Penetration Testing Project

Created by S22 Lam Jin Hong, Ken

Contents

Introduction

This script will map your current network and find different attack vectors. Based on the active hosts, open ports will be identified. Next, the script will find users with weak passwords and potential vulnerabilities based on service detection.

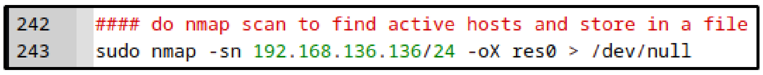
The user will be prompted to provide a list of Usernames and Passwords. The list will be used to check for weak passwords in the network services.

Finally, the user may view the results in the command line or, view it in a saved report file.

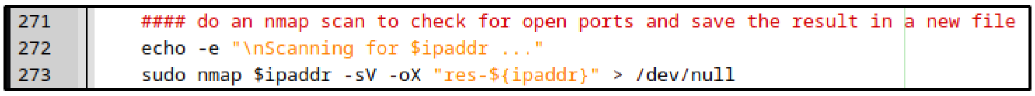
Key Concepts

Tools used: nmap, hydra

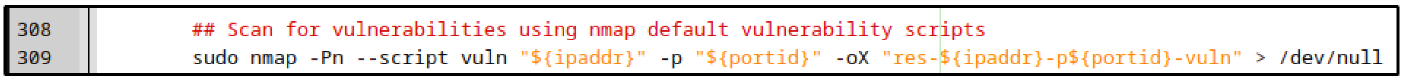
[**nmap**] – at the first part of the script, the nmap tool was used to do active host discovery and port scanning. Next, the default nmap NSE “vuln” script was used to search for potential vulnerabilities in each active host.



```
#### do nmap scan to find active hosts and store in a file
sudo nmap -sn 192.168.136.136/24 -oX res0 > /dev/null
```

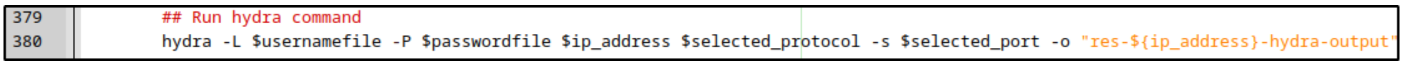


```
    #### do an nmap scan to check for open ports and save the result in a new file
    echo -e "\nScanning for $ipaddr ..."
    sudo nmap $ipaddr -sV -oX "res-${ipaddr}" > /dev/null
```



```
        ## Scan for vulnerabilities using nmap default vulnerability scripts
        sudo nmap -Pn --script vuln "${ipaddr}" -p "${portid}" -oX "res-${ipaddr}-p${portid}-vuln" > /dev/null
```

[**hydra**] – the hydra tool was used to identify any available login service and brute force it.



```
        ## Run hydra command
        hydra -L $usernamefile -P $passwordfile $ip_address $selected_protocol -s $selected_port -o "res-${ip_address}-hydra-output"
```

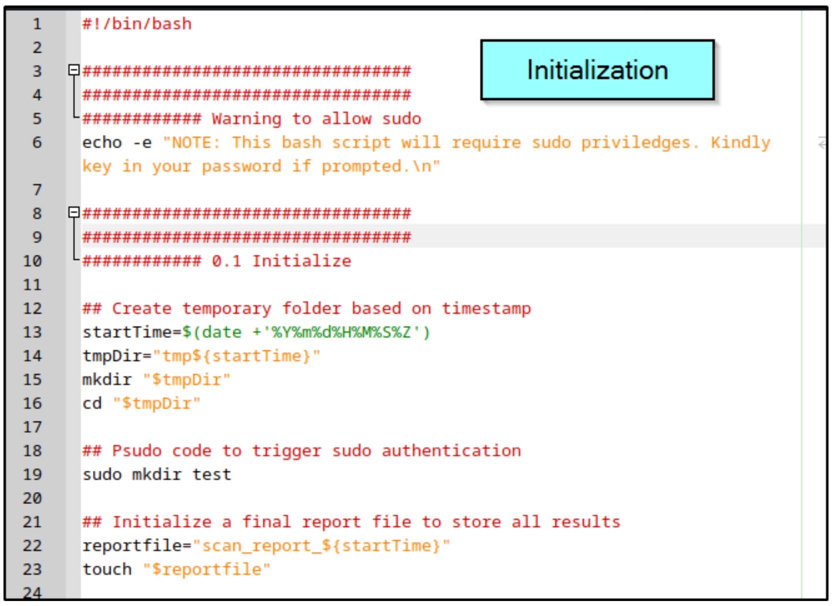
Possible Enhancements

Scope of the search for potential vulnerability could be expanded by using online databases, for example, ***scipag_vulscan*** resource. However, this would require a git clone to copy the repository into the user’s local directory.

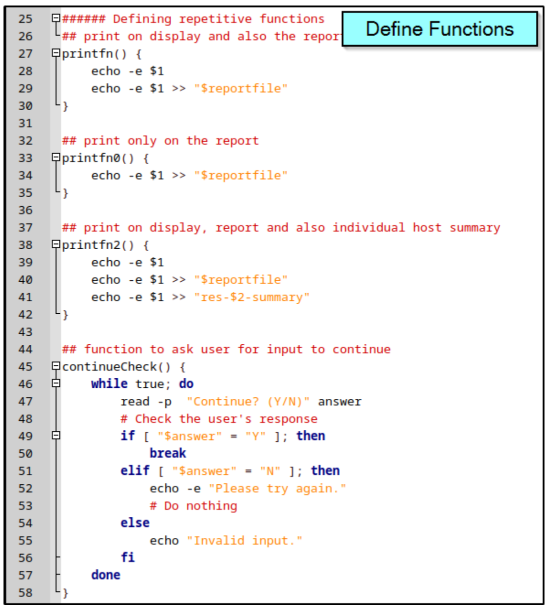
User could be recommended a prepared list of password file templates, and the estimated time it might take to run the brute forcing. This allows the user to make a assessment weighing the scope, time and accuracy.

Multiple host & port scanning tools can be used to further enhance the accuracy of device & port scanning. For example, using nmap UDP scans as well as the massscan tool. However, this would greatly increase the time taken to complete the scan.

1- Initialisation of folders, sudo access and functions

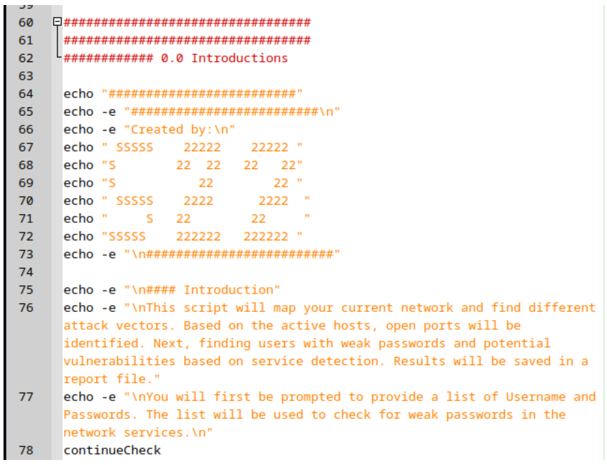


```
#!/bin/bash

##################################
##################################
############ Warning to allow sudo
echo -e "NOTE: This bash script will require sudo priviledges. Kindly key in your password if prompted.\n"

##################################
##################################
############ 0.1 Initialize

## Create temporary folder based on timestamp
startTime=$(date +'%Y%m%d%H%M%S%Z')
tmpDir="tmp${startTime}"
mkdir "$tmpDir"
cd "$tmpDir"

## Psudo code to trigger sudo authentication
sudo mkdir test

## Initialize a final report file to store all results
reportfile="scan_report_${startTime}"
touch "$reportfile"
```

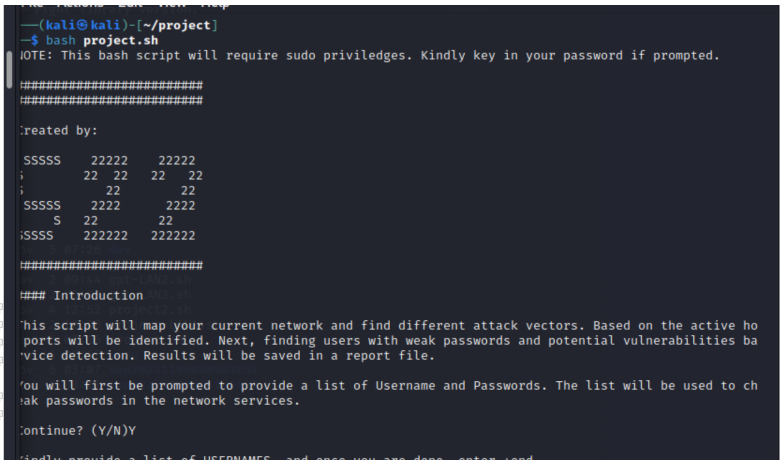


```
###### Defining repetitive functions
## print on display and also the repor
printfn() {
    echo -e $1
    echo -e $1 >> "$reportfile"
}

## print only on the report
printfn0() {
    echo -e $1 >> "$reportfile"
}

## print on display, report and also individual host summary
printfn2() {
    echo -e $1
    echo -e $1 >> "$reportfile"
    echo -e $1 >> "res-$2-summary"
}

## function to ask user for input to continue
continueCheck() {
    while true; do
        read -p  "Continue? (Y/N)" answer
        # Check the user's response
        if [ "$answer" = "Y" ]; then
            break
        elif [ "$answer" = "N" ]; then
            echo -e "Please try again."
            # Do nothing
        else
            echo "Invalid input."
        fi
    done
}
```

2- Simple introductions to the code

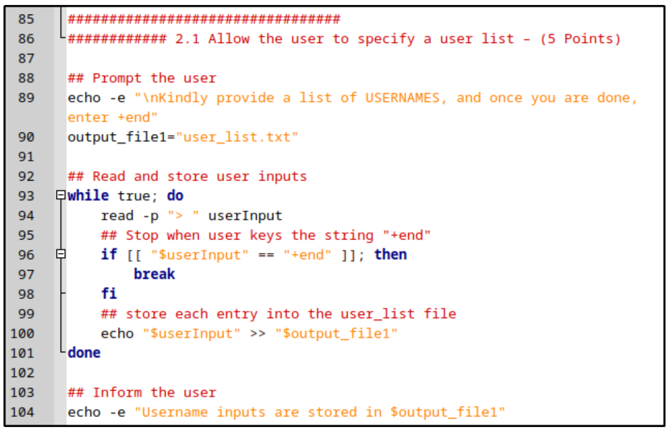


```
##################################
##################################
############ 0.0 Introductions

echo "#########################"
echo -e "#########################\n"
echo -e "Created by:\n"
echo " SSSSS    22222    22222 "
echo "S        22  22   22   22"
echo "S           22         22 "
echo " SSSSS    2222       2222  "
echo "     S   22         22     "
echo "SSSSS    222222   222222 "
echo -e "\n#########################"

echo -e "\n#### Introduction"
echo -e "\nThis script will map your current network and find different attack vectors. Based on the active hosts, open ports will be identified. Next, finding users with weak passwords and potential vulnerabilities based on service detection. Results will be saved in a report file."
echo -e "\nYou will first be prompted to provide a list of Username and Passwords. The list will be used to check for weak passwords in the network services.\n"
continueCheck
```

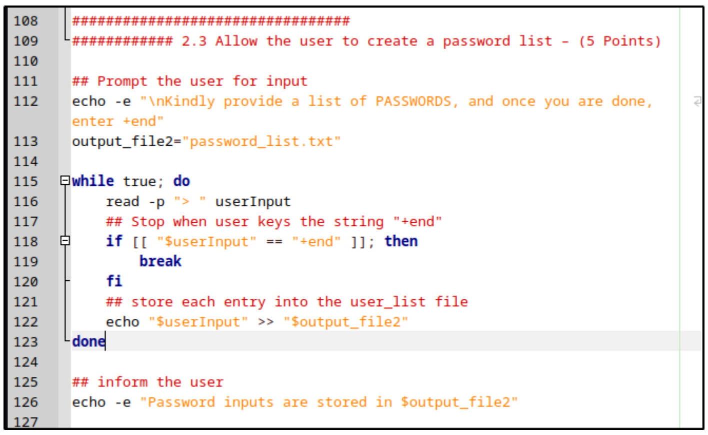


```
──(kali㉿kali)-[~/project]
─$ bash project.sh
IOTE: This bash script will require sudo priviledges. Kindly key in your password if prompted.

#########################
#########################

Created by:

 SSSSS    22222    22222
S        22  22   22   22
S           22         22
 SSSSS    2222       2222
     S   22         22
SSSSS    222222   222222

#########################

#### Introduction

This script will map your current network and find different attack vectors. Based on the active ho
 ports will be identified. Next, finding users with weak passwords and potential vulnerabilities ba
rvice detection. Results will be saved in a report file.

You will first be prompted to provide a list of Username and Passwords. The list will be used to ch
eak passwords in the network services.

Continue? (Y/N)Y
```

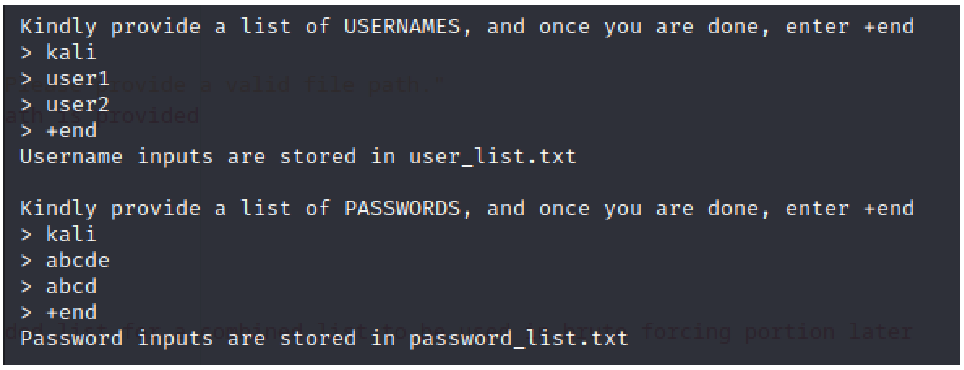
3 – Allow user to specify a list of usernames and list of passwords



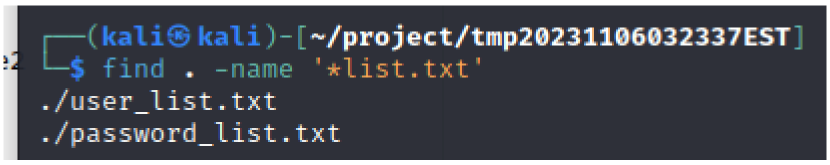
```
###################################
############ 2.1 Allow the user to specify a user list - (5 Points)

## Prompt the user
echo -e "\nKindly provide a list of USERNAMES, and once you are done, enter +end"
output_file1="user_list.txt"

## Read and store user inputs
while true; do
    read -p "> " userInput
    ## Stop when user keys the string "+end"
    if [[ "$userInput" == "+end" ]]; then
        break
    fi
    ## store each entry into the user_list file
    echo "$userInput" >> "$output_file1"
done

## Inform the user
echo -e "Username inputs are stored in $output_file1"
```



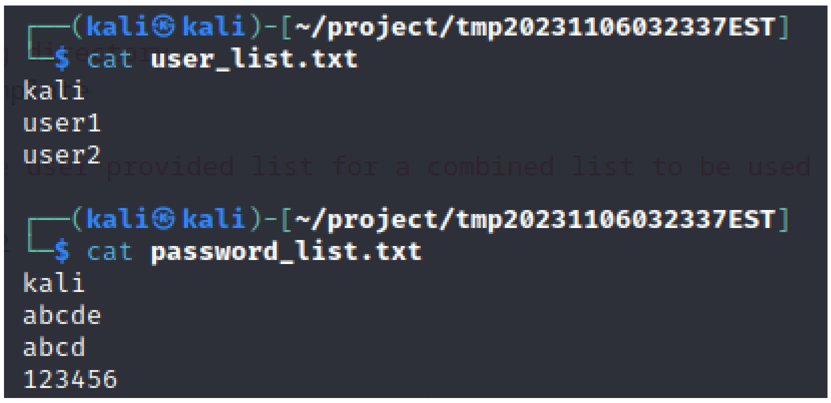
```
###################################
############ 2.3 Allow the user to create a password list - (5 Points)

## Prompt the user for input
echo -e "\nKindly provide a list of PASSWORDS, and once you are done, enter +end"
output_file2="password_list.txt"

while true; do
    read -p "> " userInput
    ## Stop when user keys the string "+end"
    if [[ "$userInput" == "+end" ]]; then
        break
    fi
    ## store each entry into the user_list file
    echo "$userInput" >> "$output_file2"
done

## inform the user
echo -e "Password inputs are stored in $output_file2"
```



```
Kindly provide a list of USERNAMES, and once you are done, enter +end
> kali
> user1
> user2
> +end
Username inputs are stored in user_list.txt

Kindly provide a list of PASSWORDS, and once you are done, enter +end
> kali
> abcde
> abcd
> +end
Password inputs are stored in password_list.txt
```

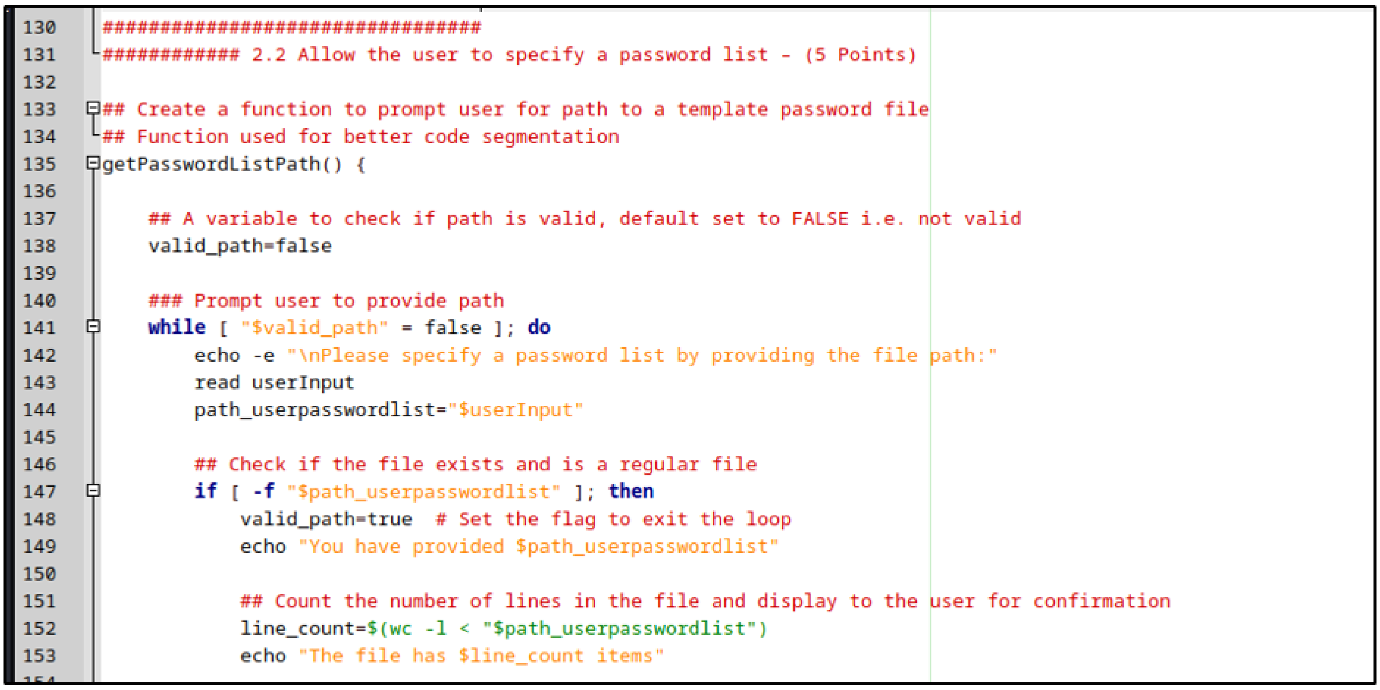


```
┌──(kali㉿kali)-[~/project/tmp20231106032337EST]
└─$ find . -name '*list.txt'
./user_list.txt
./password_list.txt
```

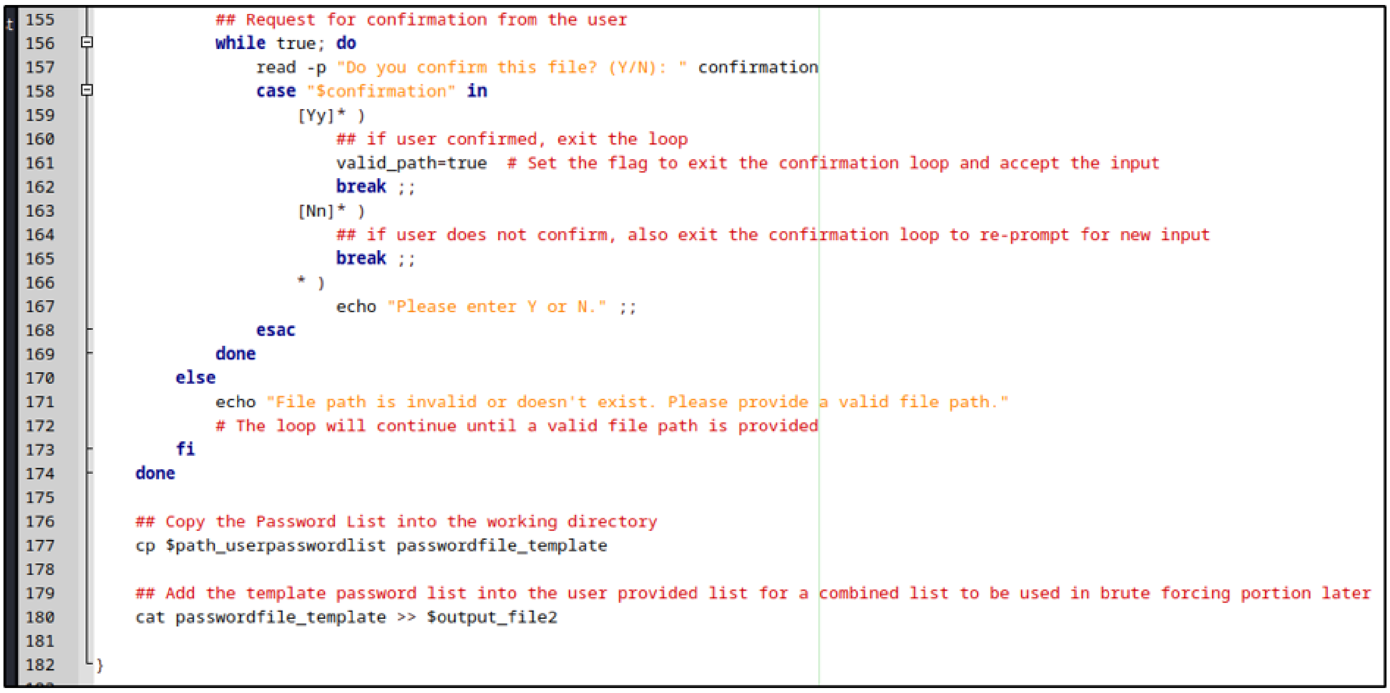


```
┌──(kali㉿kali)-[~/project/tmp20231106032337EST]
└─$ cat user_list.txt
kali
user1
user2

┌──(kali㉿kali)-[~/project/tmp20231106032337EST]
└─$ cat password_list.txt
kali
abcde
abcd
123456
```

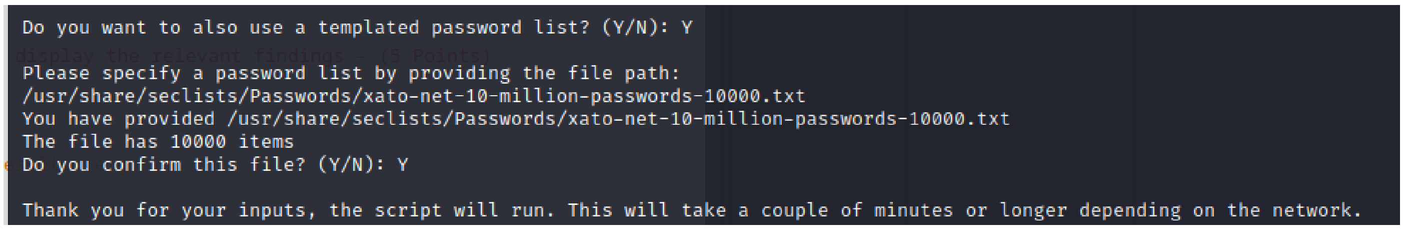
4 – Allow the user to specify a password list template



```bash
#################################
############ 2.2 Allow the user to specify a password list - (5 Points)

## Create a function to prompt user for path to a template password file
## Function used for better code segmentation
getPasswordListPath() {

    ## A variable to check if path is valid, default set to FALSE i.e. not valid
    valid_path=false

    ### Prompt user to provide path
    while [ "$valid_path" = false ]; do
        echo -e "\nPlease specify a password list by providing the file path:"
        read userInput
        path_userpasswordlist="$userInput"

        ## Check if the file exists and is a regular file
        if [ -f "$path_userpasswordlist" ]; then
            valid_path=true  # Set the flag to exit the loop
            echo "You have provided $path_userpasswordlist"

            ## Count the number of lines in the file and display to the user for confirmation
            line_count=$(wc -l < "$path_userpasswordlist")
            echo "The file has $line_count items"
```

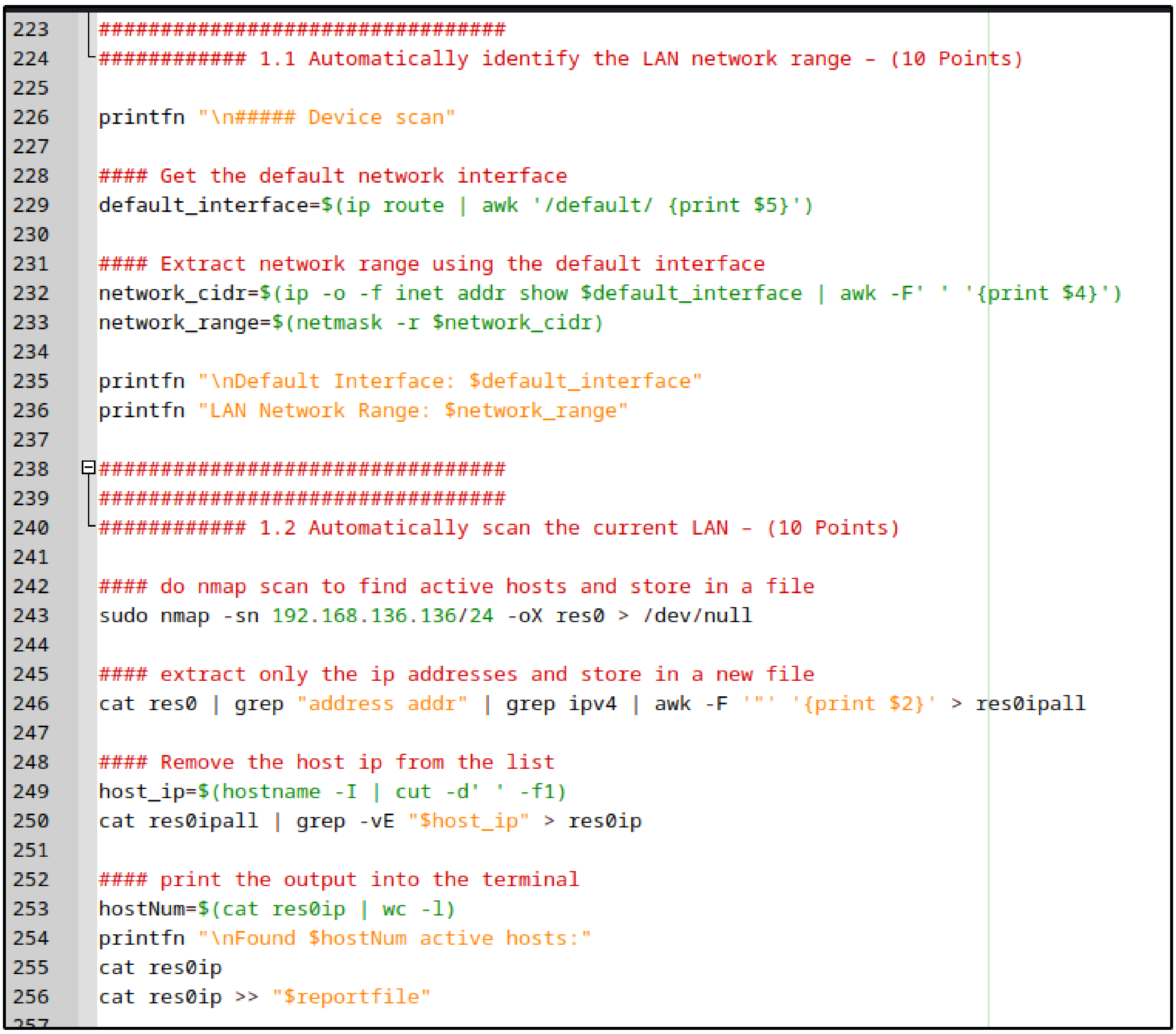


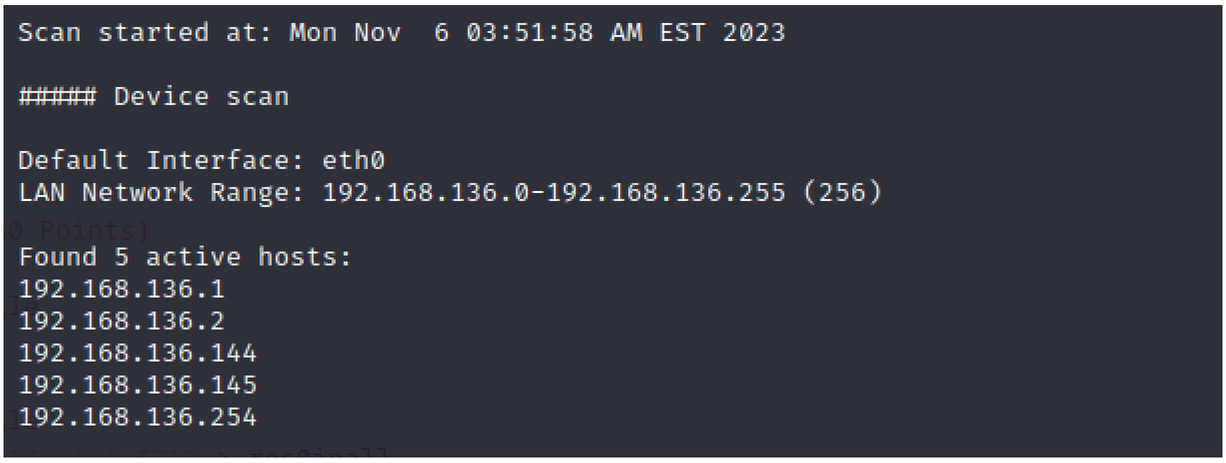
```bash
            ## Request for confirmation from the user
            while true; do
                read -p "Do you confirm this file? (Y/N): " confirmation
                case "$confirmation" in
                    [Yy]* )
                        ## if user confirmed, exit the loop
                        valid_path=true  # Set the flag to exit the confirmation loop and accept the input
                        break ;;
                    [Nn]* )
                        ## if user does not confirm, also exit the confirmation loop to re-prompt for new input
                        break ;;
                    * )
                        echo "Please enter Y or N." ;;
                esac
            done
        else
            echo "File path is invalid or doesn't exist. Please provide a valid file path."
            # The loop will continue until a valid file path is provided
        fi
    done

    ## Copy the Password List into the working directory
    cp $path_userpasswordlist passwordfile_template

    ## Add the template password list into the user provided list for a combined list to be used in brute forcing portion later
    cat passwordfile_template >> $output_file2

}
```



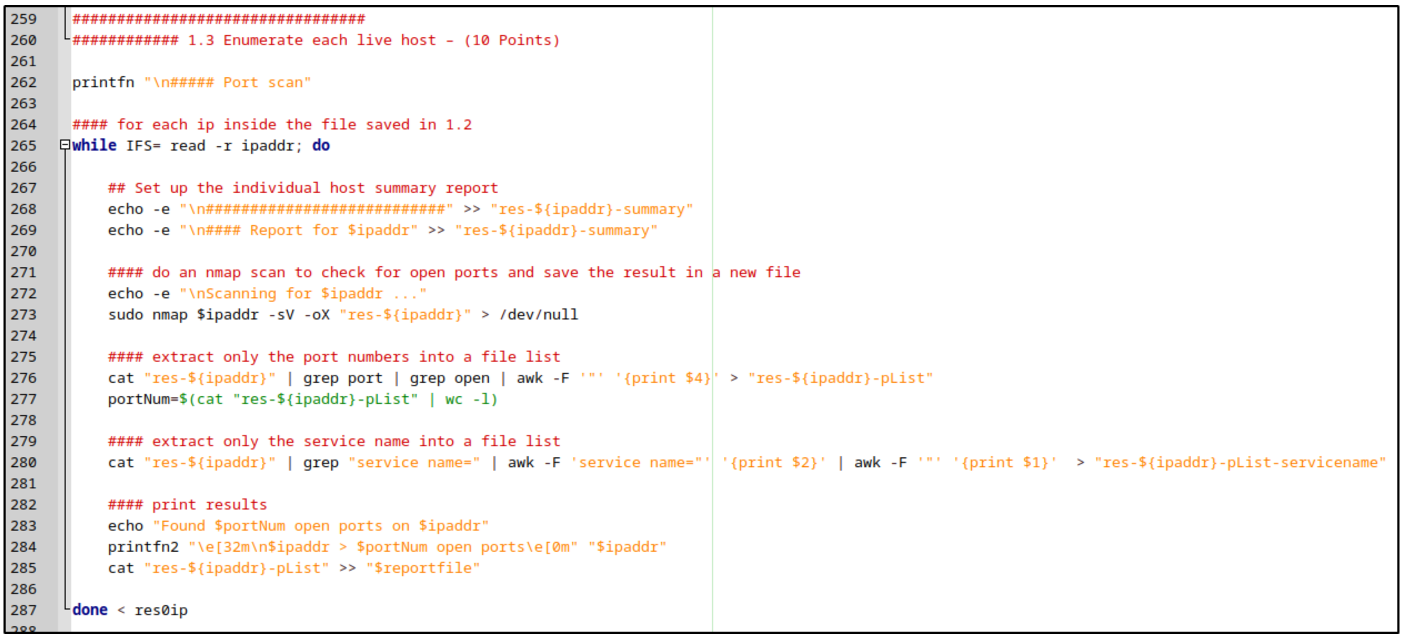
```
Do you want to also use a templated password list? (Y/N): Y

Please specify a password list by providing the file path:
/usr/share/seclists/Passwords/xato-net-10-million-passwords-10000.txt
You have provided /usr/share/seclists/Passwords/xato-net-10-million-passwords-10000.txt
The file has 10000 items
Do you confirm this file? (Y/N): Y

Thank you for your inputs, the script will run. This will take a couple of minutes or longer depending on the network.
```

5 – Start the scan for devices

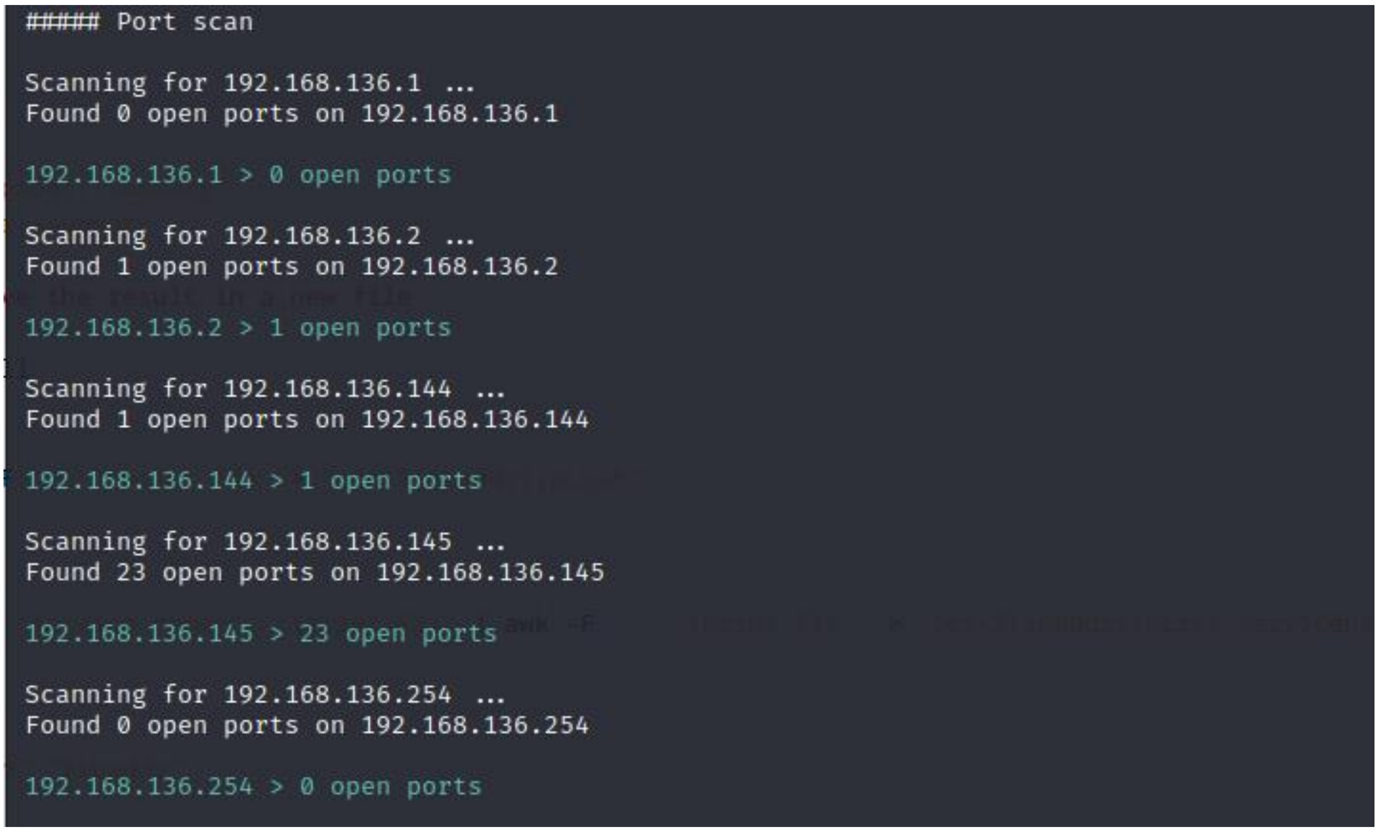

```
#################################
############ 1.1 Automatically identify the LAN network range - (10 Points)

printfn "\n##### Device scan"

#### Get the default network interface
default_interface=$(ip route | awk '/default/ {print $5}')

#### Extract network range using the default interface
network_cidr=$(ip -o -f inet addr show $default_interface | awk -F' ' '{print $4}')
network_range=$(netmask -r $network_cidr)

printfn "\nDefault Interface: $default_interface"
printfn "LAN Network Range: $network_range"

#################################
#################################
############ 1.2 Automatically scan the current LAN - (10 Points)

#### do nmap scan to find active hosts and store in a file
sudo nmap -sn 192.168.136.136/24 -oX res0 > /dev/null

#### extract only the ip addresses and store in a new file
cat res0 | grep "address addr" | grep ipv4 | awk -F '"' '{print $2}' > res0ipall

#### Remove the host ip from the list
host_ip=$(hostname -I | cut -d' ' -f1)
cat res0ipall | grep -vE "$host_ip" > res0ip

#### print the output into the terminal
hostNum=$(cat res0ip | wc -l)
printfn "\nFound $hostNum active hosts:"
cat res0ip
cat res0ip >> "$reportfile"
```


```
Scan started at: Mon Nov  6 03:51:58 AM EST 2023

##### Device scan

Default Interface: eth0
LAN Network Range: 192.168.136.0-192.168.136.255 (256)

Found 5 active hosts:
192.168.136.1
192.168.136.2
192.168.136.144
192.168.136.145
192.168.136.254
```

6 – Start scan of ports on each device found

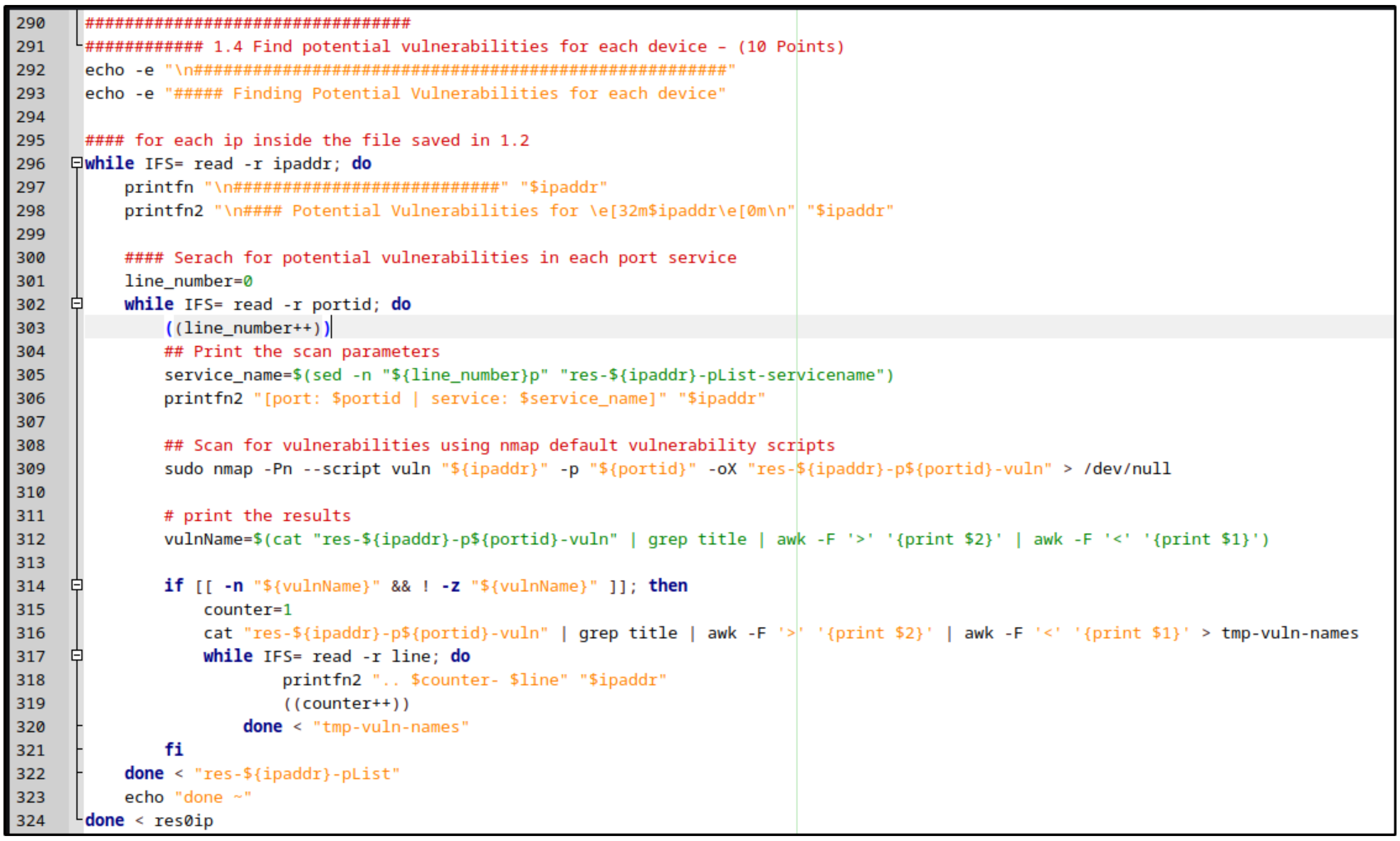


```
##################################
############ 1.3 Enumerate each live host - (10 Points)

printfn "\n##### Port scan"

#### for each ip inside the file saved in 1.2
while IFS= read -r ipaddr; do

    ## Set up the individual host summary report
    echo -e "\n###########################" >> "res-${ipaddr}-summary"
    echo -e "\n#### Report for $ipaddr" >> "res-${ipaddr}-summary"

    #### do an nmap scan to check for open ports and save the result in a new file
    echo -e "\nScanning for $ipaddr ..."
    sudo nmap $ipaddr -sV -oX "res-${ipaddr}" > /dev/null

    #### extract only the port numbers into a file list
    cat "res-${ipaddr}" | grep port | grep open | awk -F '"' '{print $4}' > "res-${ipaddr}-pList"
    portNum=$(cat "res-${ipaddr}-pList" | wc -l)

    #### extract only the service name into a file list
    cat "res-${ipaddr}" | grep "service name=" | awk -F 'service name="' '{print $2}' | awk -F '"' '{print $1}'  > "res-${ipaddr}-pList-servicename"

    #### print results
    echo "Found $portNum open ports on $ipaddr"
    printfn2 "\e[32m\n$ipaddr > $portNum open ports\e[0m" "$ipaddr"
    cat "res-${ipaddr}-pList" >> "$reportfile"

done < res0ip
```

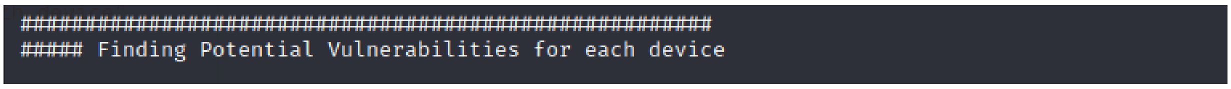


```
##### Port scan

Scanning for 192.168.136.1 ...
Found 0 open ports on 192.168.136.1

192.168.136.1 > 0 open ports

Scanning for 192.168.136.2 ...
Found 1 open ports on 192.168.136.2

192.168.136.2 > 1 open ports

Scanning for 192.168.136.144 ...
Found 1 open ports on 192.168.136.144

192.168.136.144 > 1 open ports

Scanning for 192.168.136.145 ...
Found 23 open ports on 192.168.136.145

192.168.136.145 > 23 open ports

Scanning for 192.168.136.254 ...
Found 0 open ports on 192.168.136.254

192.168.136.254 > 0 open ports
```

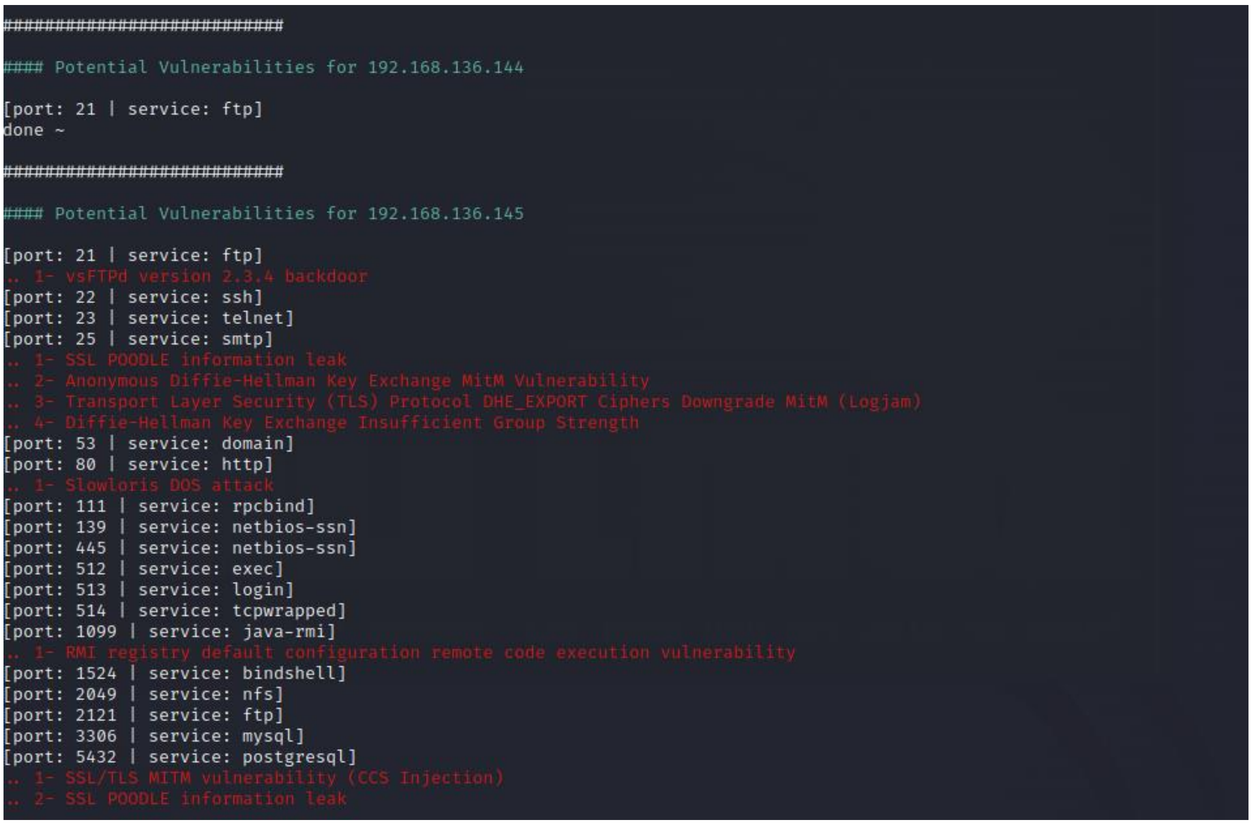
7 – Find potential vulnerabilities in each device



```bash
##################################
############ 1.4 Find potential vulnerabilities for each device - (10 Points)
echo -e "\n#######################################################"
echo -e "##### Finding Potential Vulnerabilities for each device"

#### for each ip inside the file saved in 1.2
while IFS= read -r ipaddr; do
    printfn "\n############################" "$ipaddr"
    printfn2 "\n#### Potential Vulnerabilities for \e[32m$ipaddr\e[0m\n" "$ipaddr"

    #### Serach for potential vulnerabilities in each port service
    line_number=0
    while IFS= read -r portid; do
        ((line_number++))
        ## Print the scan parameters
        service_name=$(sed -n "${line_number}p" "res-${ipaddr}-pList-servicename")
        printfn2 "[port: $portid | service: $service_name]" "$ipaddr"

        ## Scan for vulnerabilities using nmap default vulnerability scripts
        sudo nmap -Pn --script vuln "${ipaddr}" -p "${portid}" -oX "res-${ipaddr}-p${portid}-vuln" > /dev/null

        # print the results
        vulnName=$(cat "res-${ipaddr}-p${portid}-vuln" | grep title | awk -F '>' '{print $2}' | awk -F '<' '{print $1}')

        if [[ -n "${vulnName}" && ! -z "${vulnName}" ]]; then
            counter=1
            cat "res-${ipaddr}-p${portid}-vuln" | grep title | awk -F '>' '{print $2}' | awk -F '<' '{print $1}' > tmp-vuln-names
            while IFS= read -r line; do
                    printfn2 ".. $counter- $line" "$ipaddr"
                    ((counter++))
                done < "tmp-vuln-names"
        fi
    done < "res-${ipaddr}-pList"
    echo "done ~"
done < res0ip
```

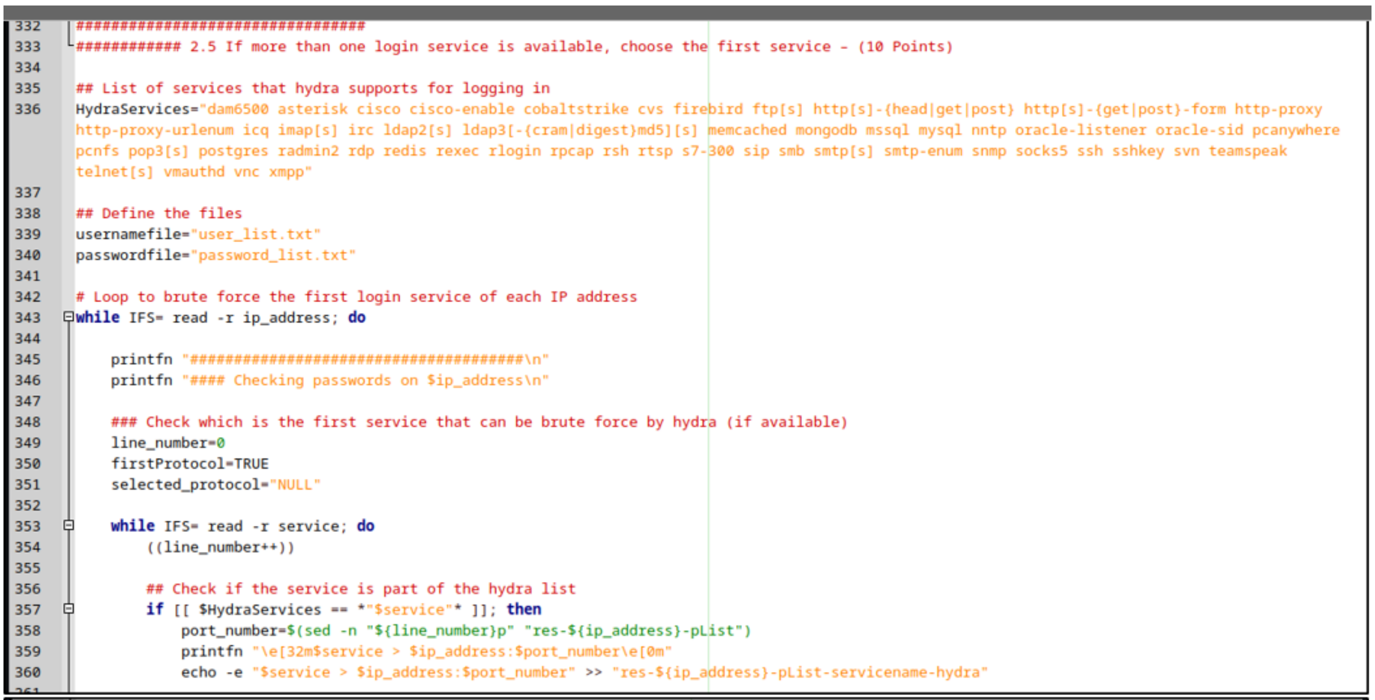


```
#######################################################
##### Finding Potential Vulnerabilities for each device
```

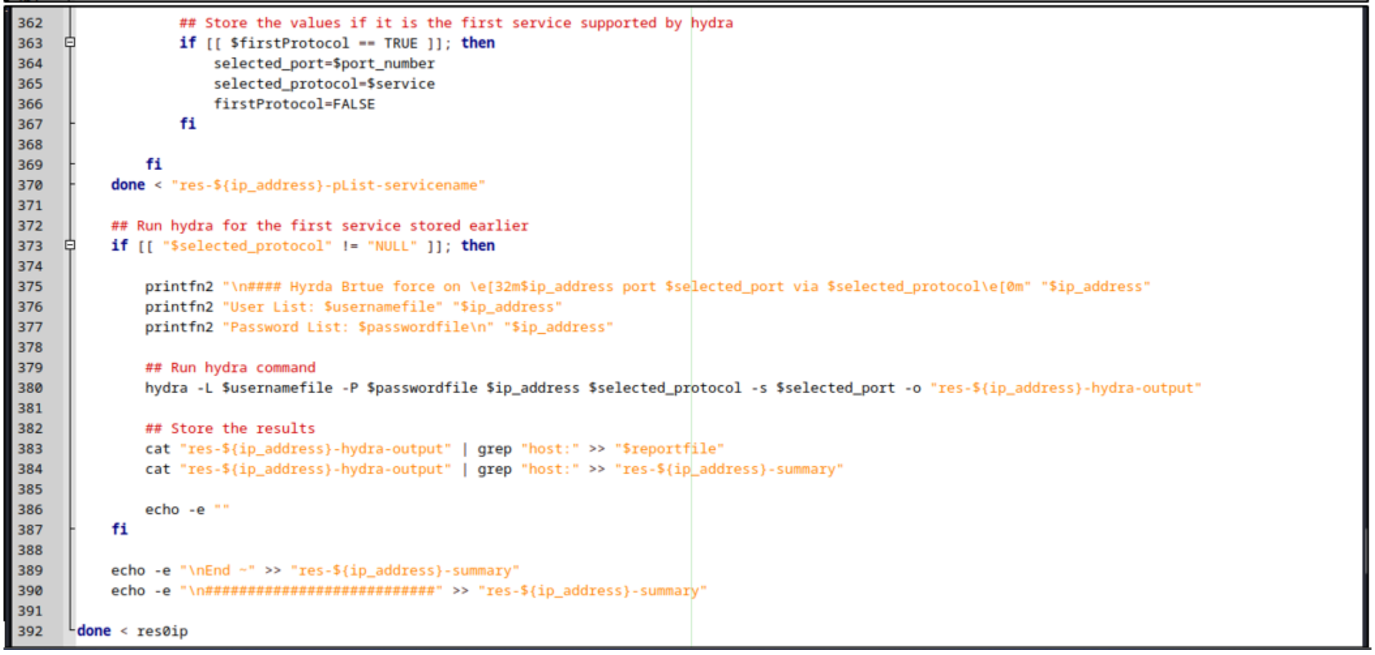


```
############################

#### Potential Vulnerabilities for 192.168.136.144

[port: 21 | service: ftp]
done ~

############################

#### Potential Vulnerabilities for 192.168.136.145

[port: 21 | service: ftp]
.. 1- vsFTPd version 2.3.4 backdoor
[port: 22 | service: ssh]
[port: 23 | service: telnet]
[port: 25 | service: smtp]
.. 1- SSL POODLE information leak
.. 2- Anonymous Diffie-Hellman Key Exchange MitM Vulnerability
.. 3- Transport Layer Security (TLS) Protocol DHE_EXPORT Ciphers Downgrade MitM (Logjam)
.. 4- Diffie-Hellman Key Exchange Insufficient Group Strength
[port: 53 | service: domain]
[port: 80 | service: http]
.. 1- Slowloris DOS attack
[port: 111 | service: rpcbind]
[port: 139 | service: netbios-ssn]
[port: 445 | service: netbios-ssn]
[port: 512 | service: exec]
[port: 513 | service: login]
[port: 514 | service: tcpwrapped]
[port: 1099 | service: java-rmi]
.. 1- RMI registry default configuration remote code execution vulnerability
[port: 1524 | service: bindshell]
[port: 2049 | service: nfs]
[port: 2121 | service: ftp]
[port: 3306 | service: mysql]
[port: 5432 | service: postgresql]
.. 1- SSL/TLS MITM vulnerability (CCS Injection)
.. 2- SSL POODLE information leak
```

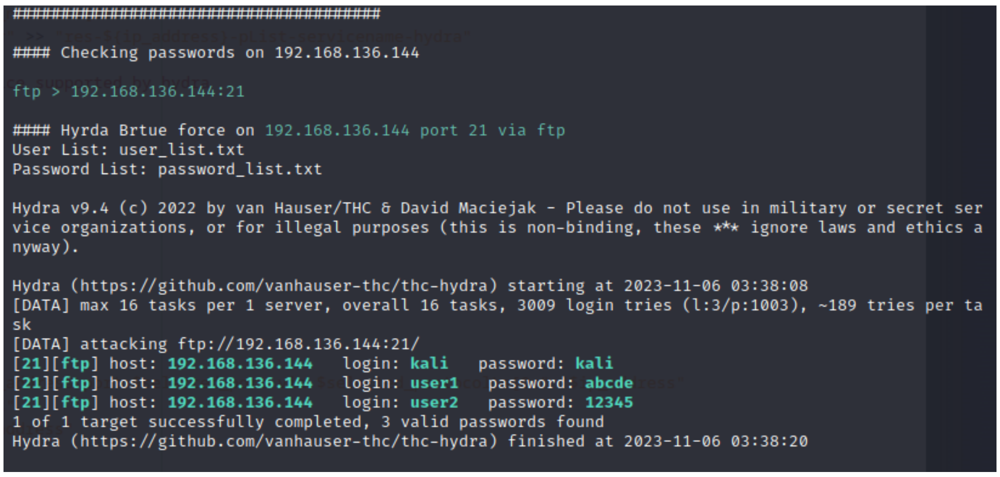
8 – Brute force the first available login service



```bash
##################################
############ 2.5 If more than one login service is available, choose the first service - (10 Points)

## List of services that hydra supports for logging in
HydraServices="dam6500 asterisk cisco cisco-enable cobaltstrike cvs firebird ftp[s] 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] memcached mongodb mssql mysql nntp oracle-listener oracle-sid pcanywhere pcnfs pop3[s] postgres radmin2 rdp redis rexec rlogin rpcap rsh rtsp s7-300 sip smb smtp[s] smtp-enum snmp socks5 ssh sshkey svn teamspeak telnet[s] vmauthd vnc xmpp"

## Define the files
usernamefile="user_list.txt"
passwordfile="password_list.txt"

# Loop to brute force the first login service of each IP address
while IFS= read -r ip_address; do

    printfn "########################################\n"
    printfn "#### Checking passwords on $ip_address\n"

    ### Check which is the first service that can be brute force by hydra (if available)
    line_number=0
    firstProtocol=TRUE
    selected_protocol="NULL"

    while IFS= read -r service; do
        ((line_number++))

        ## Check if the service is part of the hydra list
        if [[ $HydraServices == *"$service"* ]]; then
            port_number=$(sed -n "${line_number}p" "res-${ip_address}-pList")
            printfn "\e[32m$service > $ip_address:$port_number\e[0m"
            echo -e "$service > $ip_address:$port_number" >> "res-${ip_address}-pList-servicename-hydra"
```

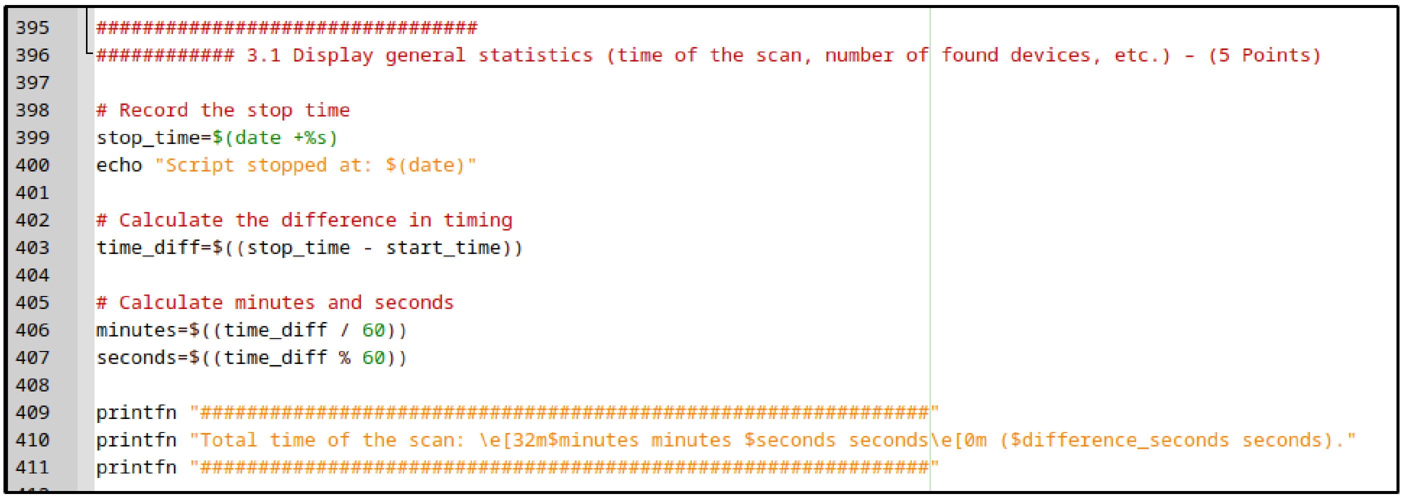


```bash
            ## Store the values if it is the first service supported by hydra
            if [[ $firstProtocol == TRUE ]]; then
                selected_port=$port_number
                selected_protocol=$service
                firstProtocol=FALSE
            fi

        fi
    done < "res-${ip_address}-pList-servicename"

    ## Run hydra for the first service stored earlier
    if [[ "$selected_protocol" != "NULL" ]]; then

        printfn2 "\n#### Hyrda Brtue force on \e[32m$ip_address port $selected_port via $selected_protocol\e[0m" "$ip_address"
        printfn2 "User List: $usernamefile" "$ip_address"
        printfn2 "Password List: $passwordfile\n" "$ip_address"

        ## Run hydra command
        hydra -L $usernamefile -P $passwordfile $ip_address $selected_protocol -s $selected_port -o "res-${ip_address}-hydra-output"

        ## Store the results
        cat "res-${ip_address}-hydra-output" | grep "host:" >> "$reportfile"
        cat "res-${ip_address}-hydra-output" | grep "host:" >> "res-${ip_address}-summary"

        echo -e ""
    fi

    echo -e "\nEnd ~" >> "res-${ip_address}-summary"
    echo -e "\n############################" >> "res-${ip_address}-summary"

done < res0ip
```

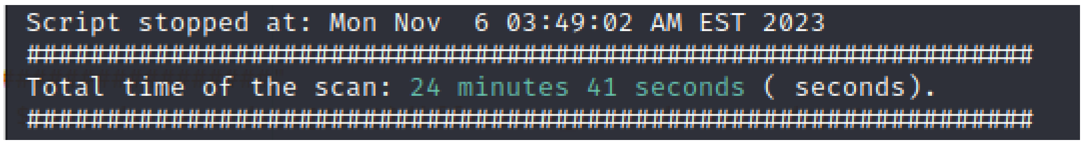


```
###########################################

#### Checking passwords on 192.168.136.144

ftp > 192.168.136.144:21

#### Hyrda Brtue force on 192.168.136.144 port 21 via ftp
User List: user_list.txt
Password List: password_list.txt

Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret ser
vice organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics a
nyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-11-06 03:38:08
[DATA] max 16 tasks per 1 server, overall 16 tasks, 3009 login tries (l:3/p:1003), ~189 tries per ta
sk
[DATA] attacking ftp://192.168.136.144:21/
[21][ftp] host: 192.168.136.144   login: kali   password: kali
[21][ftp] host: 192.168.136.144   login: user1   password: abcde
[21][ftp] host: 192.168.136.144   login: user2   password: 12345
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-11-06 03:38:20
```

9 – Display the total time of the scan

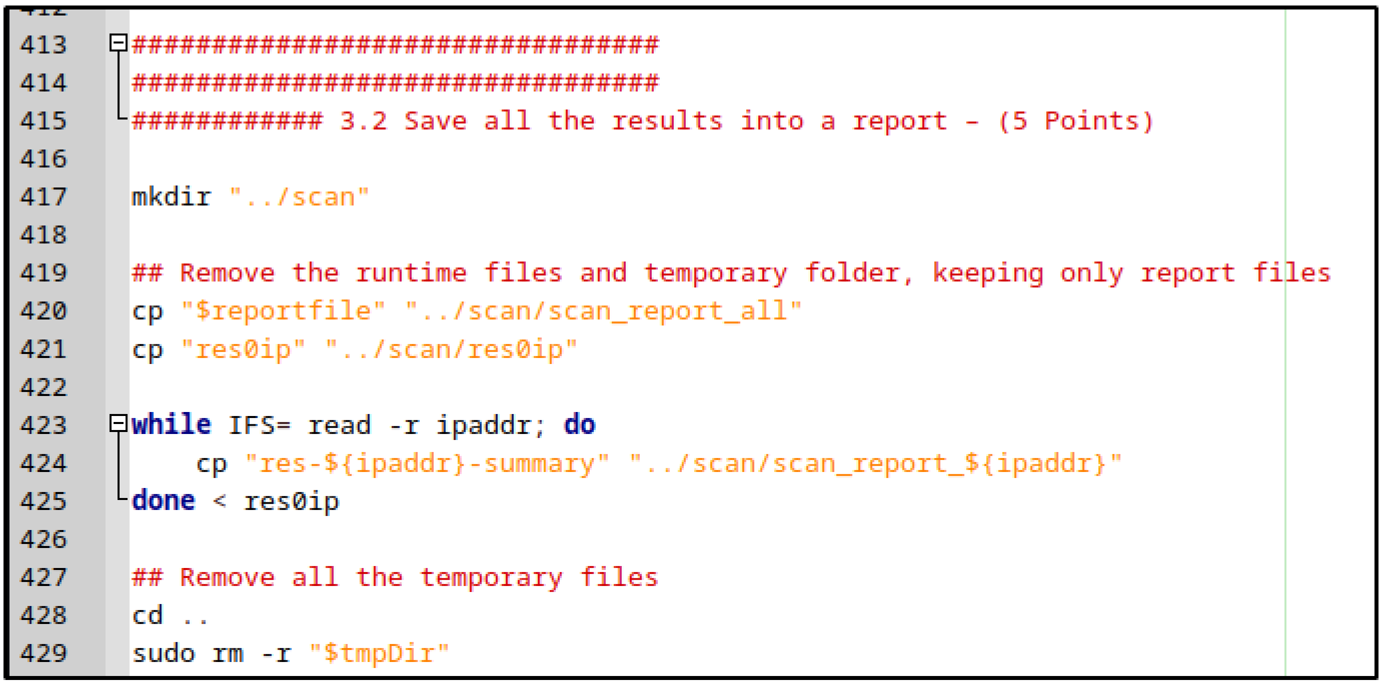


```
####################################
############ 3.1 Display general statistics (time of the scan, number of found devices, etc.) - (5 Points)

# Record the stop time
stop_time=$(date +%s)
echo "Script stopped at: $(date)"

# Calculate the difference in timing
time_diff=$((stop_time - start_time))

# Calculate minutes and seconds
minutes=$((time_diff / 60))
seconds=$((time_diff % 60))

printfn "######################################################################"
printfn "Total time of the scan: \e[32m$minutes minutes $seconds seconds\e[0m ($difference_seconds seconds)."
printfn "######################################################################"
```

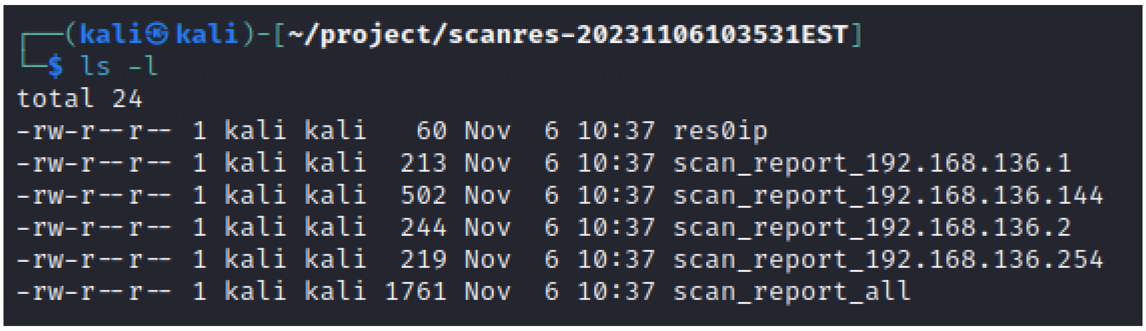


```
Script stopped at: Mon Nov  6 03:49:02 AM EST 2023
##############################################################################
Total time of the scan: 24 minutes 41 seconds ( seconds).
##############################################################################
```

10 – Save the essential report files and delete the rest

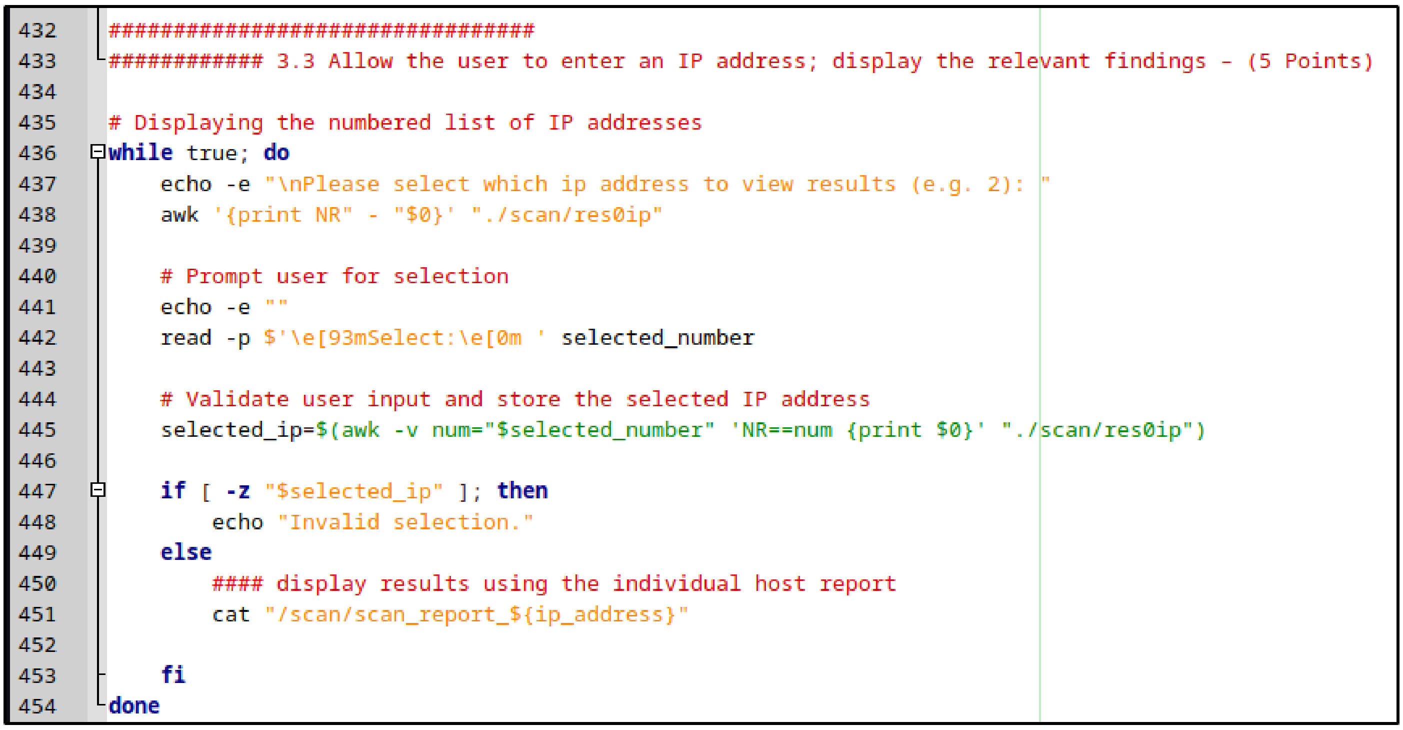


```
####################################
####################################
############ 3.2 Save all the results into a report - (5 Points)

mkdir "../scan"

## Remove the runtime files and temporary folder, keeping only report files
cp "$reportfile" "../scan/scan_report_all"
cp "res0ip" "../scan/res0ip"

while IFS= read -r ipaddr; do
    cp "res-${ipaddr}-summary" "../scan/scan_report_${ipaddr}"
done < res0ip

## Remove all the temporary files
cd ..
sudo rm -r "$tmpDir"
```

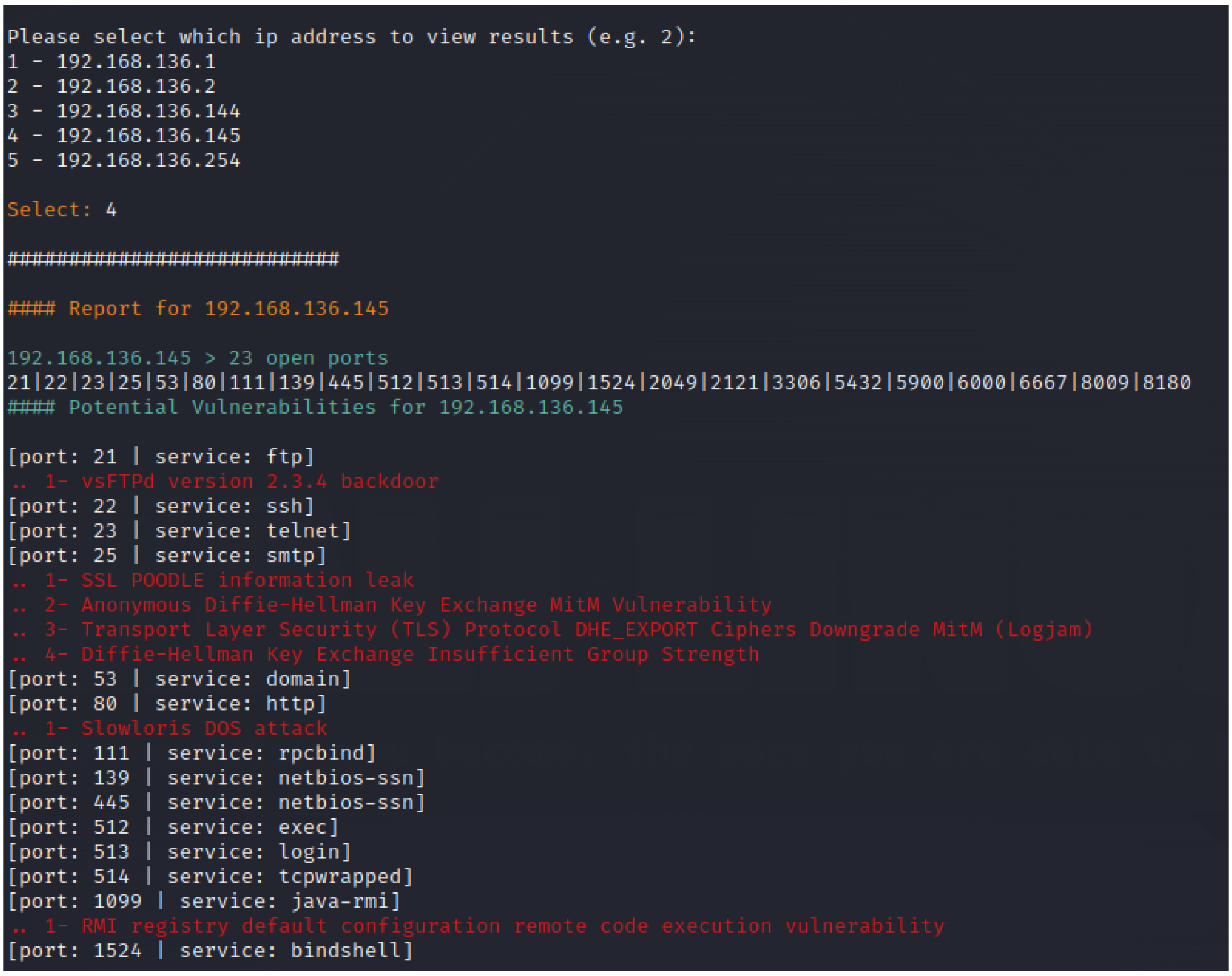


```
┌──(kali㉿kali)-[~/project/scanres-20231106103531EST]
└─$ ls -l
total 24
-rw-r--r-- 1 kali kali   60 Nov  6 10:37 res0ip
-rw-r--r-- 1 kali kali  213 Nov  6 10:37 scan_report_192.168.136.1
-rw-r--r-- 1 kali kali  502 Nov  6 10:37 scan_report_192.168.136.144
-rw-r--r-- 1 kali kali  244 Nov  6 10:37 scan_report_192.168.136.2
-rw-r--r-- 1 kali kali  219 Nov  6 10:37 scan_report_192.168.136.254
-rw-r--r-- 1 kali kali 1761 Nov  6 10:37 scan_report_all
```

11 – Allow the user to select an ip address to view the results



```
##################################
############ 3.3 Allow the user to enter an IP address; display the relevant findings - (5 Points)

# Displaying the numbered list of IP addresses
while true; do
    echo -e "\nPlease select which ip address to view results (e.g. 2): "
    awk '{print NR" - "$0}' "./scan/res0ip"

    # Prompt user for selection
    echo -e ""
    read -p $'\e[93mSelect:\e[0m ' selected_number

    # Validate user input and store the selected IP address
    selected_ip=$(awk -v num="$selected_number" 'NR==num {print $0}' "./scan/res0ip")

    if [ -z "$selected_ip" ]; then
        echo "Invalid selection."
    else
        #### display results using the individual host report
        cat "/scan/scan_report_${ip_address}"

    fi
done
```



```
Please select which ip address to view results (e.g. 2):
1 - 192.168.136.1
2 - 192.168.136.2
3 - 192.168.136.144
4 - 192.168.136.145
5 - 192.168.136.254

Select: 4

###########################

#### Report for 192.168.136.145

192.168.136.145 > 23 open ports
21|22|23|25|53|80|111|139|445|512|513|514|1099|1524|2049|2121|3306|5432|5900|6000|6667|8009|8180
#### Potential Vulnerabilities for 192.168.136.145

[port: 21 | service: ftp]
.. 1- vsFTPd version 2.3.4 backdoor
[port: 22 | service: ssh]
[port: 23 | service: telnet]
[port: 25 | service: smtp]
.. 1- SSL POODLE information leak
.. 2- Anonymous Diffie-Hellman Key Exchange MitM Vulnerability
.. 3- Transport Layer Security (TLS) Protocol DHE_EXPORT Ciphers Downgrade MitM (Logjam)
.. 4- Diffie-Hellman Key Exchange Insufficient Group Strength
[port: 53 | service: domain]
[port: 80 | service: http]
.. 1- Slowloris DOS attack
[port: 111 | service: rpcbind]
[port: 139 | service: netbios-ssn]
[port: 445 | service: netbios-ssn]
[port: 512 | service: exec]
[port: 513 | service: login]
[port: 514 | service: tcpwrapped]
[port: 1099 | service: java-rmi]
.. 1- RMI registry default configuration remote code execution vulnerability
[port: 1524 | service: bindshell]
```