for gaining access and post-exploits stages. The script is monitored by its respective log on its usage.

Shell selection

For controlling system resources, automating system administration operations and carrying out other standard tasks in Unix/Linux systems, bash scripting is an adaptable tool.

Automation and Flexibility: By using bash shell scripts, it saves time and lowers the possibility of errors that might arise from the manual execution of repetitive processes that are coded in other shell languages. It is very adaptable and simple to adjust to meet particular requirements and future enhancements.

Portability and Accessibility: Bash shell scripts can be executed on several platforms and operating systems, including Windows, Linux, macOS, and Unix. Bash shell scripts don't require any specialized software or tools to write, and they are simple to write. Any text editor can be used to edit them, and the majority of operating systems come with a shell interpreter already installed.

Scope

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.

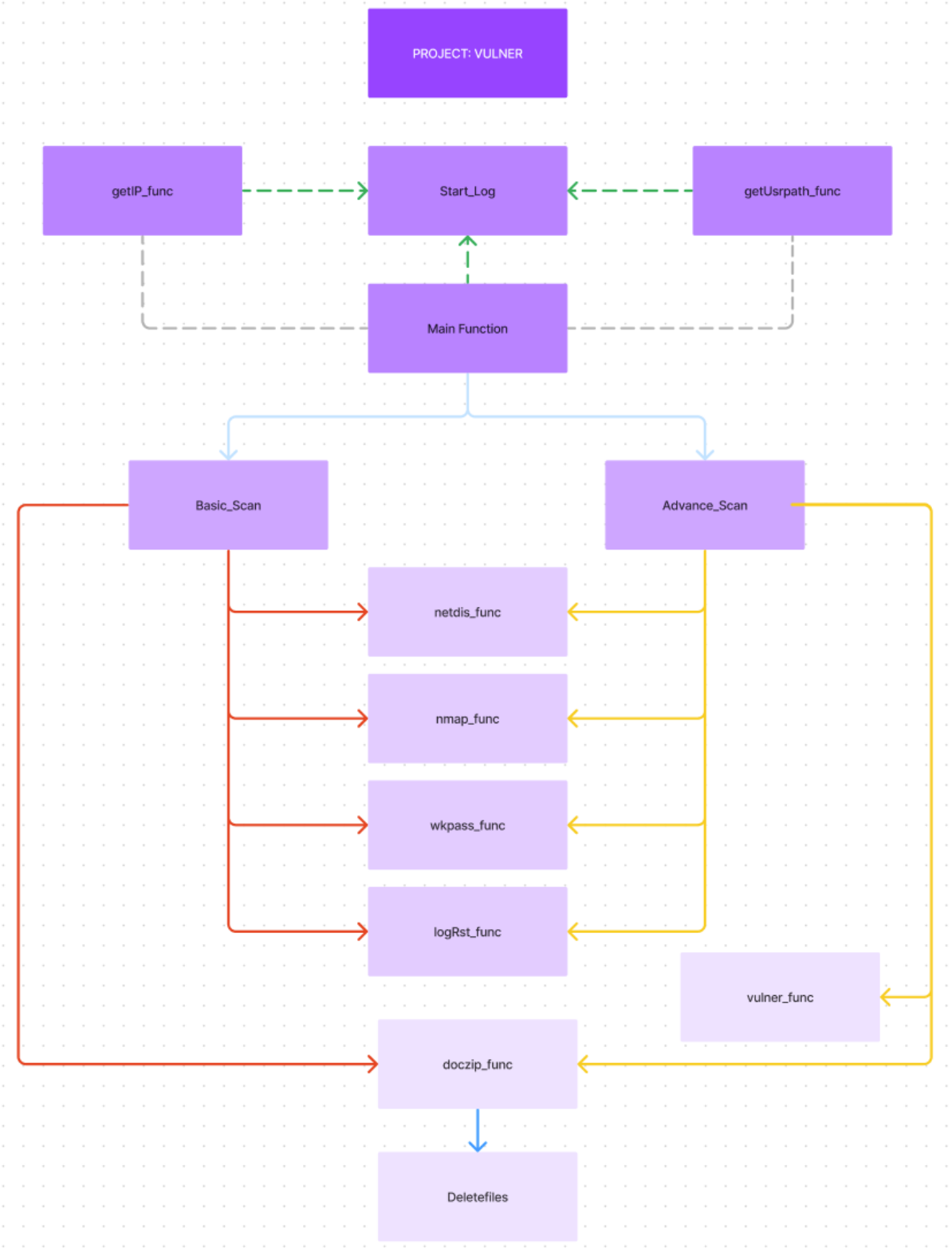
Methodology

The script has been grouped into multiple functions to ease the management of the codes and allow the reusability of respective code blocks that simplify scripts, organize code, and carry out the intended tasks.

Functions

Functions	Intended Purpose
Main	Calls and controls the sequence of events that would have to be taken upon the user’s input. It determines the start and completion of the entire automation process.
getIP_func	Request the user to provide a valid Network address.
getUsrpath_func	Request the user to provide a valid output directory path.
netdis_func	Determines the number of active hosts within the given network. Temporary input files are created to quicken the execution of commands.
nmap_func	Quickly map the whole network and locate any open ports or services for security auditing and network investigation. Temporary input files are created to quicken the execution of commands.
vulner_func	Determines the CVE (Common Vulnerabilities and Exposures) information for a particular service and provides a list of possible exploits related to scanned ports/services which are vulnerable.
wkpass_func	Brute force passwords for SSH, RDP, FTP, and TELNET services.
logRst_func	Print out the results of all previously executed functions and provide an opportunity to perform a search with it.

doczip_func	Allows flexibility for the output of the search results to be zipped together.
Deletefiles	Delete Temporary input files before routing them back to the Main function.



Refer to the Functional block diagram for a detailed overview.

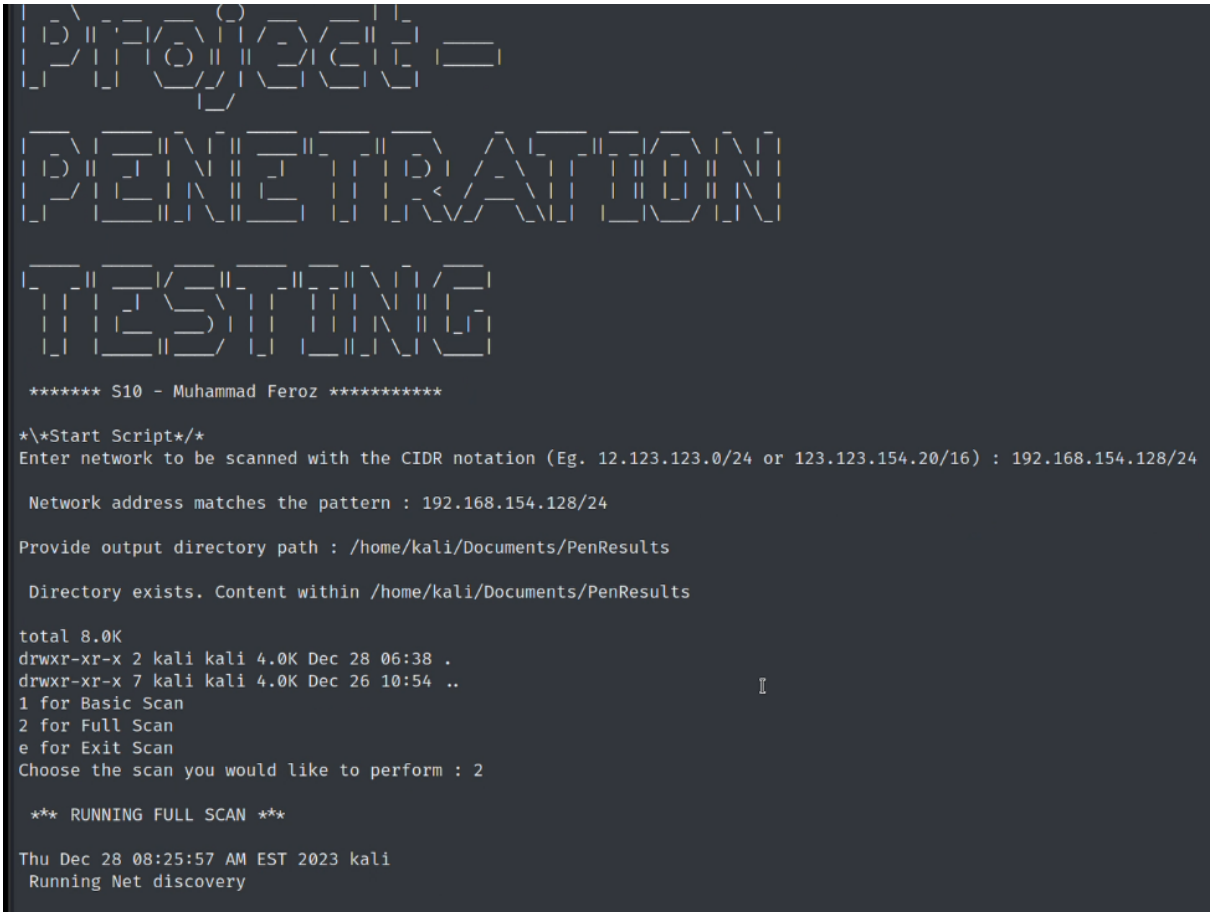
Main function

The case function behaves as a central point to orchestrate the events in the Main function. This allows the script to be easily expanded with more functionality without affecting existing automated processes.

The script starts with the creation of a log file and is followed through by calling a list of functions in a sequential order. Each function is executed together with the **tee** command to ride the logs of all standard output of commands being executed within the function. Before each function gets executed the **date** and **hostname** are logged for time stamp purpose.

Sequence	Description of Sequence
1	A new log file was created to monitor the script, date and hostname captured for tracking.
2	Banner of the script and author
3	Calls for getIP_func and outputs from the function are captured in the log.
4	Calls for getUsrpath_func and outputs from the function are captured in the log.
5	**Case** function used to determine the Basic scan, Advance scan, exit of script and validation of user input. This function is encapsulated with a **while loop** to prevent exit of script until user requires of it.
5a	Basic scan calls for netdis_func, nmap_func, wkpass_func, logRst_func and doczip_func. Outputs from each function are captured in the log.
5b	Advance scan calls for netdis_func, nmap_func, wkpass_func, vulner_func, logRst_func and doczip_func. Outputs from each function are captured in the log.
5c	The user has to ‘e’ to exit the script.
5d	Invalid characters are captured by a wild card, which forces users to re-enter a choice due to the continue command which triggers the **while** function to be re-excuted.

> Commands used: tee, figlet, sleep, while loop, case, exit, continue.




```
### start script log ###
touch ./Penl.log # create log file
(date; hostname) |tr '\n' '\t'|tee -a ./Penl.log #timestamp to log file

#Banner introduction
figlet -mini Project - PENETRATION TESTING
echo -e " ******* S10 - Muhammad Feroz (PRJ: VULNER)********** \n"
echo "*\*Start Script*/*"

### Main Function ###
getIP_func > >(tee -ap ./Penl.log) #get network address and screen output to log file
getUsrpath_func > >(tee -ap ./Penl.log) #get output path and screen output to log file
while true
do
	echo "1 for Basic Scan"
	echo "2 for Full Scan"
	echo "e for Exit Scan"
	read -p 'Choose the scan you would like to perform : ' usr_Option #get user option on scan output
	case $usr_Option in
		1)
			echo -e '\n *** RUNNING BASIC SCAN *** \n'
			(date; hostname) |tr '\n' '\t'|tee -a ./Penl.log
			### Functions ###
			sleep 5s
			netdis_func > >(tee -ap ./Penl.log) #run netdiscovery and screen output to log file
```

```
                nmap_func   > >(tee -ap ./Penl.log)                #run nmap and screen output to log file
                wkpass_func > >(tee -ap ./Penl.log)                #run brute force tools and screen output to log
file
                logRst_func > >(tee -ap ./Penl.log)            #get outputs to be displayed and screen output to
log file
                doczip_func > >(tee -ap ./Penl.log)            #run zip options for file and screen output to log file
                echo -e '\n *** BASIC SCAN COMPLETED *** \n'
            ;;
            2)
                echo -e '\n *** RUNNING FULL SCAN *** \n'
                (date; hostname) |tr '\n' '\t'|tee -a ./Penl.log
                ### Functions ###
                sleep 5s
                netdis_func > >(tee -ap ./Penl.log)
                nmap_func   > >(tee -ap ./Penl.log)
                wkpass_func > >(tee -ap ./Penl.log)
                vulner_func > >(tee -ap ./Penl.log)            #run vulerbility testing and exploit suggestion and
screen output to log file
                logRst_func > >(tee -ap ./Penl.log)
                doczip_func > >(tee -ap ./Penl.log)
                echo -e '\n *** FULL SCAN COMPLETED *** \n'
            ;;
            e)
                echo "Exit script"
                exit                                           #exit script
            ;;
            *)
                echo "Invalid option"
                continue                                       #re-run the loop due to invalid option
            ;;
        esac
done
echo "*\*End Script*/*"
```

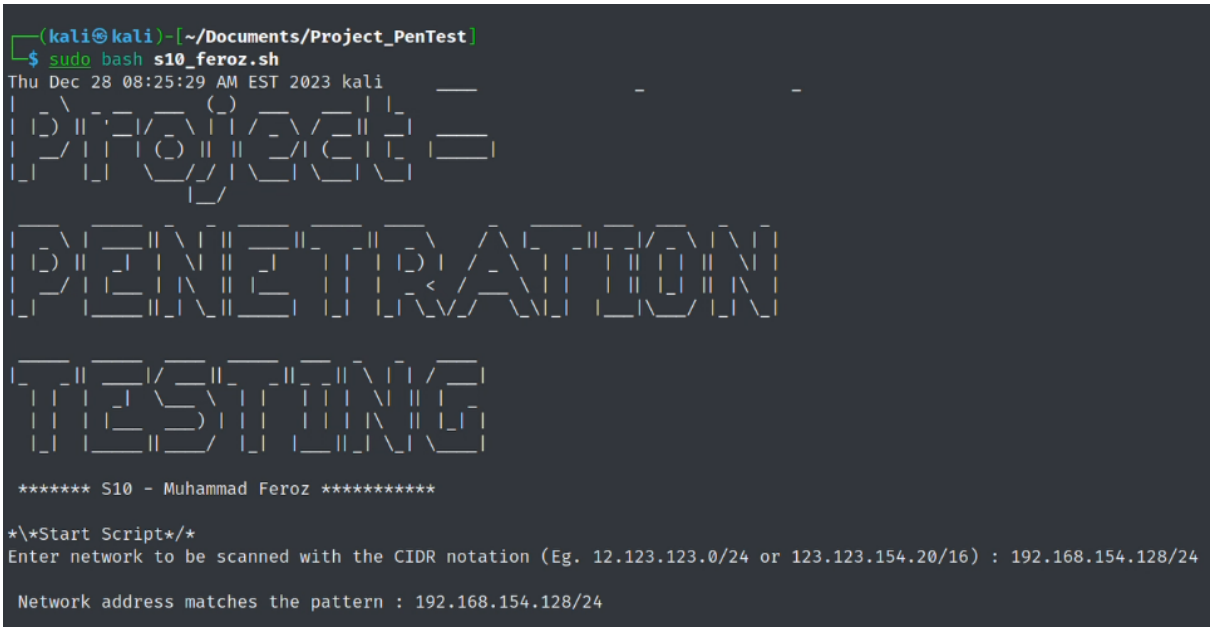
---

getIP_func Function

This function manages the receiving and validating of the Network address. The network address is stored in as a variable that will be used in the next function.

Sequence	Description of Sequence
1	Reads the user's input for a Network address
2	Validates the input against a **rgex** (regular expression) pattern, where there needs to be a total of four blocks of numbers and each block has a maximum of 3 digits. Each digit can range from numerals 1 to 9. End of IP address bus be accompanied by a '/' with a maximum of 2 digits to denote the CIDR notation.
3	Validation is managed by the **if-else** function, which re-loads the getIP_func function if pattern validation fails.

> Commands used: if-else, rgex, global variable.




```
function getIP_func
{
read -p 'Enter network to be scanned with the CIDR notation (Eg. 12.123.123.0/24 or 123.123.154.20/16) : '
ip_add
pattern="[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\/[0-9]{1,2}" #rgex pattern for IP address
#Validation of user input
if [[ $ip_add =~ $pattern ]]; then
  echo -e "\n Network address matches the pattern : $ip_add \n"
  sleep 3s
else
  echo -e "\n Network address does not match the pattern \n"
   getIP                                                                    # repeat function due to invalid
IP address
fi
}
```

---

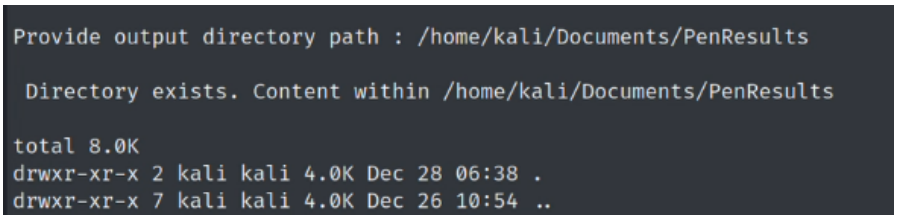
getUsrpath_func Function

This function manages the receiving and validating of a directory path. This path is stored as a variable that will be used in the next function.

Sequence	Description of Sequence
1	Reads the user's input for an Output directory path, this is managed within the while loop.
2	Validates the input using the test command '**-d**' to ensure input provided by the user is a valid directory. **While** loop will continue the function until a valid directory is provided.

3	List the contents of the output path after a successful validation

> Commands used: test, ls, while loop



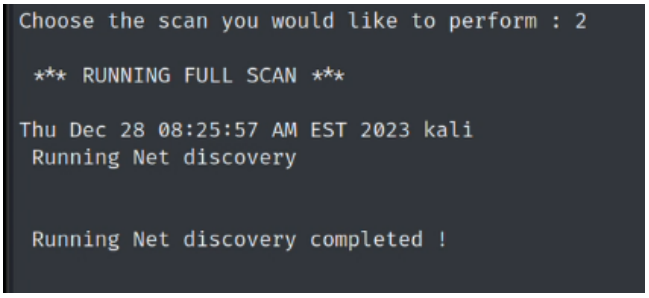

```
function getUsrpath_func
{
#Validation of user input
while true
read -p 'Provide output directory path : ' usr_path
do
	if [ -d $usr_path ] 					# check if input is a directory
	then
		echo -e "\n Directory exists. Content within $usr_path \n"
		ls -lah $usr_path 				# print contents within path
		break
	else
		echo -e "\n Directory not exists. \n" 		# repeat function due to invalid dir
		continue
	fi
done
}
```

netdis_func Function

This function manages and discovers the number of active hosts within the provided network. An analyst can benefit greatly from the system information that is collected form this function, such as MAC address, hostname, and manufacturer of devices connected to a network. From the results, a snippet of active IP address is extracted as a temporary file, that will be fed commands in other functions. This will help to speed up the execution of codes with a defined set of active IPs.

Sequence	Description of Sequence
1	Runs **netdiscover** with a defined range (**-r**) provided by user and it will stop execution (**-p**) after scanning the given ranges.
2	Results of the scanned are logged to in a file, **netdis_results.txt**.
3	From the network scanned results, only the IP address are extracted with a **rgex** (regular expression) pattern, where there needs to be a total of four blocks of numbers and each block has a maximum of 3 digits. Each digit can range from numerals 1 to 9. [prevent each tool from re-discovering the active host in the given network]

> Commands used: netdiscover, grep, regex




```
function netdis_func
{
echo -e '\n Running Net discovery \n'
netdis_results=$(sudo netdiscover -r $ip_add -P)                    # run netdiscovery for given network
echo "$netdis_results" > netdis_results.txt                              # log results to a file
echo "$netdis_results"|grep -oP '([0-9]{1,3}\.){3}[0-9]{1,3}' > IP_only.txt # extract only IP address base on the pattern
echo -e '\n Running Net discovery completed ! \n'
}
```

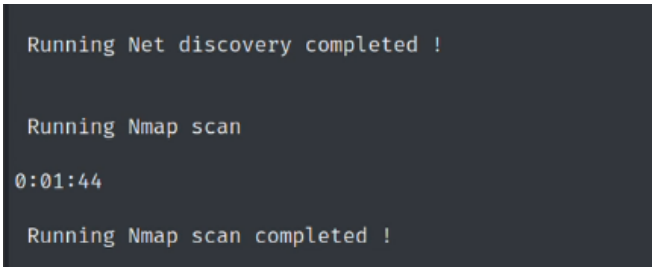
---

nmap_func Function

This function is a great benefit to an analyst as the function finds out which devices are connected to the network, identifies open ports/services and identifies security holes. The function checks for both TCP & UDP (common ports) and their services. Two outputs are generated during this event, first is the results of **namp** scan and the second is an XML out which will be fed to **searchsploit** to find possible vulnerabilities.

Sequence	Description of Sequence
1	Runs **nmap** with a defined range of IPs taken from the temp file. Search for UDP ports (-sU), TCP ports (-sS) and Version detection (-sV) for command 100 ports (-F) instead of usual 1000 ports. [**pv** tool acts has progress timer while commands are excluded, as it monitors data flow]
2	Results of the scan are logged to in a file, nmap_Vulresults.txt.
3	Runs **nmap** with a defined range of IPs taken from the temp file. Search for Version detection (-sV) for command 100 ports (-F). [Temp file created for **searchsploit**] [**pv** tool acts as progress timer while commands are excluded, as it monitor data flow] [error output is suppressed with **2> /dev/null**.]
4	Results of the scanned are logged to in a file, **nmap_resultsXML.xml**.

> Commands used: nmap, grep, regex, pv




```
function nmap_func
{
echo -e '\n Running Nmap scan \n'
#nmap_results=$(sudo masscan -p1-65535,U:1-65535 -iL ./IP_only.txt --banners --rate=1000|pv -t)                #
Takes long time to complete
nmap_results=$(sudo nmap -sU -sS -sV -F -iL ./IP_only.txt --open -T4 --version-intensity 0|pv -t)              # get
TCP/UDP port/services details
echo "$nmap_results" > nmap_results.txt                                                                        # log results to a file
sudo nmap -sV -F -iL ./IP_only.txt -oX nmap_resultsXML.xml >/dev/null # get XML for NMAP Vul analysis
echo -e '\n Running Nmap scan completed ! \n'
}
```

> 📌 Nmap Command had to optimized to achieve quick turnaround, the following command took longer time to complete for 2 active host.
> sudo nmap -sU -Ss -sV -F -iL ./IP_only.txt (Total time taken 28 minutes)
> masscan -p1-65535,U:1-65535 -iL ./IP_only.txt --banners --rate=1000 (Total time taken 15 minutes)

---

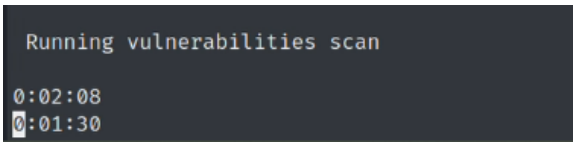
vulner_func Function

This function provides an in-depth analysis of the possible vulnerabilities of the active hosts in the network. Function leverages on **nmap** tool to map vulnerabilities with known CVEs that can be used to exploit these services. In addition to this **nmap** is complimented with **searchsploit** tool to provide related exploits against these vulnerable services. This gives the analysis a nearly complete picture before heading to the “gaining access and post exploit’ phase.

Sequence	Description of Sequence
1	Runs **nmap** with a defined range of IPs taken from the temp file. Search for UDP ports (-sU), TCP ports (-sS), Version detection (-sV) for command 100 ports (-F) with nse script called vulners. [**pv** tool acts as progress timer while commands are excluded, as it monitor data flow]

	[error output is suppressed with **2> /dev/null**.]
2	Results of the scan are logged to in a file, **nmap_Vulresults.txt**.
3	Runs **searchsploit** with **nmap** service analysis in XML format. [**pv** tool acts as progress timer while commands are excluded, as it monitor data flow] [error output is suppressed with **2> /dev/null**.]
4	Results of the scan are logged to in a file, **ssp_results.txt**.

> Commands used: nmap, searchsploit, pv, test




```
function vulner_func
{
echo -e '\n Running vulnerabilities scan \n'
nmap_Vulresults=$(sudo nmap -sU -sS -sV -F -iL ./IP_only.txt --open -T4 --version-intensity 0 --
script=vulners.nse 2> /dev/null |pv -t) # get CEV details
ssp_results=$(searchsploit --nmap ./nmap_resultsXML.xml 2> /dev/null |pv -t) # get posible expolit details
echo "$nmap_Vulresults" > nmap_Vulresults.txt                    # log results to a file
echo "$ssp_results" > ssp_results.txt                            # log results to a file
echo -e '\n Running vulnerabilities scan completed ! \n'
}
```

---

wkpass_func Function

This function performs a brute force password cracking on 4 selected ports: SSH, RDP, FTP and TELNET. With the use of this function, analysis may demonstrate how simple it would be to obtain unauthorized remote access to a system with the given network. User is allowed to use a pre-defined password list or able to replace it with a new list of their choice.

Sequence	Description of Sequence
1	Reads the user's input for a password list choice.
2	The **case** function manages the decision process, if the option is selected as 'n' a full path with the file name needs to be provided. User input is validated with if-else function with the test (-f) command to ensure a file is provided. After successful validation file properties (RWX) is displayed. if the option is selected as 'y' existing password list will be used. Invalid entries are captured by wildcard and made to re-execute the function.

3	Run **hydra** tool for SSH port with -t 6 option to speed up the brute force. [**pv** tool acts as progress timer while commands are excluded, as it monitors data flow] [error output is suppressed with **2> /dev/null**.]
4	Results of the scan are logged in a file, **weakp_ssh.txt**.
5	Run **hydra** tool for RDP port with -t 6 option to speed up the brute force. [**pv** tool acts as progress timer while commands are excluded, as it monitors data flow] [error output is suppressed with **2> /dev/null**.]
6	Results of the scan are logged in a file, **weakp_rdp.txt**.
7	Run **hydra** tool for FTP port with -t 6 option to speed up the brute force. [**pv** tool acts as progress timer while commands are excluded, as it monitors data flow] [error output is suppressed with **2> /dev/null**.]
8	Results of the scan are logged in a file, **weakp_ftp.txt**.
9	Run **medusa** tool for TELNET port . [**pv** tool acts as progress timer while commands are excluded, as it monitors data flow] [error output is suppressed with **2> /dev/null**.]
10	Results of the scan are logged in a file, **weakp_telnet.tx**t.

> Commands used: hydra, medusa, pv, test

```
Would like to to use the default password list [y/n] : y
Running weak password scan /n
Existing password list : ./wordlst/top-passwords-shortlist.txt
Testing weak password for ssh servicer (Kindly be patient)
0:01:08
Testing weak password for rdp servicer (Kindly be patient)
0:01:31
Testing weak password for ftp servicer (Kindly be patient)
0:08:13
Testing weak password for telnet servicer (Kindly be patient)
0:00:43
```

```
function wkpass_func
{
read -p 'Would like to to use the default password list [y/n] : ' choice_wk
echo 'Running weak password scan /n'
# Descesion making proceess to supply new lst
case $choice_wk in
        n)
                while true
                read -p 'Please provide full path and file name : ' usr_file
                do
                        if [ -f $usr_file ]                                     # check path provided inpu is a file
                        then
                                echo -e "\n File exists : $usr_file \n"
```

```
                ls -lah $usr_file                                   # print file properties
                break
            else
                echo -e "\n File not exists. \n"
                continue
            fi
        done
    ;;
    y)

        passlst='./wordlst/top-passwords-shortlist.txt'          # default password list
        echo "Existing password list : $passlst"
    ;;
    *)
        echo "Invalid option"
        wkpass_func                                                 # repeat case for an invalid
option
    ;;
esac

# ssh brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for ssh servicer (Kindly be patient)'
weakp_ssh=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt ssh -t6 2> /dev/null |pv -
t)
echo "$weakp_ssh" > weakp_ssh.txt

# rdp brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for rdp servicer (Kindly be patient)'
weakp_rdp=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt rdp -t6 2> /dev/null |pv -
t)
echo "$weakp_rdp" > weakp_rdp.txt

# ftp brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for ftp servicer (Kindly be patient)'
weakp_ftp=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt ftp -t6 2> /dev/null |pv -t)
echo "$weakp_ftp" > weakp_ftp.txt

# telnet brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for telnet servicer (Kindly be patient)'
#weakp_telnet=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt telnet -t12 2>
/dev/null |pv -t) # takes longer time for excution
weakp_telnet=$(medusa -U ./wordlst/top-usernames-shortlist.txt -P $passlst -H ./IP_only.txt -M telnet 2>
/dev/null |pv -t)
echo "$weakp_telnet" > weakp_telnet.txt

echo 'Running weak password scan comeplted ! /n'
}
```

📌 Quicker completion time was achieved when medusa was used for telnet brute force, nmap took a total of 25 minutes for 2 active hosts.

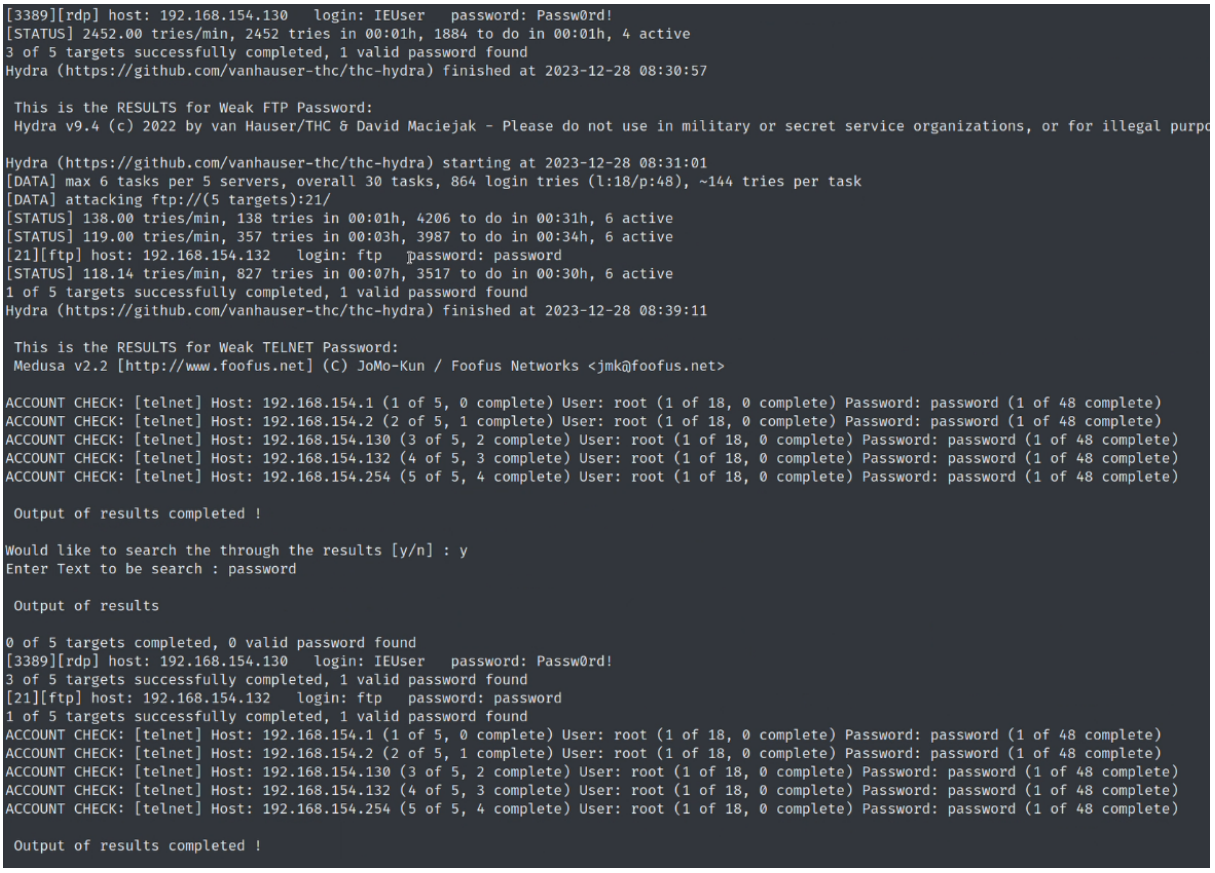
---

logRst_func Function

The purpose of this function is to manage the display of the results to the terminal, with the ability to search keywords.

Sequence	Description of Sequence
1	Displays results of each function onto the terminal 1 - Display Netdiscovery results 2 - Display Namp results - ports and services 3 - Display Nmap Vulnerability results 4 - Display Searchsploit results 5 - Display Weak Password results (for all 4 ports) [output of the results is taken from variables instead of a file/log, taking advantage of the default global variable provided by BASH]
2	Reads the user's input to perform a search through the results.
3	The **case** function manages the decision process. If the option is selected as 'n' it breaks the **while loop** else if 'y' is selected, the user will be allowed to perform a search based on the information printed on the terminal. the **grep** command captures the search criteria applied against each variable for meaning return. Invalid entries are captured by wildcard and made to re-execute the function.

> Commands used: hydra, medusa, pv, test




```
function logRst_func
{
# Output of results are displayed at the end
echo -e '\n Output of results \n'
echo -e "\n This is the RESULTS for Netdiscovery : \n $netdis_results"
echo -e "\n This is the RESULTS for Nmap Ports/Services : \n $nmap_results"
echo -e "\n This is the RESULTS for Nmap Vulnerability (CVES): \n $nmap_Vulresults"
echo -e "\n This is the RESULTS for Searchsploit (possible exploit) : \n $ssp_results"
echo -e "\n This is the RESULTS for Weak SSH Password: \n $weakp_ssh"
echo -e "\n This is the RESULTS for Weak RDP Password: \n $weakp_rdp"
echo -e "\n This is the RESULTS for Weak FTP Password: \n $weakp_ftp"
echo -e "\n This is the RESULTS for Weak TELNET Password: \n $weakp_telnet"
echo -e '\n Output of results completed ! \n'

while true
do
	read -p 'Would like to search the through the results [y/n] : ' choice_lg
# serach of contents within the results with grep
	case $choice_lg in
		n)
		  break
		;;

		y)
		 read -p "Enter Text to be search : " searcht
		 # grep with variable use to filter the results
```

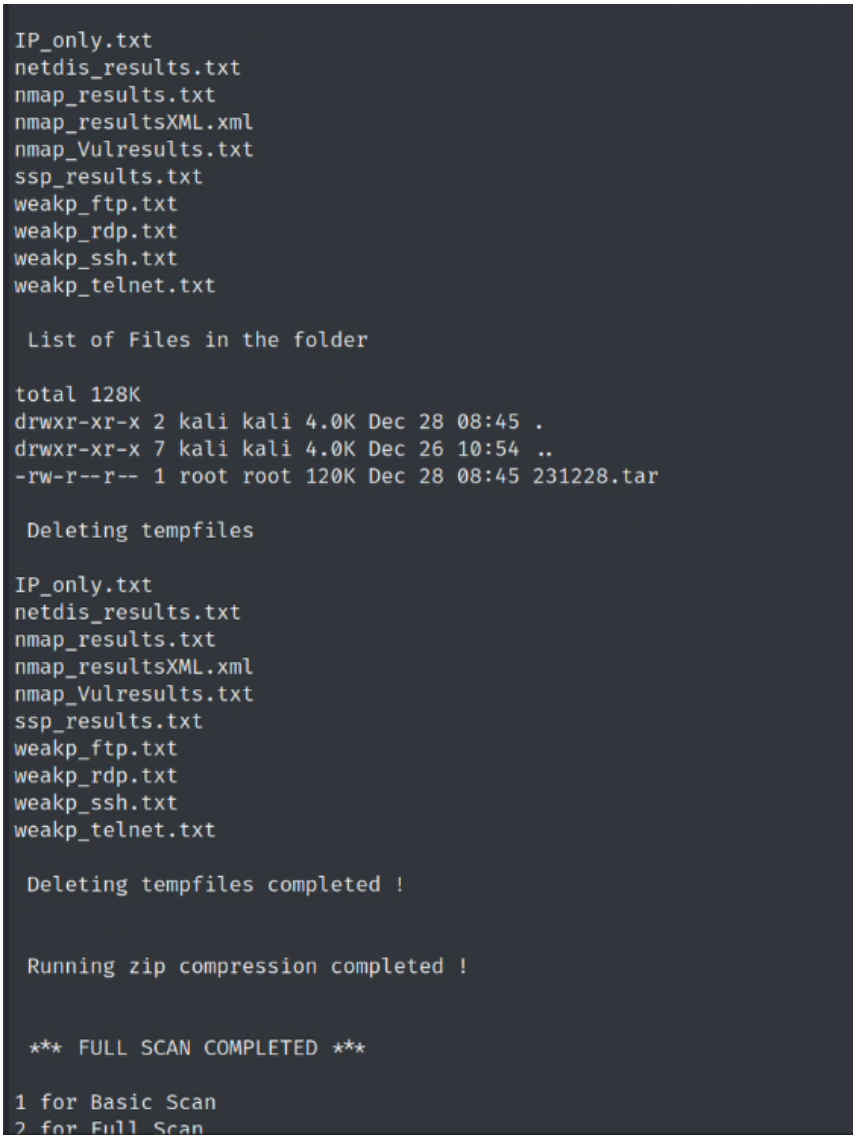
```
            echo -e '\n Output of results \n'
            echo -e "\n This is the RESULTS for Netdiscovery : \n $netdis_results" | grep "$searcht"
            echo -e "\n This is the RESULTS for Nmap Ports/Services : \n $nmap_results" | grep "$searcht"
            echo -e "\n This is the RESULTS for Nmap Vulnerability (CVES): \n $nmap_Vulresults" | grep
"$searcht"
            echo -e "\n This is the RESULTS for Searchsploit (possible exploit) : \n $ssp_results" | grep "$searcht"
            echo -e "\n This is the RESULTS for Weak SSH Password: \n $weakp_ssh" | grep "$searcht"
            echo -e "\n This is the RESULTS for Weak RDP Password: \n $weakp_rdp" | grep "$searcht"
            echo -e "\n This is the RESULTS for Weak FTP Password: \n $weakp_ftp" | grep "$searcht"
            echo -e "\n This is the RESULTS for Weak TELNET Password: \n $weakp_telnet" | grep "$searcht"
            echo -e '\n Output of results completed ! \n'
            continue
            ;;

            *)
            echo "Invalid option"                    # repeat case for an invalid option
            continue
            ;;

    esac
done
}
```

doczip_func Function

The purpose of this function is to allow the compression of multiple outs into a single compressed file and/or move file(s) to the designated output directory required by the user. The function also calls for a sub-function to delete Temp files..

Sequence	Description of Sequence
1	Reads the user's input to perform a compression of files.
2	The **case** function manages the decision process. If the option is selected as 'n', files will not be zip but moved to the appropriate output directory. If option 'y' is selected, files within the current folder will compress into a single **tar** (-uf) file and be moved to the appropriate output directory. The **tar** file name is fixed with the current date (YYMMDD) when the event is executed.
3	Calls for subfunction to delete explicitly named temp files.
4	List contents within the output directory.

> Commands used: rm, ls, tar, test



```
IP_only.txt
netdis_results.txt
nmap_results.txt
nmap_resultsXML.xml
nmap_Vulresults.txt
ssp_results.txt
weakp_ftp.txt
weakp_rdp.txt
weakp_ssh.txt
weakp_telnet.txt

 List of Files in the folder

total 128K
drwxr-xr-x 2 kali kali 4.0K Dec 28 08:45 .
drwxr-xr-x 7 kali kali 4.0K Dec 26 10:54 ..
-rw-r--r-- 1 root root 120K Dec 28 08:45 231228.tar

 Deleting tempfiles

IP_only.txt
netdis_results.txt
nmap_results.txt
nmap_resultsXML.xml
nmap_Vulresults.txt
ssp_results.txt
weakp_ftp.txt
weakp_rdp.txt
weakp_ssh.txt
weakp_telnet.txt

 Deleting tempfiles completed !


 Running zip compression completed !


 *** FULL SCAN COMPLETED ***

1 for Basic Scan
2 for Full Scan
```

```
function doczip_func
{
read -p 'Would like to zip file [y/n] : ' choice_zp
# Descesion making proceess to zip files and/or move files to output dir
case $choice_zp in
    n)
        echo -e '\n Files are not ziped and they are located in : $usr_path \n'
        Deletefiles                                          # delete unrequired tempfiles
        for count in $(ls |grep -e .txt)                     # get list of txt files from dir
        do
            echo "$count"
            mv $count $usr_path                              # move txt files
        done
        echo -e '\n List of Files in the folder \n'
        ls -lah $usr_path
    ;;
    y)
        echo -e '\n Files are ziped and they are located in : $usr_path \n'
        Deletefiles                                          # delete unrequired tempfiles
```

```
            echo -e '\n Running zip compression \n'
            for count in $(ls |grep -e .txt -e .xml)                    # get list of txt and xml files from dir
            do
                  echo "$count"
                  tar -uf $(date +"%y%m%d" ).tar $count                 # zip multiple logs to a single tar file with
current date a the name
            done
            mv ./$(date +"%y%m%d" ).tar $usr_path                       # move tar file to a output dir
            echo -e '\n List of Files in the folder \n'
            ls -lah $usr_path
les
            echo -e '\n Running zip compression completed !\n'
      ;;
      *)
            echo "Invalid option"
            doczip
      ;;
esac
}

function Deletefiles
{
echo -e "\n Deleting tempfiles \n"
rm -f IP_only.txt                                                       #delete temp ip file
rm -f nmap_resultsXML.xml                                               # delete temp files
echo -e "\n Deleting tempfiles completed ! \n"
echo -e "\n Deleting tempfiles completed ! \n"
}
```

Conclusion

To sum up, the automation script is an invaluable resource for security experts to evaluate a network's security posture. It can swiftly identify weaknesses in passwords, vulnerabilities, and potential exploits. Additionally, the script assists organizations in fortifying their defences against potential cyber threats.

Recommendation

The script could benefit from the following improvements:

- Enhancing the reporting functionality as reporting is the final phase in the pen testing process. Good reporting tools assist pen testers in creating clear reports to produce a quality pen test documentation.
- Implementing additional checks against user input such as validation and sanitization
- Integrating with more vulnerability assessment tools (eg Archery, Intrigue Core etc) to expand the scope of vulnerability analysis
- Implementing more advanced password-cracking techniques and strategies.

References

Bash scripting tutorial – linux shell script and command line for beginners. (2023). *freeCodeCamp.org* [online]. Available from: https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/ [accessed 28 December 2023].

Chandel, R. (2018). Comprehensive Guide on SearchSploit. *Hacking Articles* [online], 27 October 2018. Available from: https://www.hackingarticles.in/comprehensive-guide-on-searchsploit/ [accessed 29 December 2023].

tee command in Linux with examples. (2017). *GeeksforGeeks* [online], 5 December 2017. Available from: https://www.geeksforgeeks.org/tee-command-linux-example/ [accessed 29 December 2023].CloseDeleteEdit

Bash Regular Expressions. *Zach Gollwitzer* [online]. Available from: https://www.zachgollwitzer.com/posts/bash-regular-expressions [accessed 29 December 2023].CloseDeleteEdit

Using 'break' and 'continue' to exit loops in bash. *Network World* [online]. Available from: https://www.networkworld.com/article/971492/using-break-and-continue-to-exit-loops-in-bash.html [accessed 29 December 2023].CloseDeleteEdit

Determine whether a directory exists in Bash. (2023). *Sentry* [online]. Available from: https://sentry.io/answers/determine-whether-a-directory-exists-in-bash/ [accessed 29 December 2023].

Bash - Loops - Documentation. Available from: https://docs.rockylinux.org/books/learning_bash/07-loops/#:~:text=The%20break%20command%20allows%20you,the%20first%20command%20after%20done%20. [accessed 29 December 2023].

Checking the Validity of an IP Address in Linux. Jimmy Azar [online]. from: https://www.baeldung.com/linux/ip-address-test-valid. [accessed 29 December 2023].

NetDiscover: A Powerful Information Gathering Tool in Kali Linux. Available from: https://eightify.app/summary/technology-and-hacking/netdiscover-a-powerful-information-gathering-tool-in-kali-linux [accessed 29 December 2023].CloseDeleteEdit

How to Easily Detect CVEs with Nmap Scripts. (2018). *WonderHowTo* [online]. Available from: https://null-byte.wonderhowto.com/how-to/easily-detect-cves-with-nmap-scripts-0181925/ [accessed 29 December 2023].CloseDeleteEdit

pv command in Linux with Examples. (2020). *GeeksforGeeks* [online], 7 July 2020. Available from: https://www.geeksforgeeks.org/pv-command-in-linux-with-examples/ [accessed 29 December 2023].CloseDeleteEdit

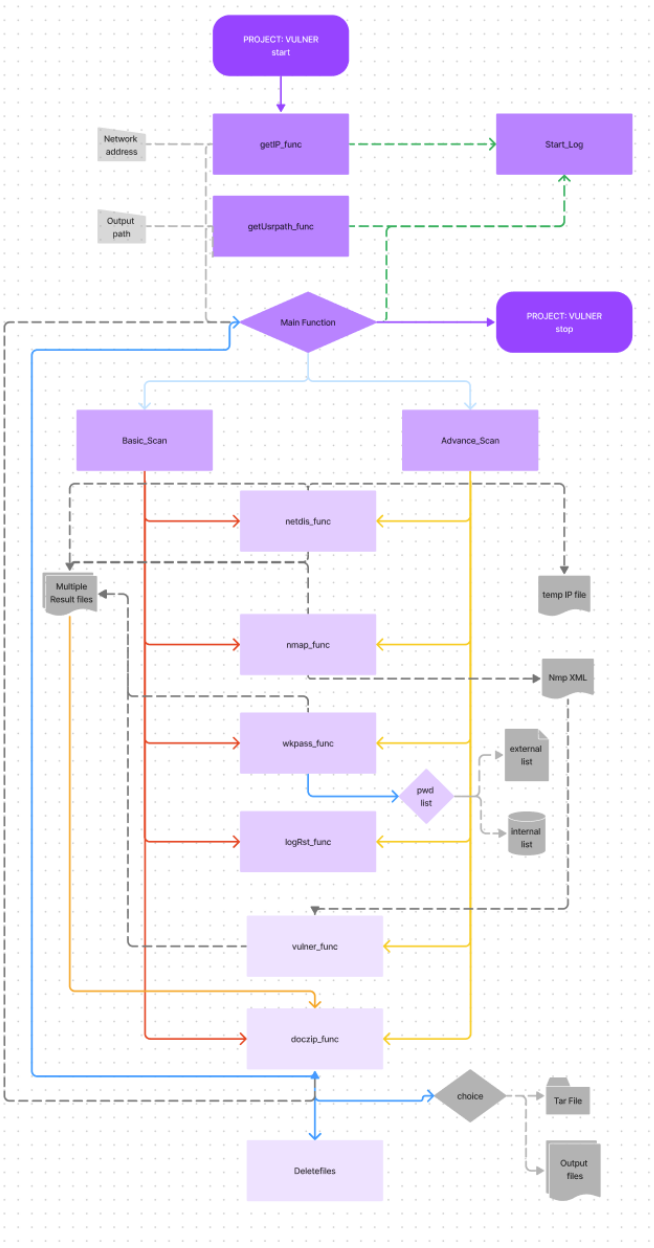
hydra: a very fast network logon cracker which supports many different services | hydra Commands | Man Pages | ManKier. Available from: https://www.mankier.com/1/hydra [accessed 29 December 2023].CloseDeleteEdit

How to Compress Files in Linux | Tar Command. (2017). *GeeksforGeeks* [online], 4 October 2017. Available from: https://www.geeksforgeeks.org/tar-command-linux-examples/ [accessed 30 December 2023].CloseDeleteEdit

Duffy, C. (2015). Answer to 'In bash tee is making function variables local, how do I escape this?' *Stack Overflow* [online], 22 July 2015. Available from: https://stackoverflow.com/a/31552333 [accessed 30 December 2023].CloseDeleteEdit

Appendix

Functional Block diagram



Script

```
#! /bin/bash

function getIP_func
{
read -p 'Enter network to be scanned with the CIDR notation (Eg. 12.123.123.0/24 or 123.123.154.20/16) : '
ip_add
pattern="[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\/[0-9]{1,2}" #rgex pattern for IP address
#Validation of user input
if [[ $ip_add =~ $pattern ]]; then
echo -e "\n Network address matches the pattern : $ip_add \n"
sleep 3s
```

```
else
echo -e "\n Network address does not match the pattern \n"
getIP                                                    # repeat function due to invalid
IP address
fi
}
function netdis_func
{
echo -e '\n Running Net discovery \n'
netdis_results=$(sudo netdiscover -r $ip_add -P)                    # run netdiscovery for given network
echo "$netdis_results" > netdis_results.txt                         # log results to a file
echo "$netdis_results"|grep -oP '([0-9]{1,3}\.){3}[0-9]{1,3}' > IP_only.txt # extract only IP address base on the
pattern
echo -e '\n Running Net discovery completed ! \n'
}
function getUsrpath_func
{
#Validation of user input
while true
read -p 'Provide output directory path : ' usr_path
do
if [ -d $usr_path ]                                              # check if input is a directory
then
echo -e "\n Directory exists. Content within $usr_path \n"
ls -lah $usr_path                                                # print contents within path
break
else
echo -e "\n Directory not exists. \n"                        # repeat function due to invalid dir
continue
fi
done
}
function nmap_func
{
echo -e '\n Running Nmap scan \n'
#nmap_results=$(sudo masscan -p1-65535,U:1-65535 -iL ./IP_only.txt --banners --rate=1000|pv -t)          #
Takes long time to complete
nmap_results=$(sudo nmap -sU -sS -sV -F -iL ./IP_only.txt --open -T4 --version-intensity 0|pv -t)       # get
TCP/UDP port/services details
echo "$nmap_results" > nmap_results.txt                          # log results to a file
sudo nmap -sV -F -iL ./IP_only.txt -oX nmap_resultsXML.xml >/dev/null # get XML for NMAP Vul analysis
echo -e '\n Running Nmap scan completed ! \n'
}
function vulner_func
{
echo -e '\n Running vulnerabilities scan \n'
nmap_Vulresults=$(sudo nmap -sU -sS -sV -F -iL ./IP_only.txt --open -T4 --version-intensity 0 --
script=vulners.nse 2> /dev/null |pv -t) # get CEV details
ssp_results=$(searchsploit --nmap ./nmap_resultsXML.xml 2> /dev/null |pv -t) # get posible expolit details
echo "$nmap_Vulresults" > nmap_Vulresults.txt                    # log results to a file
```

```
echo "$ssp_results" > ssp_results.txt                                    # log results to a file
echo -e '\n Running vulnerabilities scan completed ! \n'
}
function wkpass_func
{
read -p 'Would like to to use the default password list [y/n] : ' choice_wk
echo 'Running weak password scan /n'
Descesion making proceess to supply new lst
case $choice_wk in
n)
while true
read -p 'Please provide full path and file name : ' usr_file
do
if [ -f $usr_file ]                                          # check path provided inpu is a file
then
echo -e "\n File exists : $usr_file \n"
ls -lah $usr_file                                          # print file properties
break
else
echo -e "\n File not exists. \n"
continue
fi
done
;;
y)
        passlst='./wordlst/top-passwords-shortlist.txt'          # default password list
        echo "Existing password list : $passlst"
;;
*)
        echo "Invalid option"
        wkpass_func                                                                  # repeat case for an invalid option
;;

esac
ssh brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for ssh servicer (Kindly be patient)'
weakp_ssh=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt ssh -t6 2> /dev/null |pv -
t)
echo "$weakp_ssh" > weakp_ssh.txt
rdp brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for rdp servicer (Kindly be patient)'
weakp_rdp=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt rdp -t6 2> /dev/null |pv -
t)
echo "$weakp_rdp" > weakp_rdp.txt
ftp brute force, out put of commands not displayed to terminal and counter in place to measure progress
echo 'Testing weak password for ftp servicer (Kindly be patient)'
weakp_ftp=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt ftp -t6 2> /dev/null |pv -t)
```

```
echo "$weakp_ftp" > weakp_ftp.txt

telnet brute force, out put of commands not displayed to terminal and counter in place to measure progress

echo 'Testing weak password for telnet servicer (Kindly be patient)'
#weakp_telnet=$(hydra -L ./wordlst/top-usernames-shortlist.txt  -P $passlst -M ./IP_only.txt telnet -t12 2>
/dev/null |pv -t) # takes longer time for excution
weakp_telnet=$(medusa -U ./wordlst/top-usernames-shortlist.txt -P $passlst -H ./IP_only.txt -M telnet 2>
/dev/null |pv -t)
echo "$weakp_telnet" > weakp_telnet.txt

echo 'Running weak password scan comeplted ! /n'
}

function logRst_func
{

Output of results are displayed at the end

echo -e '\n Output of results \n'
echo -e "\n This is the RESULTS for Netdiscovery : \n $netdis_results"
echo -e "\n This is the RESULTS for Nmap Ports/Services : \n $nmap_results"
echo -e "\n This is the RESULTS for Nmap Vulnerability (CVES): \n $nmap_Vulresults"
echo -e "\n This is the RESULTS for Searchsploit (possible exploit) : \n $ssp_results"
echo -e "\n This is the RESULTS for Weak SSH Password: \n $weakp_ssh"
echo -e "\n This is the RESULTS for Weak RDP Password: \n $weakp_rdp"
echo -e "\n This is the RESULTS for Weak FTP Password: \n $weakp_ftp"
echo -e "\n This is the RESULTS for Weak TELNET Password: \n $weakp_telnet"
echo -e '\n Output of results completed ! \n'

while true
do
read -p 'Would like to search the through the results [y/n] : ' choice_lg

serach of contents within the results with grep

case $choice_lg in
        n)
          break
        ;;

        y)
         read -p "Enter Text to be search : " searcht
         # grep with variable use to filter the results
         echo -e '\\n Output of results \\n'
         echo -e "\\n This is the RESULTS for Netdiscovery : \\n $netdis_results" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Nmap Ports/Services : \\n $nmap_results" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Nmap Vulnerability (CVES): \\n $nmap_Vulresults" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Searchsploit (possible exploit) : \\n $ssp_results" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Weak SSH Password: \\n $weakp_ssh" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Weak RDP Password: \\n $weakp_rdp" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Weak FTP Password: \\n $weakp_ftp" | grep "$searcht"
         echo -e "\\n This is the RESULTS for Weak TELNET Password: \\n $weakp_telnet" | grep "$searcht"
         echo -e '\\n Output of results completed ! \\n'
         continue
        ;;
```

```
        *)
        echo "Invalid option"                    # repeat case for an invalid option
        continue
        ;;

esac


done
}
function doczip_func
{
read -p 'Would like to zip file [y/n] : ' choice_zp
Descesion making proceess to zip files and/or move files to output dir
case $choice_zp in
n)
echo -e '\n Files are not ziped and they are located in : $usr_path \n'
Deletefiles                                        # delete unrequired tempfiles
for count in $(ls |grep -e .txt)                   # get list of txt files from dir
do
echo "$count"
mv $count $usr_path                                # move txt files
done
echo -e '\n List of Files in the folder \n'
ls -lah $usr_path
;;
y)
echo -e '\n Files are ziped and they are located in : $usr_path \n'
Deletefiles                                        # delete unrequired tempfiles
echo -e '\n Running zip compression \n'
for count in $(ls |grep -e .txt -e .xml)           # get list of txt and xml files from dir
do
echo "$count"
tar -uf $(date +"%y%m%d" ).tar $count              # zip multiple logs to a single tar file with current date a the
name
done
mv ./$(date +"%y%m%d" ).tar $usr_path              # move tar file to a output dir
echo -e '\n List of Files in the folder \n'
ls -lah $usr_path
les
echo -e '\n Running zip compression completed !\n'
;;
*)
echo "Invalid option"
doczip
;;
esac
}
function Deletefiles
{
```

```
echo -e "\n Deleting tempfiles \n"
rm -f IP_only.txt                                   #delete temp ip file
rm -f nmap_resultsXML.xml                           # delete temp files
echo -e "\n Deleting tempfiles completed ! \n"
echo -e "\n Deleting tempfiles completed ! \n"
}

start script log

touch ./Penl.log # create log file
(date; hostname) |tr '\n' '\t'|tee -a ./Penl.log #timestamp to log file

#Banner introduction
figlet -mini Project - PENETRATION TESTING
echo -e " ******* S10 - Muhammad Feroz (PRJ: VULNER)********** \n"
echo "
\Start Script/"

Main Function

getIP_func > >(tee -ap ./Penl.log) #get network address and screen output to log file
getUsrpath_func > >(tee -ap ./Penl.log) #get output path and screen output to log file
while true
do
echo "1 for Basic Scan"
echo "2 for Full Scan"
echo "e for Exit Scan"
read -p 'Choose the scan you would like to perform : ' usr_Option #get user option on scan output
case $usr_Option in
1)
echo -e '\n *** RUNNING BASIC SCAN *** \n'
(date; hostname) |tr '\n' '\t'|tee -a ./Penl.log
### Functions ###
sleep 5s
netdis_func > >(tee -ap ./Penl.log) #run netdiscovery and screen output to log file
nmap_func   > >(tee -ap ./Penl.log)                 #run nmap and screen output to log file
wkpass_func > >(tee -ap ./Penl.log)                 #run brute force tools and screen output to log file
logRst_func > >(tee -ap ./Penl.log)             #get outputs to be displayed and screen output to log file
doczip_func > >(tee -ap ./Penl.log)             #run zip options for file and screen output to log file
echo -e '\n *** BASIC SCAN COMPLETED *** \n'
;;
2)
echo -e '\n *** RUNNING FULL SCAN *** \n'
(date; hostname) |tr '\n' '\t'|tee -a ./Penl.log
### Functions ###
sleep 5s
netdis_func > >(tee -ap ./Penl.log)
nmap_func   > >(tee -ap ./Penl.log)
wkpass_func > >(tee -ap ./Penl.log)
vulner_func > >(tee -ap ./Penl.log)             #run vulerbility testing and exploit suggestion and screen output to
log file
logRst_func > >(tee -ap ./Penl.log)
doczip_func > >(tee -ap ./Penl.log)
echo -e '\n *** FULL SCAN COMPLETED *** \n'
```

```
;;
e)
echo "Exit script"
exit                                    #exit script
;;

)
echo "Invalid option"
continue                                #re-run the loop due to invalid option
;;
esac
done
echo "
\End Script/*"
```