Ensure that the VPN is working

```
Microsoft Windows [Version 10.0.22635.4515]
(c) Microsoft Corporation. All rights reserved.

C:\Users\20115>ssh mallory@neuseeland.informatik.tu-cottbus.de -p 1335
The authenticity of host '[neuseeland.informatik.tu-cottbus.de]:1335 ([141.43.202.225]:1335)' can't be established.
ED25519 key fingerprint is SHA256:p+iqeHEsLFhLfKdlQpkYbsfFn/pn48OkE0uUde7FWM4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[neuseeland.informatik.tu-cottbus.de]:1335' (ED25519) to the list of known hosts.
mallory@neuseeland.informatik.tu-cottbus.de's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
mallory@4d097f366d6e:~$ pwd
/home/mallory
mallory@4d097f366d6e:~$ ls
mallory@4d097f366d6e:~$ whoami
mallory
mallory@4d097f366d6e:~$
```

useful tables


```
Microsoft Windows [Version 10.0.22635.4515]
(c) Microsoft Corporation. All rights reserved.

C:\Users\20115>ssh mallory@neuseeland.informatik.tu-cottbus.de -p 1335
The authenticity of host '[neuseeland.informatik.tu-cottbus.de]:1335 ([141.43.202.225]:1335)' can't be established.
ED25519 key fingerprint is SHA256:p+iqeHEsLFhLfKdlQpkYbsfFn/pn48OkE0uUde7FWM4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[neuseeland.informatik.tu-cottbus.de]:1335' (ED25519) to the list of known hosts.
mallory@neuseeland.informatik.tu-cottbus.de's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
mallory@4d097f366d6e:~$ pwd
/home/mallory
mallory@4d097f366d6e:~$ ls
mallory@4d097f366d6e:~$ whoami
mallory
mallory@4d097f366d6e:~$
```


Summary and Examples

The table below offers a summary of the command line options that we covered.

Command	Explanation
`tcpdump host IP` or `tcpdump host HOSTNAME`	Filters packets by IP address or hostname
`tcpdump src host IP` or	Filters packets by a specific source host
`tcpdump dst host IP`	Filters packets by a specific destination host
`tcpdump port PORT_NUMBER`	Filters packets by port number
`tcpdump src port PORT_NUMBER`	Filters packets by the specified source port number
`tcpdump dst port PORT_NUMBER`	Filters packets by the specified destination port number
`tcpdump PROTOCOL`	Filters packets by protocol; examples include `ip`, `ip6`, and `icmp`

Header Bytes

The purpose of this section is to be able to filter packets based on the contents of a header byte. Consider the following protocols: ARP, Ethernet, ICMP, IP, TCP, and UDP. These are just a few networking protocols we have studied. How can we tell Tcpdump to filter packets based on the contents of protocol header bytes? (We will not go into details about the headers of each protocol as this is beyond the scope of this room; instead, we will focus on TCP flags.)

Using pcap-filter, Tcpdump allows you to refer to the contents of any byte in the header using the following syntax `proto[exprisize]`, where:

- `proto` refers to the protocol. For example, `arp`, `ether`, `icmp`, `ip`, `ip6`, `tcp`, and `udp` refer to ARP, Ethernet, ICMP, IPv4, IPv6, TCP, and UDP respectively.
- `expr` indicates the byte offset, where `0` refers to the first byte.
- `size` indicates the number of bytes that interest us, which can be one, two, or four. It is optional and is one by default.

To better understand this, consider the following two examples from the pcap-filter manual page (and don't worry if you find them difficult):

- `ether[0] & 1 != 0` takes the first byte in the Ethernet header and the decimal number 1 (i.e., `0000 0001` in binary) and applies the `&` (the And binary operation). It will return true if the result is not equal to the number 0 (i.e., `0000 0000`). The purpose of this filter is to show packets sent to a multicast address. A multicast Ethernet address is a particular address that identifies a group of devices intended to receive the same data.
- `ip[0] & 0xf != 5` takes the first byte in the IP header and compares it with the hexadecimal number F (i.e., `0000 1111` in binary). It will return true if the result is not equal to the (decimal) number 5 (i.e., `0000 0101` in binary). The purpose of this filter is to catch all IP packets with options.

Don't worry if you find the above two examples complex. We included them so you know what you can achieve with this; however, fully understanding the above examples is not necessary to finish this task. Instead, we will focus on filtering TCP packets based on the set TCP flags.

You can use `tcp[tcpflags]` to refer to the TCP flags field. The following TCP flags are available to compare with:

- `tcp-syn` TCP SYN (Synchronize)
- `tcp-ack` TCP ACK (Acknowledge)
- `tcp-fin` TCP FIN (Finish)
- `tcp-rst` TCP RST (Reset)
- `tcp-push` TCP Push

Based on the above, we can write:

- `tcpdump "tcp[tcpflags] == tcp-syn"` to capture TCP packets with **only** the SYN (Synchronize) flag set, while all the other flags are unset.
- `tcpdump "tcp[tcpflags] & tcp-syn != 0"` to capture TCP packets with **at least** the SYN (Synchronize) flag set.
- `tcpdump "tcp[tcpflags] & (tcp-syn|tcp-ack) != 0"` to capture TCP packets with **at least** the SYN (Synchronize) **or** ACK (Acknowledge) flags set.

Summary and Examples

The table below provides a summary of the command line options that we covered.

Command	Explanation
`tcpdump -q`	Quick and quite: brief packet information
`tcpdump -e`	Include MAC addresses
`tcpdump -A`	Print packets as ASCII encoding
`tcpdump -xx`	Display packets in hexadecimal format
`tcpdump -X`	Show packets in both hexadecimal and ASCII formats

Resolving IP Addresses

- **According to the following screenshot:**

```
Microsoft Windows [Version 10.0.22635.4515]
(c) Microsoft Corporation. All rights reserved.

C:\Users\20115>ssh mallory@neuseeland.informatik.tu-cottbus.de -p 1335
The authenticity of host '[neuseeland.informatik.tu-cottbus.de]:1335 ([141.43.202.225]:1335)' can't be established.
ED25519 key fingerprint is SHA256:p+iqeHEsLFhLfKdlQpkYbsfFn/pn48OkE0uUde7FWM4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[neuseeland.informatik.tu-cottbus.de]:1335' (ED25519) to the list of known hosts.
mallory@neuseeland.informatik.tu-cottbus.de's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
mallory@4d097f366d6e:~$ pwd
/home/mallory
mallory@4d097f366d6e:~$ ls
mallory@4d097f366d6e:~$ whoami
mallory
mallory@4d097f366d6e:~$
```

- **Bob IP is 10.10.22.3**
- **Alice IP is 10.10.22.2**
- **my IP (mallory) 10.10.22.4**

Ensure good connection:

Summary and Examples

The table below offers a summary of the command line options that we covered.

Command	Explanation
`tcpdump host IP` or `tcpdump host HOSTNAME`	Filters packets by IP address or hostname
`tcpdump src host IP` or	Filters packets by a specific source host
`tcpdump dst host IP`	Filters packets by a specific destination host
`tcpdump port PORT_NUMBER`	Filters packets by port number
`tcpdump src port PORT_NUMBER`	Filters packets by the specified source port number
`tcpdump dst port PORT_NUMBER`	Filters packets by the specified destination port number
`tcpdump PROTOCOL`	Filters packets by protocol; examples include `ip`, `ip6`, and `icmp`

Code explaination

- **Lets examine the given code and understand how it works to be able to successfully use it.**

1. **The main goal of the file is to send a single Ethernet frame to a specified destination address, containing a given protocol and payload.**
2. **it takes 4 arguments:**
   A. **iface: the interface on which we work, in our case it is eth1**
   B. **destination: the target MAC address, in our case (Bob's, and Alice's)**
   C. **payloadProto: in our case it will be ARP**
   D. **payloadFile: the content of the payload, which in our case should be ARP reply.**
3. **we are using sockets, and arp libraries.**
4. **now lets get deeper in the logic:**
   A. **we parse all the arguments and validate them:**
      a. **so argv[1] is the interface, and we check if its length was > IFNAMSIZ we exit with error, as IFNAMSIZ is a constant defined in the <net/if.h> library specifies the maximum length of a network interface name**
      b. **after that we copy that name into our predefined structure, with the IFNAMSIZ length.**
      c. **we do the same for the destination MAC address**
      d. **then we tokenize the MAC address, using the : as splitor, and take each item before it and store it in our dst array**
      e. **then we convert the protocol string into integer using strtol, given these parameters**
         i. **the protocol name**
         ii. **NULL means that there is no need for end-pointer, which is used to know where is the address of the first invalid character in the given protocol.**
         iii. **16 means that the input is in the Hex format**
      f. **then we open the payload file and read its content**
         i. **get its size**
         ii. **ensure it is within the allowed size range**
         iii. **then reading its content into the buffer.**
         iv. **then close the buffer**
      g. **now its time for processing and dealing with the network**
         i. **create a layer2 socket, which deals with the MAC addresses withing the same network.**
         ii. **it takes the following parameters:**
            1. **PF_Packet -> specifies that this is a socket packet for working in the second layer.**
            2. **SOCK_DGRAM -> specifies that the ethernet header will be generated by OS not manually.**
            3. **htons(ETH_P_ALL) -> indicates that socket should capture all protocol types not a specific one.**
         iii. **then we configure the link layer (ll) socket address family and protocol, indicating that it works in the layer 2, and the used protocol is ARP.**
         iv. **then we try to obtain a network interface index, which is a unique value identifies a specific network interface on the network, so we need to get this of eth1 on our system.**
         v. **this is important because when dealing with raw sockets, we need to specify the exact interface**

v. this is important because when dealing with raw sockets, we need to specify the exact interface over which the packet must be send, which is **eth1** in our case.
vi. to do so, we use **ioctl()** which takes the following parameters:
1. **SIOCGIFINDEX:** ioctl operation that retrieves the interface index of the sepecified network device like **eth1**
2. ifreq is interface request previously defined, so we send it by reference to store our result in its **ifr_ifindex** parameter.
vii. now after retrieving the interface request index, we should store it in our link layer interface index.
viii. then define the length of the physical address which we know that MAC address length is 6 bytes.
ix. then we copy the destination address in this field.

h. now its time for sending the message
i. we use sendto method to send the packet over the network, with the following parameters:
1. the socket on which we need to send the data.
2. pointer to the buffer which contains the payload.
3. the length of the payload in bytes.
4. the flags needed, but in our case we need no flags.
5. the link layer, which indicates:
1. the interface address
2. the destination address
3. the protocol type.
6. the length of this link layer.

i. after we finish we just close this socket.

IP Addresses and MACs:

- Alice: 10.0.3.2 - 52:54:00:12:35:02
- Bobs: 10.0.3.3 - 52:54:00:12:35:03
- me: 10.0.3.4 - 52:54:00:12:35:04

Very important note:

- even if the poisoning was occured successfully, no packets will be sniffed until we enable sniffing in our machine, and this occurs using this command:

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
```

- and to avoid noticability, we can send our MAC address in a periodic mannar using a loop ''' while true; do sudo ./raw_packet eth1 00:11:22:33:44:55 0x0806 payloadAlice.bin sudo ./raw_packet eth1 00:11:22:33:44:55 0x0806 payloadBob.bin sleep 1 done '''

Interesting Screen

```
Microsoft Windows [Version 10.0.22635.4515]
(c) Microsoft Corporation. All rights reserved.

C:\Users\20115>ssh mallory@neuseeland.informatik.tu-cottbus.de -p 1335
The authenticity of host '[neuseeland.informatik.tu-cottbus.de]:1335 ([141.43.202.225]:1335)' can't be established.
ED25519 key fingerprint is SHA256:p+iqeHEsLFhLfKdlQpkYbsfFn/pn48OkE0uUde7FWM4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[neuseeland.informatik.tu-cottbus.de]:1335' (ED25519) to the list of known hosts.
mallory@neuseeland.informatik.tu-cottbus.de's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.
```

```
To restore this content, you can run the 'unminimize' command.
mallory@4d097f366d6e:~$ pwd
/home/mallory
mallory@4d097f366d6e:~$ ls
mallory@4d097f366d6e:~$ whoami
mallory
mallory@4d097f366d6e:~$
```

- When we see this screen this indicates that we have successfully poisoned the traffic, and we have the traffic on our device now.
- now its turn for analysis

First Blick:

- In the first blick we can notice many things:
  1. there are some dupplicates, and this should be reasonable, because we are applying spoofing, so the same packet is sent twice on the network to two different machines.
  2. there are some packets with different flags like:
     - [S] -> this is SYN packet
     - [S.] -> this is SYN-ACK packet
     - [.] -> this is ACK packet
     - [P] -> this is PSH related to the data -> means the data is sent, it is usually used in protocols like HTTP.
     - [F] -> this is FIN packet to end the communincation.
     - [R] -> this is reset packet to reset the communication

Now What?

- all what we need to do is to use tcpdump and allow it to show the output in the ASCII format using this flag -A

```
Microsoft Windows [Version 10.0.22635.4515]
(c) Microsoft Corporation. All rights reserved.

C:\Users\20115>ssh mallory@neuseeland.informatik.tu-cottbus.de -p 1335
The authenticity of host '[neuseeland.informatik.tu-cottbus.de]:1335 ([141.43.202.225]:1335)' can't be established.
ED25519 key fingerprint is SHA256:p+iqeHEsLFhLfKdlQpkYbsfFn/pn48OkE0uUde7FWM4.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[neuseeland.informatik.tu-cottbus.de]:1335' (ED25519) to the list of known hosts.
mallory@neuseeland.informatik.tu-cottbus.de's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-124-generic x86_64)

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

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
mallory@4d097f366d6e:~$ pwd
/home/mallory
mallory@4d097f366d6e:~$ ls
mallory@4d097f366d6e:~$ whoami
mallory
mallory@4d097f366d6e:~$
```

Summary for all commands:

1. Create an ARP Reply payload files, one for BOB, and the other for Alice to poison their caches following this structure:
   - hardware type 2bytes -> 0001 eth1
   - protocol type 2bytes -> 0800 ipv4
   - MAC add length 1 byte -> 0006 mac is 6 bytes
   - Protocol address length -> 0004 ipv4 is 4 bytes

- operation 2bytes -> ARP Reply
- sender MAC -> MAC of the Attacker
- sender IP -> the spoofed IP (like if we want to lie to Alice put Bob's IP)
- Target MAC -> Alice MAC
- Target IP -> Alice IP

2. do the exact same file for bob
3. convert these file to binary using xxd with .bin extensions.
4. create a file and write inside it the following script to apply stealth script:

> touch script.sh vim script.sh
>
> ```
> while true; do
>  sudo raw_packet eth1 02:42:0a:0a:16:02 0x0806 payLoadAlice.bin
>  sudo raw_packet eth1 02:42:0a:0a:16:03 0x0806 payLoadBob.bin
>  sleep 1
>  done
> ```
>
> :wq!

5. make it excutable

> chmod +x script.sh

6. run it either in background & or in another terminal

> ./script.sh &

7. allow packet forwarding on your machine

> echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward

8. execute tcpdump on interface eth1 to listen to the sent data and locate the line using grep

> sudo tcpdump -i eth1 -n -A | grep "CTF"

> Congrats you got it :)

In [ ]: