CSE 406
Computer Security Sessional

Design Report

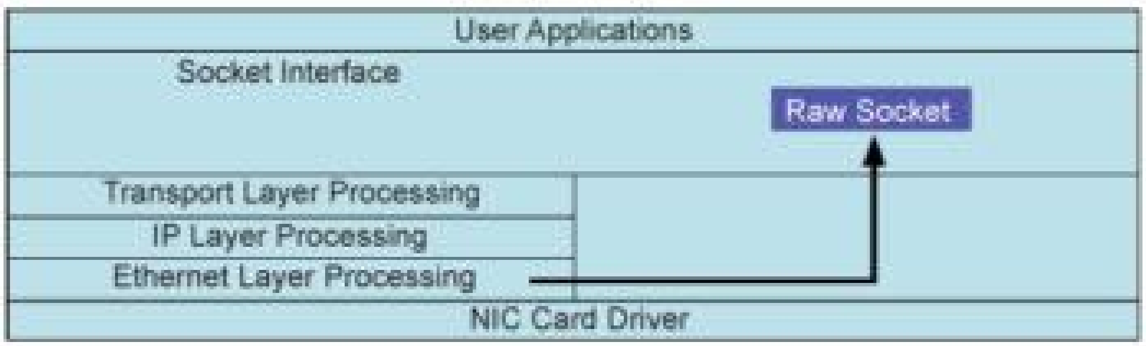
Group - 09

Attack tools and responsible students:

Topic	Attack tool	Responsible student	Student ID
2	Packet sniffing attack and sniff HTTP passwords	Nafiur Rahman Khadem	1605045
10	ICMP ping spoofing + ICMP redirect attack	Ahsanul Ameen Sabit	1605047
15	TCP reset attack on video streaming	Washief Hossain Mugdho	1605058

Packet sniffing attack and sniff HTTP passwords

The attacker will connect to the same LAN as the victim. Machines are connected to networks through Network Interface Cards (NIC). Every packet in a LAN is sent to the NIC and then the NIC drops the packets which are not meant for this machine. Using promiscuous mode, the attacker will actually sniff all the packets instead of only keeping packets meant for him. Using this, the attacker will see the HTTP requests of the victim and possibly sniff passwords.



Using pcap API and raw socket, the attack tool will receive raw Ethernet frames directly from NIC, bypassing the normal TCP/IP processing of normal sockets. Using promiscuous mode and raw socket, the attack tool will receive all data flowing through the LAN, regardless of the destination IP address or port number.

To check the HTTP payload, the attack tool will separate the ethernet header, IP header, and TCP header from the ethernet frame, the remaining bytes will be the payload and may contain HTTP request data. In the case of an HTTP request, the IP header's Protocol field will indicate TCP and the TCP header's destination_port field will be 80.

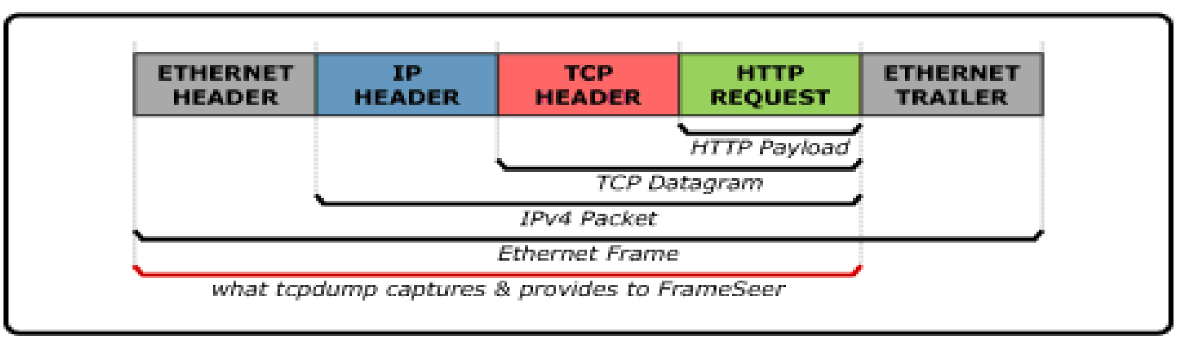


Figure 2 — Ethernet Frame Format

Data flow: HTTP request from victim --------------> HTTP server
└------> attacker

Sample usage: If the victim sends a request from a machine with IP 10.0.2.13:
curl -d "user=user1&pass=abcd" -X POST www.google.com -so /dev/null,
The attack tool running from attacker machine will display:

From: 10.0.2.13
To: 172.217.166.100
Protocol: HTTP
payload size 168
**********************Data Payload************************
POST / HTTP/1.1
Host: www.google.com
User-Agent: curl/7.64.0
Accept: */*
Content-Length: 20
Content-Type: application/x-www-form-urlencoded
user=user1&pass=abcd

Justification

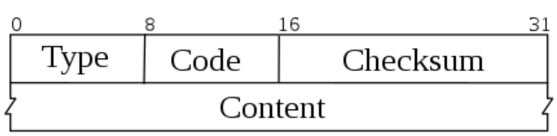
This attack should work, because the attacker is at the same LAN as the victim and the attacker has enabled promiscuous mode.

Preventions

Passwords should be encrypted before sending to the server. HTTPS should be used instead of HTTP. Using https, the traffic is encrypted as soon as it leaves the application layer. SSH should be used instead of telnet.

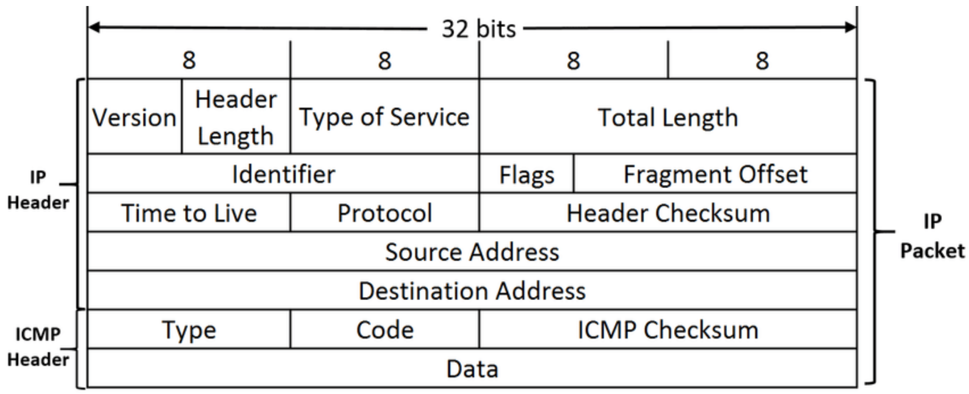
ICMP ping spoofing + ICMP redirect attack

The **Internet Control Message Protocol** (**ICMP**) is a supporting protocol in the Internet protocol suite. It is used by network devices, including routers, to send error messages and operational information indicating success or failure when communicating with another IP address.
An ICMP header has a structure like this.



In *spoofing* attacks, attackers can send out packets under a *false identity*. For example, attackers can send out packets that claim to be from another computer. For packet spoofing, we show how to use **raw sockets** to send *spoofed* IP packets, with their header fields filled with arbitrary values. When using typical socket programming to send out packets, we only have controls over a few selected fields in the header. We can choose the destination IP address but **not** the source IP address.

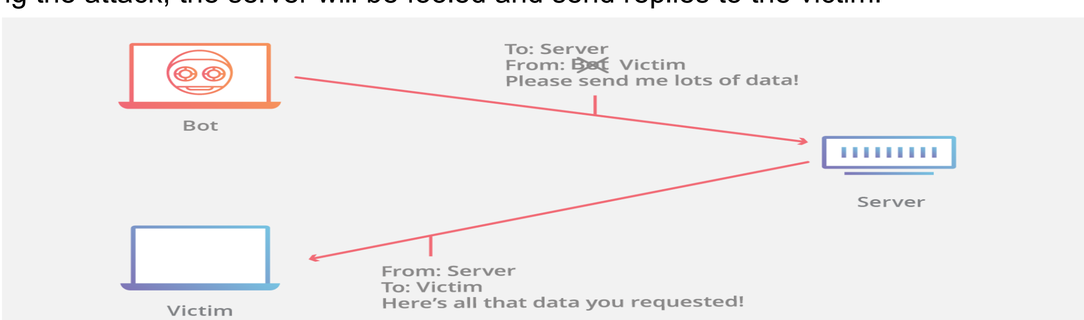
In an ICMP ping spoofing attack, the attacker may change the content of the ICMP echo request header accordingly to fool the server by changing the source IP field. As we can see here.



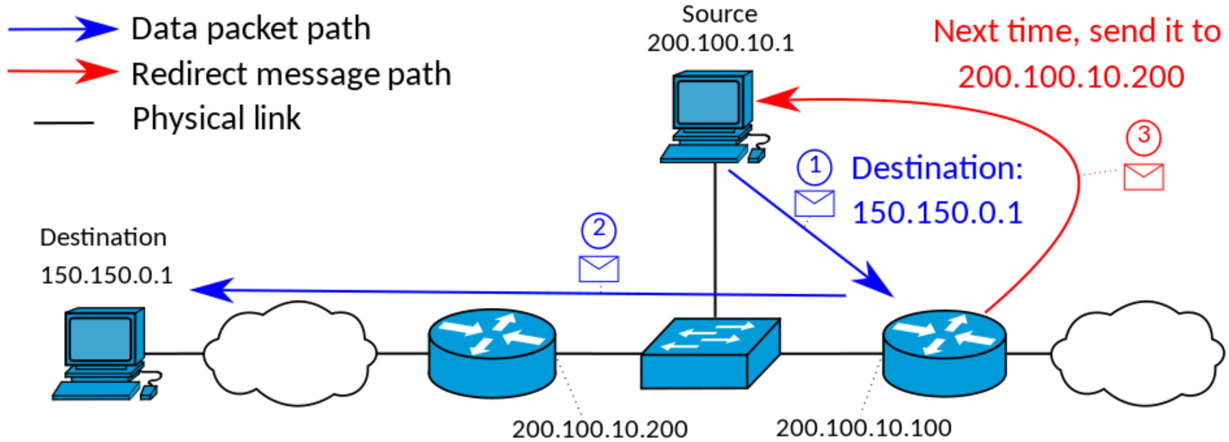
One can set the type of packet as follows.

Type 8 – Echo Request	Echo request (used to **ping**)	ICMP Type 3: Destination Unreachable Codes
Type 5 – Redirect	Error message sent to the sender	0 - Net is unreachable
Type 0 – Echo Reply	Echo reply (used to **ping**)	1 - Host is unreachable

During the attack, the server will be fooled and send replies to the victim.



An **ICMP redirect** is an error message sent by a router to the sender of an IP packet. Redirects are used when a router believes a packet is being routed incorrectly or inefficiently, and it would like to inform the sender that it should use a different router for the subsequent packets sent to that same destination. When an ICMP redirect is exploited by attackers the scenario may look like this.



-> A victim may run *ping 8.8.8.8*
-> Attacker Sniff and send ICMP4/6 redirect

Using raw packets we can request the operating system to leave us alone and change the source IP with the victim's IP.

Tool description 01: Spoof an ICMP echo request using an arbitrary source IP Address

- Step 1: Fill in the ICMP header.
- Step 2: Fill in the IP header
- Step 3: Finally, send the spoofed packet

Given an IP packet, we can send it out using a raw socket after calculating the Internet Checksum. Sample IP and ICMP header fields are shown here.

ICMP Header				
ICMP message type	Error code	Checksum for ICMP Header and data	ID for identifying request	Sequence number

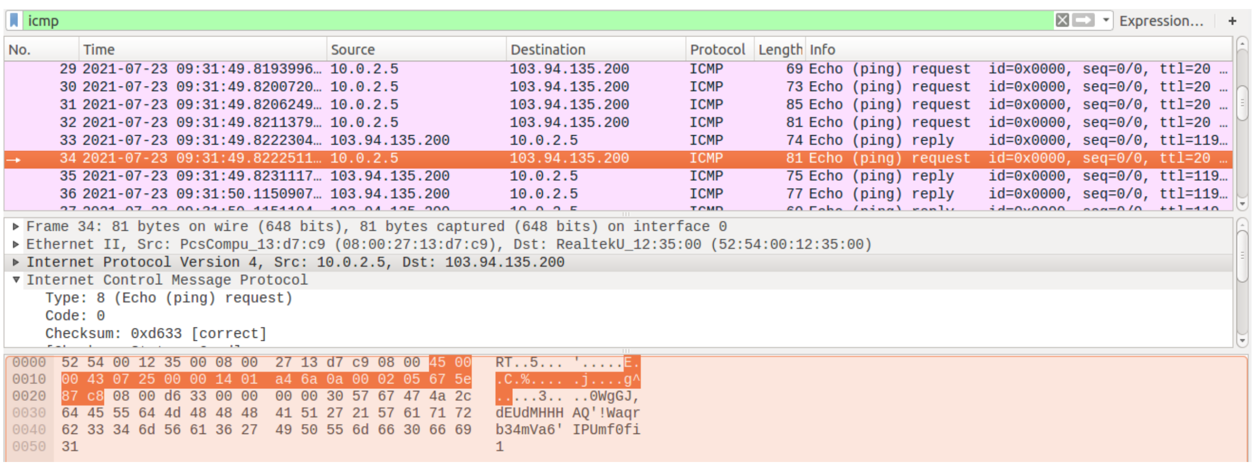
IP Header				
IP header length + version	Type of service	IP Packet length (data + header)	Identification	Fragmentation flags
Flags offset	Time to live	Protocol type	IP datagram checksum	Source IP address
Destination IP	*We create such headers using an array-like data structure and update its*			

address	*contents manually in the code. Random payloads are sent with packets*

We have to run the attack with root privilege (or with the “sudo” command) in our **SEED Ubuntu 16.04 VM (32-bit)** machine. Here the attacker (Clone 02 with IP <*10.0.2.6*>) chooses a spoofed source IP <*10.0.2.5*> (Clone 01) and sends 15 ICMP Echo requests to destination IP <*103.94.135.200*>. One can also sniff those packets to observe their contents.

```
[07/23/21]seed@VM:~/.../ICMP_ping_spoofing$ sudo ./ping.sh 10.0.2.5 103.94.135.200 15
# ---- packet[1] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103.94.135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[2] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103.94.135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[3] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103.94.135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[4] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103.94.135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[5] ----
```

Using WireShark in the same machine we can observe the ICMP packets are flooded with the above-mentioned source and destination IP.



Also, we can observe the actual scenario using a packet sniffer program written in C/C++.

```
        To: <103.94.135.200>
        Type: ping request
        Payload: ?vaNxQIAZh6mj4p!x
1aLqEywLHQJcU?bMNO9mft

 Protocol: ICMP
        From: <10.0.2.5>
        To: <103.94.135.200>
        Type: ping request
        Payload: 0WgGJ,dEUdMHHHAQ'
!Waqrb34mVa6'IPUmf0fil

 Protocol: ICMP
        From: <103.94.135.200>
        To: <10.0.2.5>
        Type: ping reply
        Payload: 5ugc76zxB708bccr8
CGKI4nBUjCKP7Ib
```

```
135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[14] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103
135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
# ---- packet[15] ----
        # --->> Sending a ICMP echo request packet to dest_ip <103
135.200>....
        # --->> Spoofed packet with source_ip <10.0.2.5> sent....
[07/23/21]seed@VM:~/.../ICMP_ping_spoofing$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:13:d7:c9
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::895:d503:db3b:ae52/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10452 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6171 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
```

Tool description 02: ICMP redirect attack.

- Our program takes *<default_gateway's_ip>, <victim's_ip>, <attacker's_ip/new_gateway's_ip>* and possibly MAC addresses of these machines.
- It will work on a particular interface such as ***wlp3s0 / enp0s3*** with a particular *<source_ip>* & *<destination_ip>*
- While the victim does a ping/traceroute (or other requests), it will capture and send an ICMP redirect message advertising a vulnerable gateway(router) controlled by the attacker.

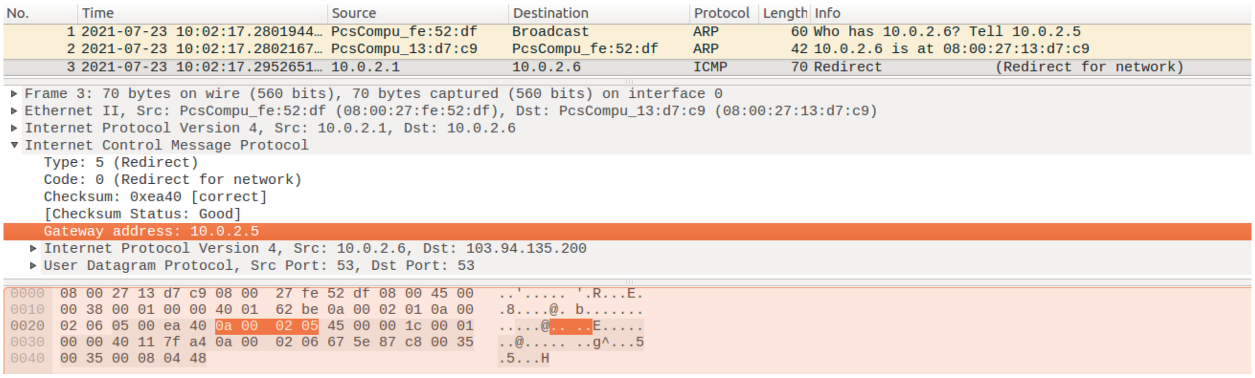
Attack strategy: Here we've Host A (a seedUbuntu clone) with IP *<10.0.2.6>* as Victim, Host M (another clone) with IP *<10.0.2.5>* as attacker/new gateway and provided default gateway is <10.0.2.1> within enp0s3 interface. We're considering *<103.94.135.200>* as destination ip.
We can check out the current gateway with the command *mtr -n 103.94.135.200*

```
                             My traceroute  [v0.86]
VM (0.0.0.0)                                         Fri Jul 23 09:44:22 2021
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                      Packets               Pings
 Host                               Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 10.0.2.1                         0.0%    33    4.5   1.1   0.6   4.5   0.7
 2. 192.168.0.1                      0.0%    33    6.7  12.0   2.9 146.3  26.9
 3. 10.20.6.1                        0.0%    33    3.5   9.5   3.3  50.8  10.8
 4. 10.20.40.2                       0.0%    32    4.5  10.4   3.3 105.5  18.1
 5. 45.118.247.37                    0.0%    32    3.9   8.3   3.1  37.1   8.0
 6. 103.125.54.90                    0.0%    32    4.7  13.3   3.7 104.9  19.9
 7. 100.100.0.14                     0.0%    32    6.7  16.1   4.8 113.8  24.6
 8. 163.47.36.130                    0.0%    32   11.7  10.7   4.7  46.5   7.6
 9. 163.47.36.130                    0.0%    32    4.9  11.2   4.2  56.7  11.7
10. 103.94.135.200                   0.0%    32    9.4  23.2   4.1 350.3  62.5
```

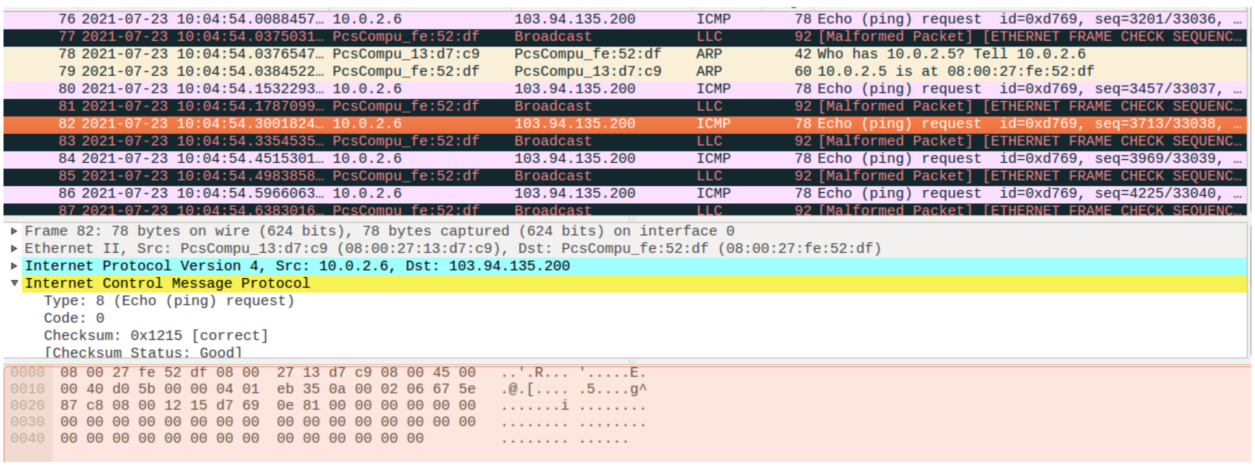
Before launching an ICMP redirect attack we've to pause the default protection mechanism in the victim's machine with the command: *sudo sysctl net.ipv4.conf.all.accept_redirects=1*.

```
[07/23/21]seed@VM:~$ sudo sysctl net.ipv4.conf.all.accept_redirects=1
net.ipv4.conf.all.accept_redirects = 1
[07/23/21]seed@VM:~$
```

After executing the **attack.sh** script, we'll see that a redirect message has been sent to the victim from the current gateway. It tells the target to use a different route (here <10.0.2.5>) as a new gateway.



Now if we run *mtr -n 103.94.135.200* again on the victim's pc, we'll observe that it is kind of blocking, and using Wireshark we observe something like this…



But running *mtr -n <any other valid IP>* will work fine as well as they will be redirected to default gateway.

We've used scapy as a tool. So our redirect program is written in python. But we've run the script using C++.

Preventions

For end-users, detecting IP spoofing is virtually impossible. They can minimize the risk of other types of spoofing, however, by using secure encryption protocols like HTTPS - and only surfing sites that also use them. Also using IPv6 may produce a good result. Looking for spoofed packets that do not originate from within your network, also known as egress filtering. Also, MAC-filtering IP-MAC pair filtering can be helpful to identify vulnerable hosts. Packets can be dropped if a matching route entry is not available.
Setting the ignore property to 1 and verifying the current value may help prevent ICMP redirects. Because ICMP redirect messages to modify the host's route table are unauthenticated. Also preventing sending redirection messages also helps.

TCP reset attack on video streaming:

An IP-TCP header looks like this-

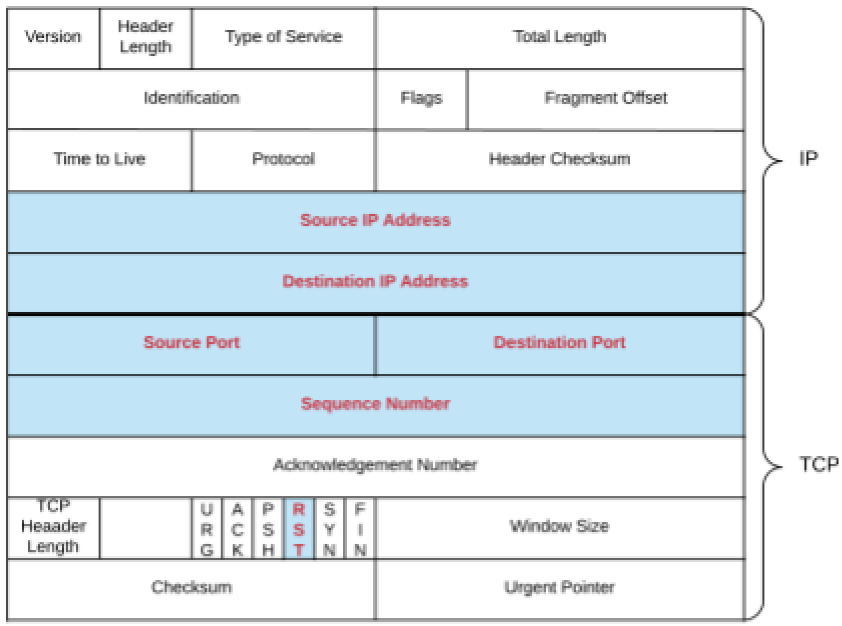


Fig: IP and TCP header (combined)

We can see that there is a reset (RST) bit in the TCP header. The purpose of the bit is to abort the connection in extreme conditions. Such as network failure, power failure, etc.

Let, two TCP endpoints A & B.

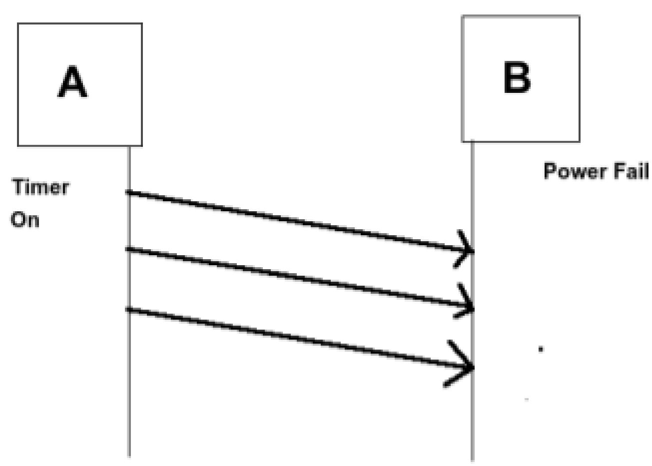


Fig: TCP timer diagram

When a TCP endpoint (B) faces network/power failure it has no way to communicate. But the other endpoint (A) optimistically waits for data or acknowledgement. So, it keeps its timer open and resends data periodically. Which is not a desired property. When B wakes up, it sends an RST signal to reset this TCP connection. So, sending RST is another way of disconnecting than a normal four-way handshake. An attacker can take advantage of this RST signal. He can send fake RST signals to disconnect TCP connections. This kind of attack is known as “TCP Reset Attack”.

Tool Description:

As our attack tool is on a “video streaming server”, we will now demonstrate procedures regarding our scenario. The procedures follow as below:

1) First we have to know the video server's IP address and port number. We have a virtual server machine (Ubuntu 16.04) running in our VirtualBox. Its IP address is 192.168.0.104. We used VLC Media Player to stream at port 8080.
2) Then we sniff the packets in network traffic. In promiscuous mode, we used pcap API and raw socket to receive raw Ethernet frames directly from the network interface card (NIC). As our tool runs in promiscuous mode, it bypasses the normal TCP/IP processing.
3) We extracted packets with desired source IP and port number.
4) Then we construct a spoof for each of these packets. The headers were set accordingly. *Some important fields* of the headers are described below.

Field Name	Value
IP Header	
Source IP	Original Header’s Destination IP
Destination IP	Original Header’s Source IP
Protocol	TCP (6)
Checksum	Calculated Checksum
TCP Header	
Source Port	Original Header’s Destination Port
Destination Port	Original Header’s Source Port
Sequence Number	Original Headers’ Acknowledgement Sequence Number
Acknowledgement Sequence Number	Original Headers’ Sequence Number + Payload Size In Bytes
Reset Bit	**1**
Checksum	Calculated Checksum

5) Then we send the constructed packet by a raw socket.
6) If we inspect packets using Wireshark, we can see the packets have been successfully sent.

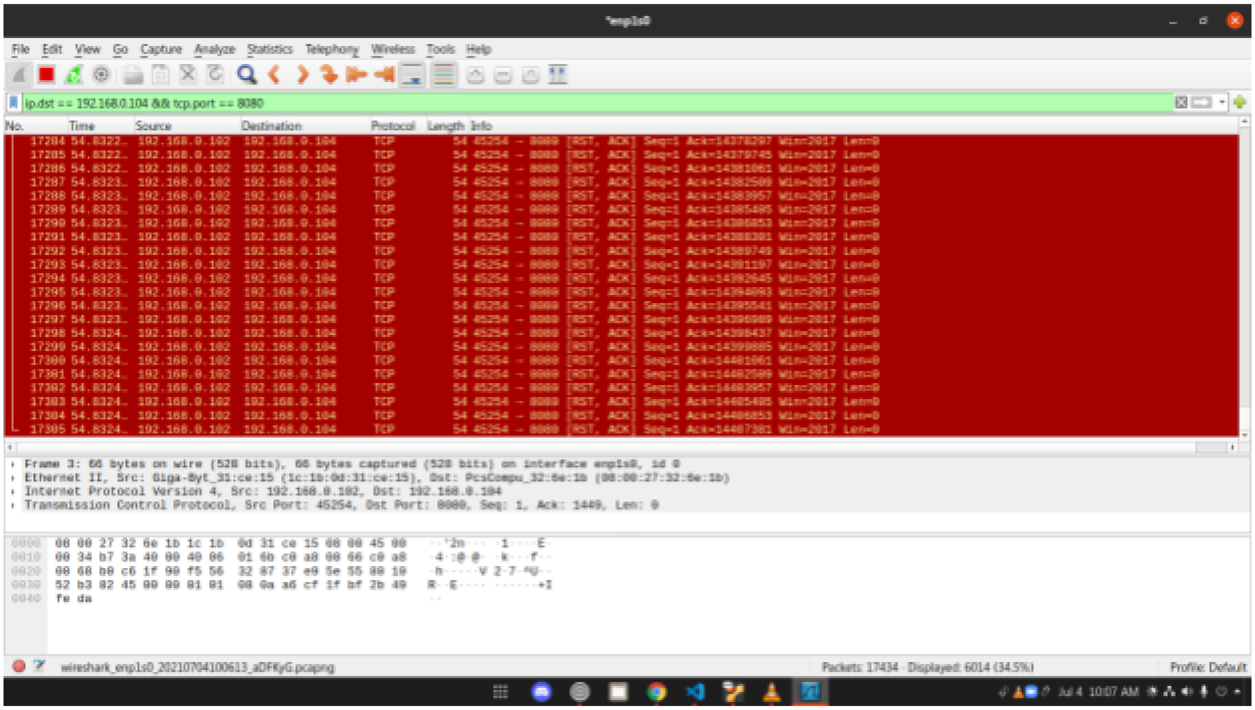


Fig: Inspection in Wireshark

7) We can also notice that video streaming stops in the client machine. So, our attack is successful.

Cautions:

In video streaming scenarios, many packets are delivered very quickly. If our program is slow, the server will receive the client's legit acknowledgement before the spoofed message. Attack will fail in that case. To prevent that, we avoided printing anything on the console. Also, our tool is written in the C programming language. As a result, our program is very fast. So, our tool can conduct the attack successfully.

Prevention:

TCP reset(RST) signal is only meant for extreme scenarios. It is very unlikely for a server to receive a reset signal with the same latency as normal frames. We can suspect that these kinds of reset signals are not legit. A server may discard these reset signals. As a result, the attack will fail.