CSE 406 Report
TCP Reset Attack
On Video Streaming Applications

Course: CSE 406
Lab Group: B1

Submitted By:
Tanjim Bin Faruk
(1505082)

Submitted To:
Dr. Md. Shohrab Hossain

September 8, 2019

Department Of Computer Science and Engineering
Bangladesh University of Engineering and Technology
(BUET)

Contents

1 Introduction

1.1 What is TCP?

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network.

TCP may be based on a connectionless network layer protocol (such as IP), but still achieves in-order delivery of a byte-stream, by means of segment sequence numbering on the sender side, packet buffering and data packet reordering on the receiver side. The sequence numbering requires two-way synchronization of segment counters during a three-step connection establishment phase.

1.2 TCP Connection Establishment

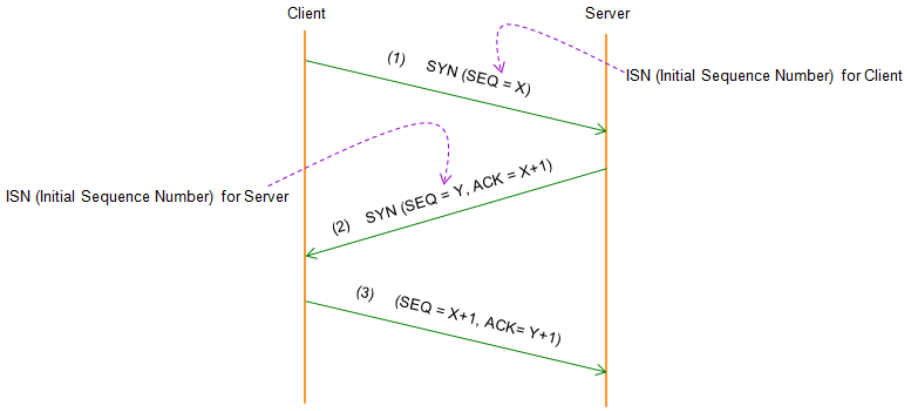


Figure 1: TCP 3 Way Handshake

TCP uses a 3 way handshake to establish an end-to-end full duplex connection. The Initial Sequence Number (ISN) is chosen at random for each host (with 2^{32} possible combinations). As it is a connection-oriented protocol, there is a retransmission timeout for each type of message (e.g: ***SYN***, ***ACK***, ***FIN***) which ensures reliable delivery even in the face of message corruption or message getting lost.

1.3 TCP Connection Termination

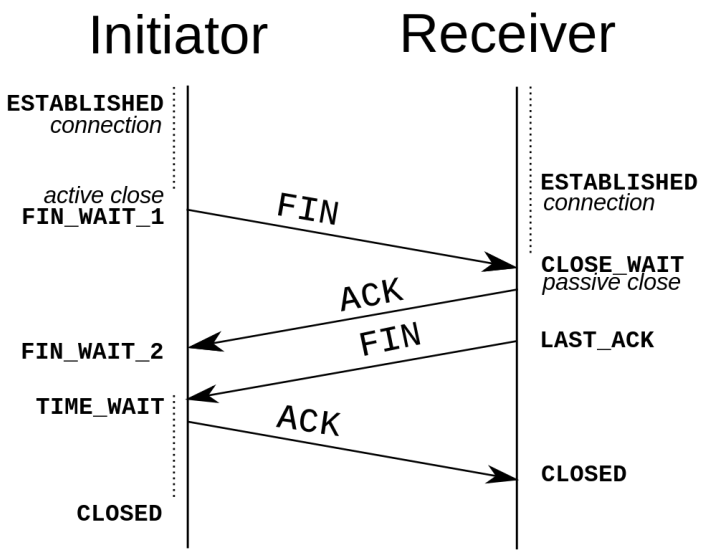


Figure 2: TCP Connection Termination

The connection termination phase uses a four-way handshake, with each side of the connection terminating independently.

When an endpoint wishes to stop its half of the connection, it transmits a ***FIN*** packet, which the other end acknowledges with an ***ACK***. Therefore, a typical tear-down requires a pair of ***FIN*** and ***ACK*** segments from each TCP endpoint. After the side that sent the first ***FIN*** has responded with the final ***ACK***, it waits for a timeout before finally closing the connection, during which time the local port is unavailable for new connections; this prevents confusion due to delayed packets being delivered during subsequent connections.

2 What is TCP Reset Attack?

TCP reset attack, also known as "forged TCP resets" or "spoofed TCP reset packets", is a way to tamper and terminate the Internet connection by sending a forged TCP reset packet. ***RST*** is a flag in TCP packets to indicate that the connection is no longer working. So, if any of the two participants in a TCP connection send a packet contains such a ***RST*** flag, the connection will be closed immediately. This tampering technique can be used by a firewall in goodwill, or abused by a malicious attacker to interrupt Internet connections.

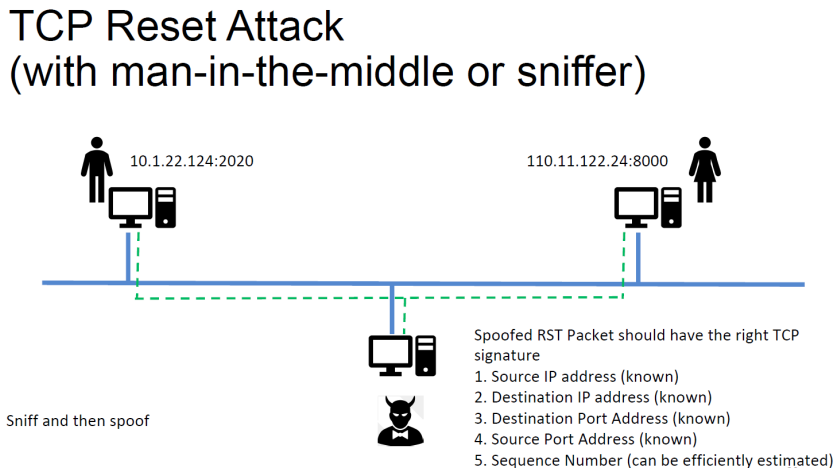


Figure 3: TCP Reset Attack

There are mainly 2 types of variation of the TCP RST attack:

- RST attack on Telnet/SSH
- RST attack on video streaming applications

In this report, we will be looking at TCP RST attack on video streaming applications in details.

2.1 Difference betwen the types of TCP RST Attack

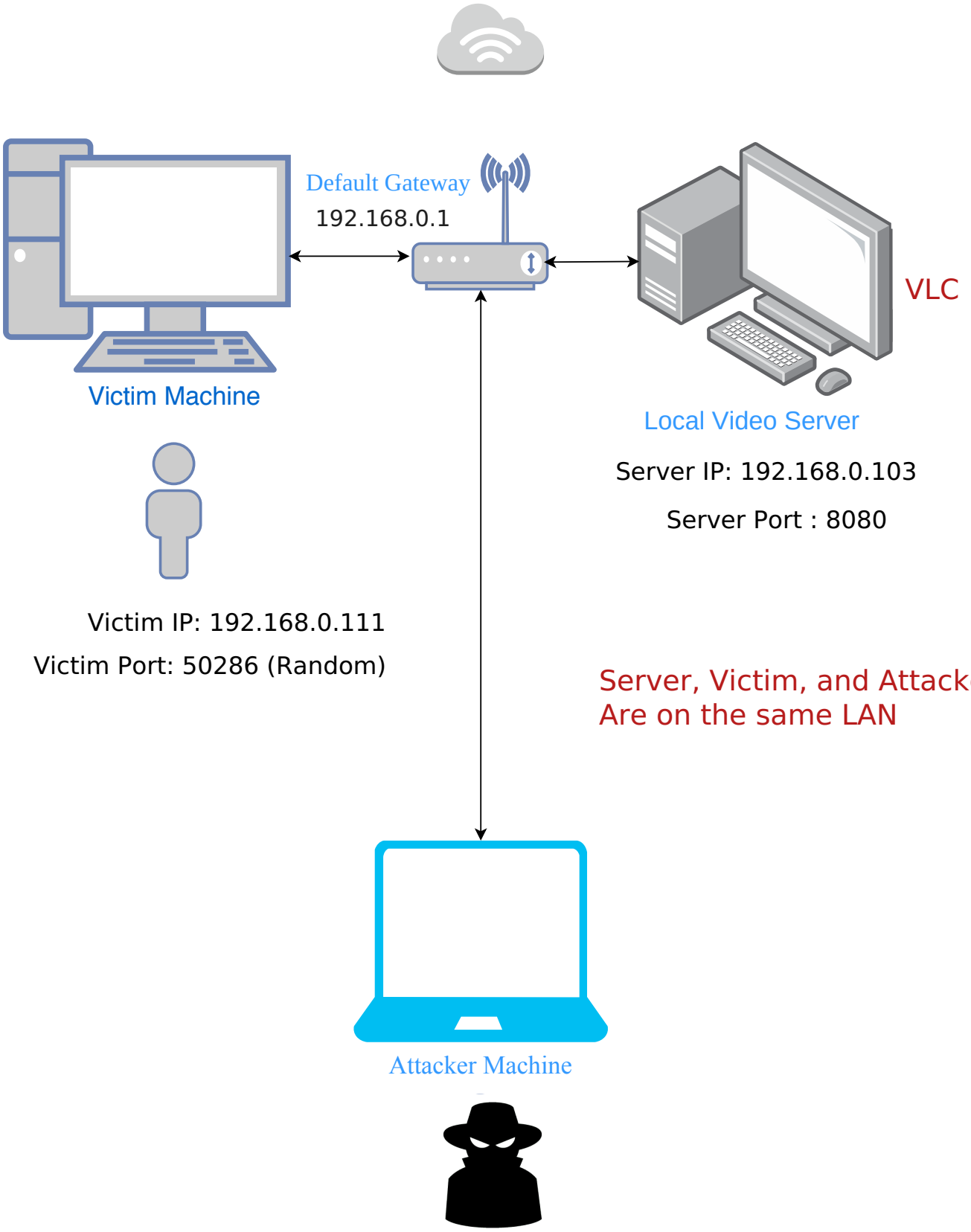
There are some differences between the challenges faced in implementing these two attacks.

In the case of Telnet/SSH, the victim establishes a connection with another host/server and browse their directory. As it requires the victim typing commands in the command line interface, the sequence number does not vary much. We can simply look up the sequence number using a tool like ***Wireshark***.

But in the case of video streaming applications, the scenario is a bit complicated. The victim just connects to a video server though a TCP connection and afterwards, no more command typing is needed. As it is a stream, the sequence number can increase very quickly, and it is not possible to find out the correct sequence number using mechanical effort (i.e: going through the Wireshark information). Some kind of automation is needed here.

3 Network Topology

The network topology is pretty straightforward. The victim, the attacker, and the server - all are on the same LAN. Server will stream the video, and victim will connect to the server to watch that video. Attacker will perform man in the middle attack between this connection.


Default Gateway
192.168.0.1
VLC
Victim Machine
Local Video Server
Server IP: 192.168.0.103
Server Port : 8080
Victim IP: 192.168.0.111
Victim Port: 50286 (Random)
Server, Victim, and Attacker
Are on the same LAN
Attacker Machine
Attacker IP: 192.168.0.112
Attacker Port: 8080

4 Attack Strategies

4.1 Packet Construction

The victim will first establish a TCP connection with the local video server when it wishes to watch video. After the connection has been established, the server machine will stream the video, and the victim will be able to watch the video normally. The attacker will first sniff the packets transferred between the video server and the victim machine in order to gain information about the IP addresses, Port numbers and Sequence numbers. There has to be some sort of automation involved, as in this case the sequence numbers can rise very quickly. After acquiring the the information, the attacker will carefully craft a TCP packet with the correct source port, destination port, sequence number and the set ***RST*** bit field. The the attacker will encapsulate it with an IP datagram after putting the correct source IP and destination IP. The relevant fields is visible from the figures below. Refer to RFC 791 and RFC 793 for more details.

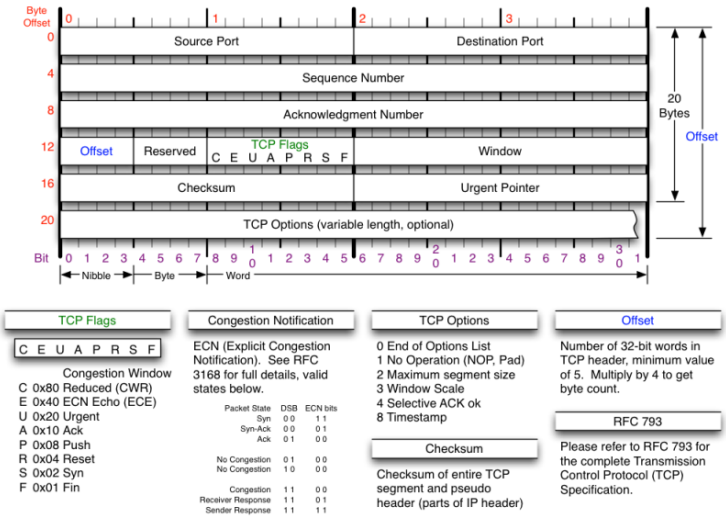
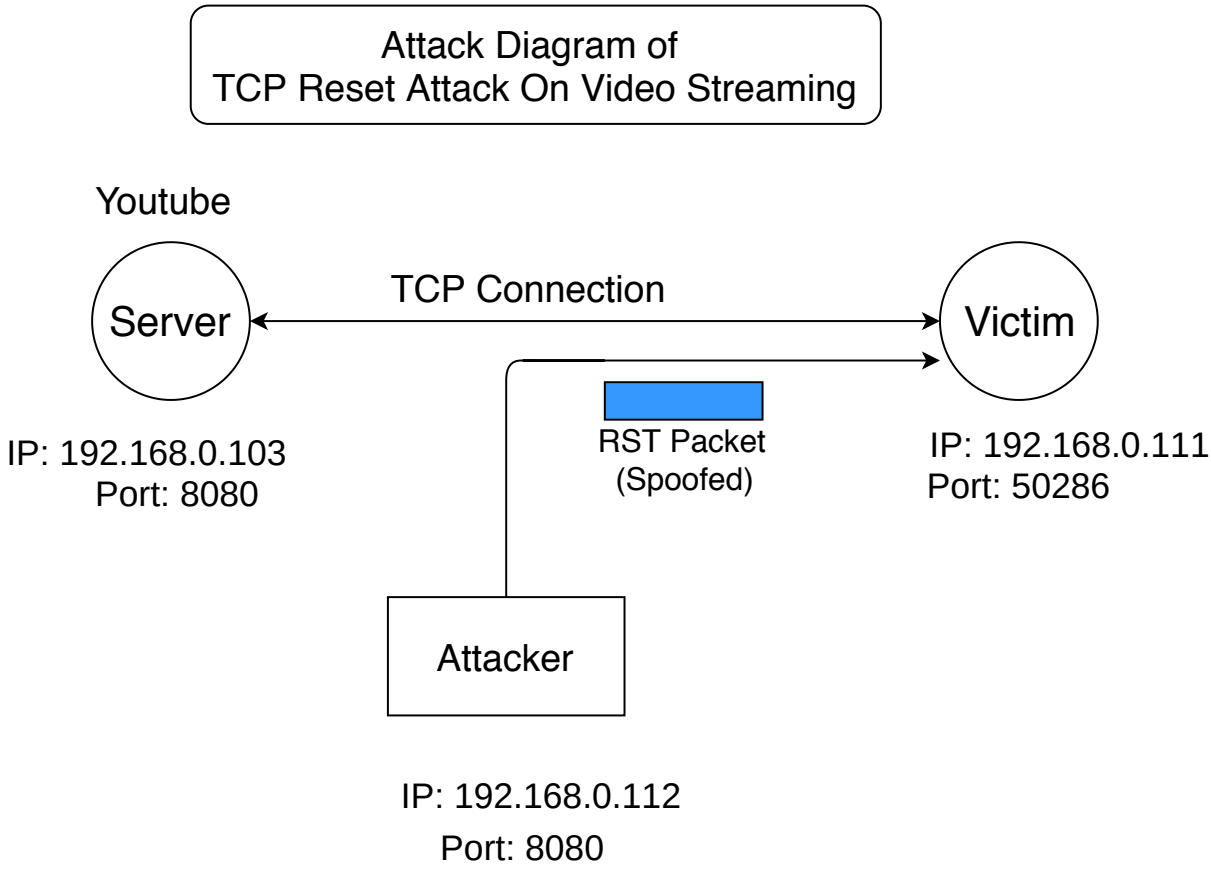


Figure 4: TCP Header

4.2 Attack Tools

We will use the ***pcap*** library and raw ***C*** socket programming for this attack. TCP reset attack needs to use the correct sequence number. When there are a lot of traffics, if the attack program does not send out the spoofed reset packet in time, the sequence number it chooses to use may have already been consumed by other packets, so the reset packet will be discarded by the receiver. Using ***C*** ensures fast transmission of the spoofed ***RST*** packet.


Attack Diagram of
TCP Reset Attack On Video Streaming
Youtube
Server
TCP Connection
Victim
IP: 192.168.0.103
Port: 8080
RST Packet
(Spoofed)
IP: 192.168.0.111
Port: 50286
Attacker
IP: 192.168.0.112
Port: 8080

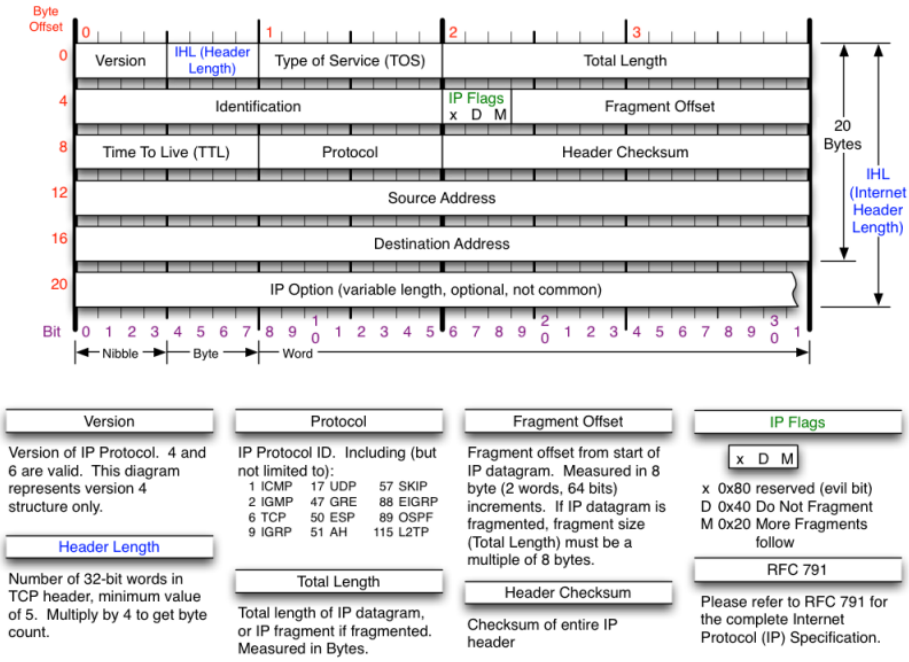


Figure 5: IP Header

5 Implementation

5.1 Environment Setup

To carry out the attack, we first need to setup the attack environment. As we've said earlier, the three entities will be on the same LAN. The server, and the attacker will be connected to a switch (A layer 2 device) via an Ethernet cable. Victim maybe connected to either Wi-Fi or to the switch via an Ethernet cable.

5.2 Sniffing

On a LAN, after the stream has started, server and victim will be able to communicate with each other because the TCP connection has taken place. Attacker will not be able to listen that communication because the packets are not intended for him/her. In order to sniff the packets, attacker will need to carry out an ARP spoofing attack.

5.2.1 ARP Spoofing Man In the Middle Attack

ARP spoofing attack will update the mac table of both the server and the victim. Both the server and the victim will send their packets to the attacker.

So the attacker needs to turn on ***IP forwarding***. On a linux based system, we can turn on IP forwarding by the following command:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

IP forwarding ensures that when the server and the victim send packet to the attacker, the attacker will redirect the packets to their originally intended destination. ARP Spoofing helps to capture the packets between the two parties, i.e., to sniff them in order to extract meaningful information.

In order to carry out the ARP Spoofing attack, we run a python program ***MITM.py*** on attacker's machine which will continuously send out ARP packets to both server and victim. In our case, the attacker (192.168.0.112) claims to be both the server (192.168.0.103) and the victim (192.168.0.111). So the server will think the attacker as the victim and send the packets intended for the victim to the attacker. The victim will do the same. So, all the packets will pass through the network interface of the attacker.

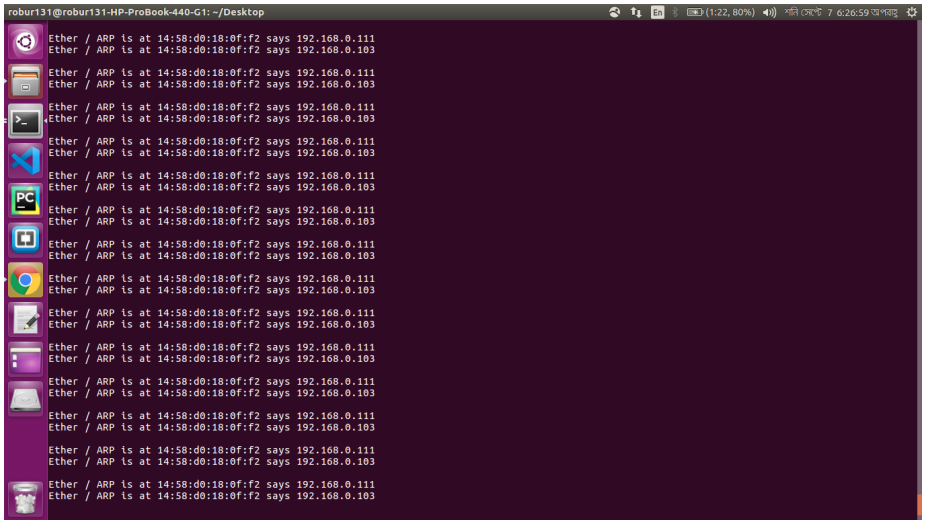


Figure 6: Attacker running ARP Spoof Attacl

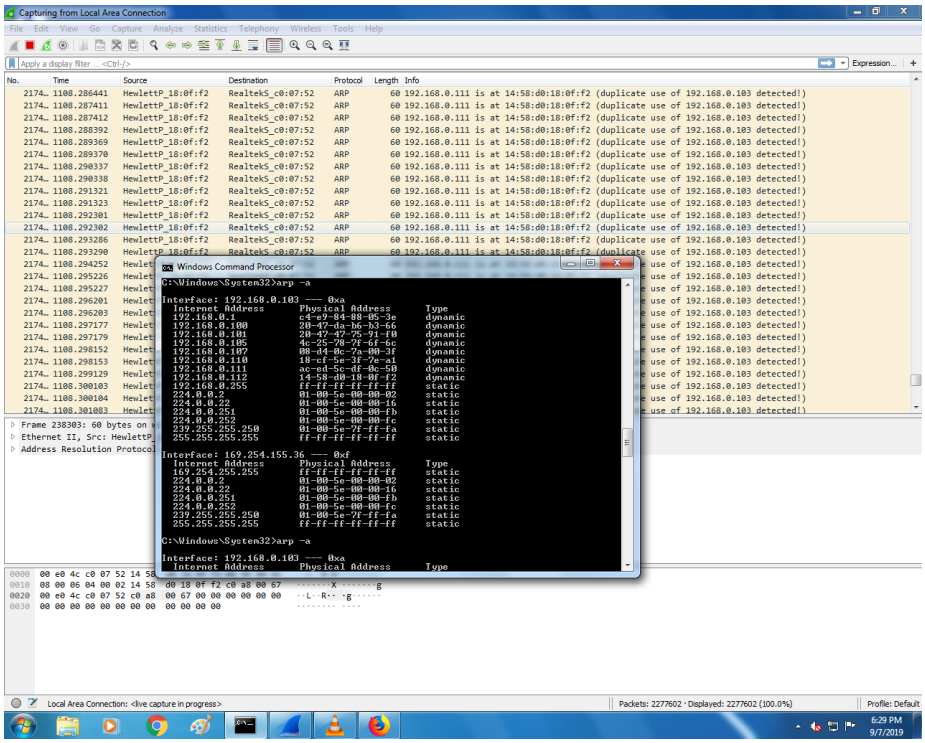


Figure 7: Correct Server MAC Table Before ARP Spoofing Attack

The mac address table can be looked up by the following command, both on windows and linux based system:

```
arp -a
```

From the figure, we can see the initial mac table of the server which contains the correct IP to MAC address translation. On the background, we can the numerous ARP Packet transmission on Wireshark as a result of running the python program.

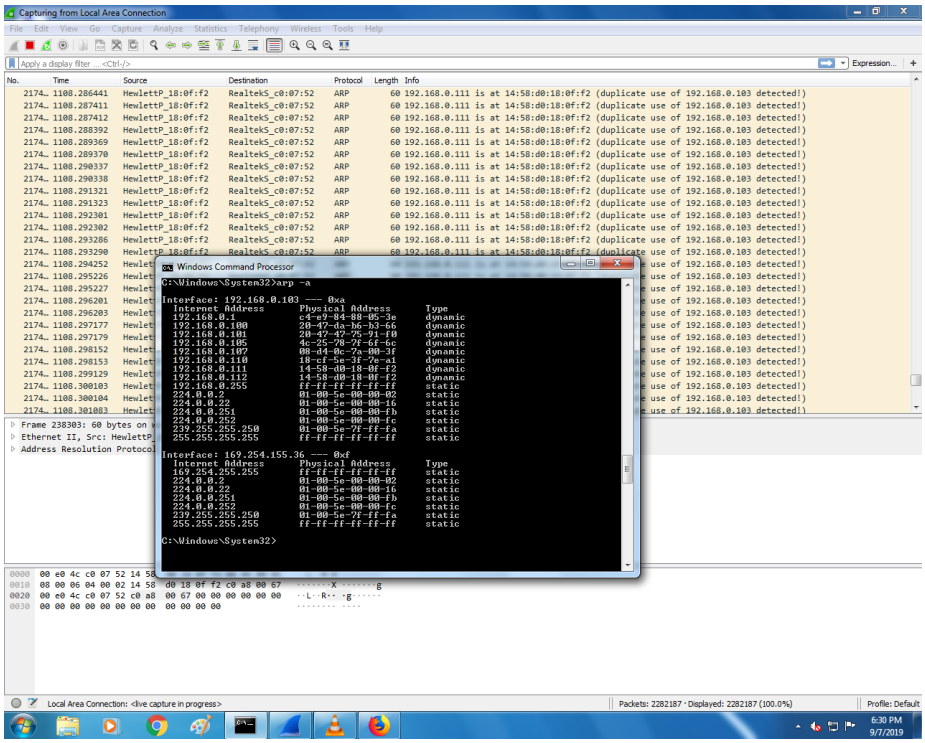


Figure 8: Incorrect Server MAC Table After ARP Spoofing Attack

After running the program, we can see from the figure that the server mac table changed and contains incorrect translation. The attacker's MAC address has replaced victim's MAC address. The attacker has successfully carried out a man in the middle attack. Likewise, we find out the same from victim's incorrect mac table:

```
nafis@nafis-X580VD:~$ ifconfig
enp4s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 10:7b:44:38:f4:53  txqueuelen 1000  (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

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2392  bytes 227116 (227.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2392  bytes 227116 (227.1 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlp3s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.111  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::411:d3b6:b8e2:826b  prefixlen 64  scopeid 0x20<link>
        ether ac:ed:5c:df:0c:50  txqueuelen 1000  (Ethernet)
        RX packets 2887596  bytes 1713040223 (1.7 GB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 409944  bytes 45131644 (45.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

nafis@nafis-X580VD:~$ arp -a
? (192.168.0.103) at 00:e0:4c:c0:07:52 [ether] on wlp3s0
? (192.168.0.112) at 14:58:d0:18:0f:f2 [ether] on wlp3s0
_gateway (192.168.0.1) at c4:e9:84:88:05:3e [ether] on wlp3s0
nafis@nafis-X580VD:~$
```

Figure 9: Incorrect Victim MAC Table After ARP Spoofing Attack

```
robur131@robur131-HP-ProBook-440-G1:~/Desktop$ ifconfig
enp2s0    Link encap:Ethernet  HWaddr 14:58:d0:18:0f:f2
          inet addr:192.168.0.112  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::aa59:9dfb:cd66:9ed7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31995 errors:0 dropped:1 overruns:0 frame:0
          TX packets:6615006 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:23298722 (23.2 MB)  TX bytes:323646450 (323.6 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:10229 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10229 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:769821 (769.8 KB)  TX bytes:769821 (769.8 KB)

robur131@robur131-HP-ProBook-440-G1:~/Desktop$ arp -a
? (192.168.0.111) at ac:ed:5c:df:0c:50 [ether] on enp2s0
? (192.168.0.103) at 00:e0:4c:c0:07:52 [ether] on enp2s0
? (192.168.0.1) at c4:e9:84:88:05:3e [ether] on enp2s0
robur131@robur131-HP-ProBook-440-G1:~/Desktop$
```

Figure 10: Correct MAC Table Of Attacker

5.2.2 Sniffing Inside C program

Once the network interface card of the attacker intercepts the packets, it begins to extract meaningful information from the packets such as source IP address,

destination IP address, source port, destination port, sequence number, acknowledgement number, and so on. Afterwards the attacker can begin with spoofing by carefully constructing an RST packet with the information that he/she has gained.

5.3 RST Packet Spoofing

The server starts streaming. We have used VLC media player for streaming. The server specifies a file, uses its IP, and specifies a port in order to start streaming. At this point the server has an incorrect mac address table.

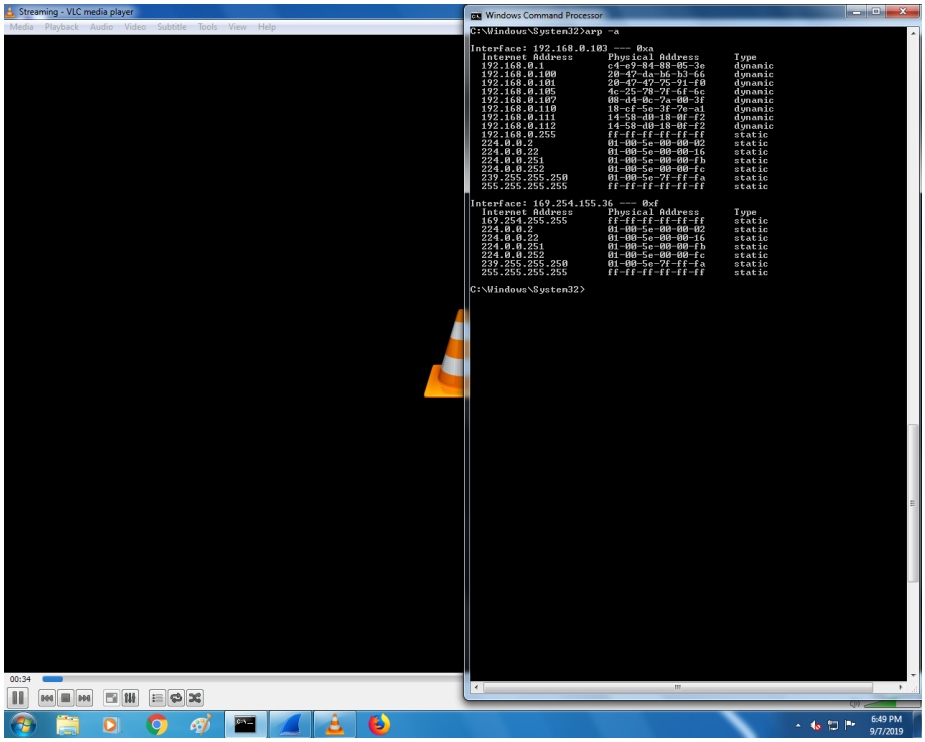


Figure 11: Server starts streaming video on VLC media player

Once the server has started streaming, the victim can connect to the server in order to watch the video. The victim also starts its VLC media player and specifies the IP and port of the server. Then the stream starts on victim's end.

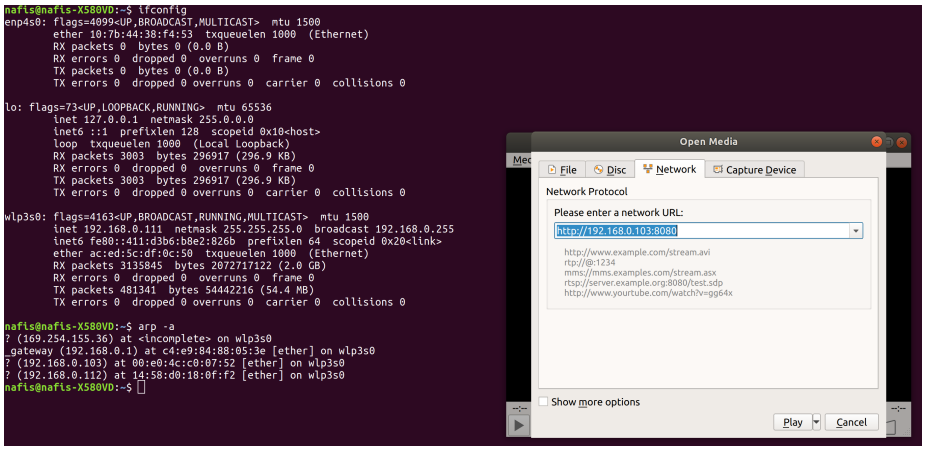


Figure 12: Victim Connecting To Stream by Specifying Server IP and Port

When the victim successfully connects to the server, the streams starts running normally on victim's end. It is evident from the figure below.

Figure 13: Stream Running Normally on Victim's Machine

Now the attacker stars his program. It is program written in ***C***. The program intercepts any TCP packet on attacker's network interface. It does so successfully in the context of the server and the victim because our ARP spoofing Man in the Middle attack is in place. Once it catches a packet, a callback function ***got_packet*** is invoked.

The callback function does two things:

- Extract information such as source IP, destination IP, source port, destination port, sequence number
- Construct an RST packet with the aforementioned information and send the spoofed packet to victim

Figure 14: Attacker's Program is running which spoofs RST packets

From the figure above, we can see that the attacker's program started running. ***enp2s0*** is the interface of the attacker. Number of packets -1 denotes that the program will continue sniffing for an unspecified number of packets until it encounters an error. Lastly, filter expression ***tcp*** denotes that we'll be capturing only tcp packets.

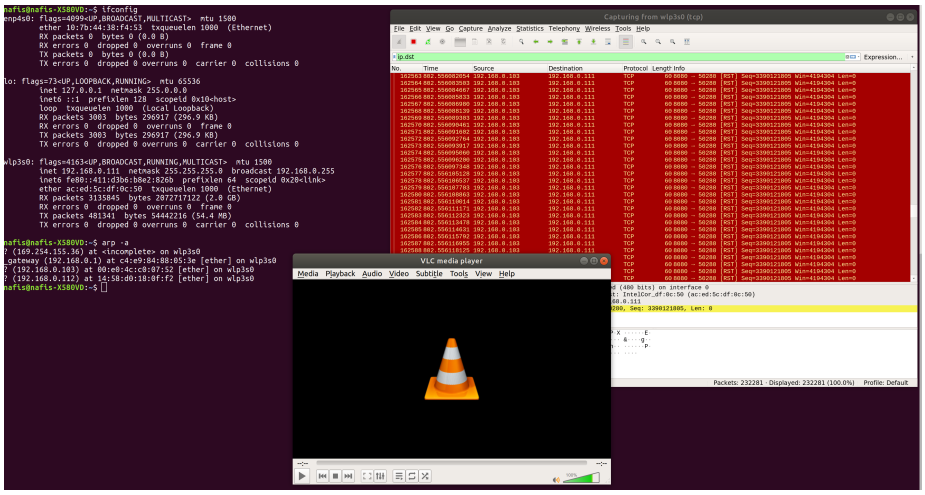


Figure 15: Stream closed on Victim as a result of RST Attack

After the attacker starts his program, the victim's machine continues running the stream for some amount of time. We may not be able to see the effect immediately, because most of the video players have buffers. Waiting for the player to finish playing the video in the buffer will cause the stream connection to terminate. It is evident from opening ***Wireshark*** on the victim machine that the attacker has successfully spoofed RST packet.

6 Conclusion

TCP Reset attack was successfully achieved on video streaming applications in a real LAN environment.