

CSE 406

Computer Security Sessional

Term Project: Design Report

Group No:06

Student IDs:

1605022 -- (Responsible for **Attack no. 2**, Packet sniffing attack and sniff HTTP/telnet passwords)

1605008 -- (Responsible for **Attack no. 16**, Optimistic TCP ACK attack (streaming server))

1605020 -- (Responsible for **Attack no. 10**, ICMP ping spoofing)

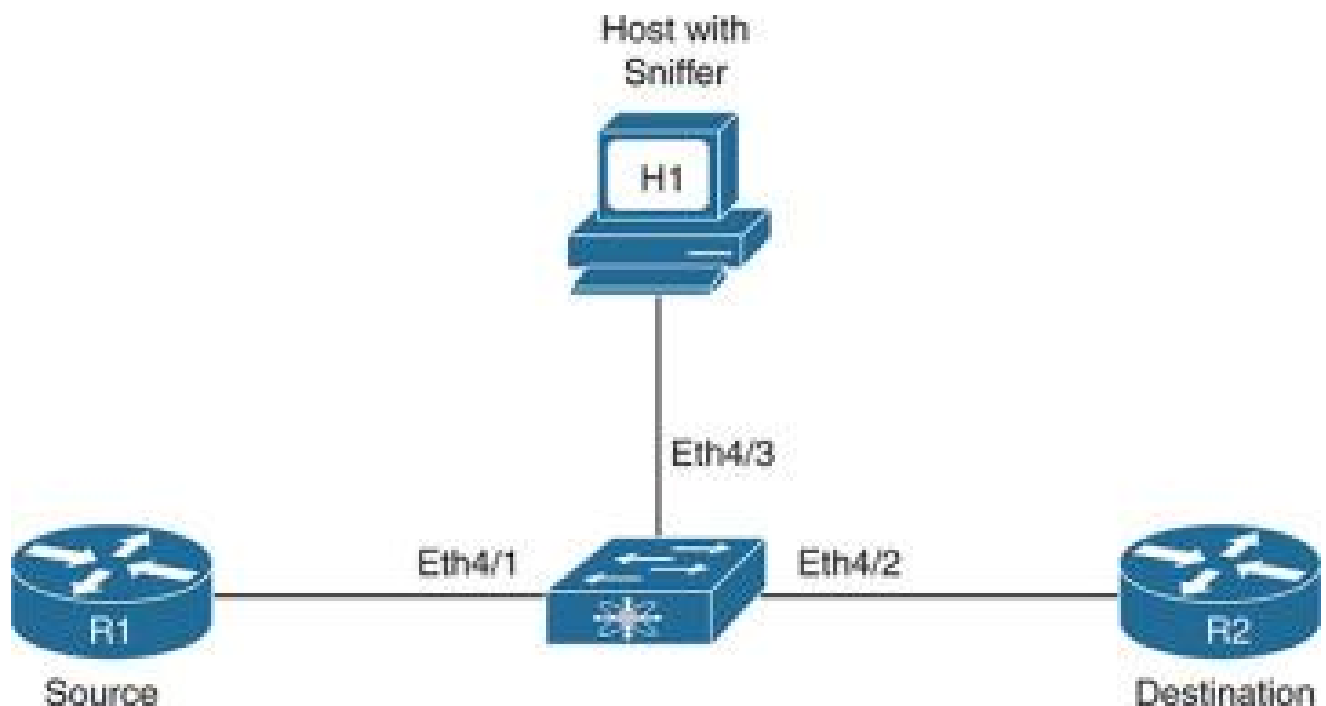
Attack No 2

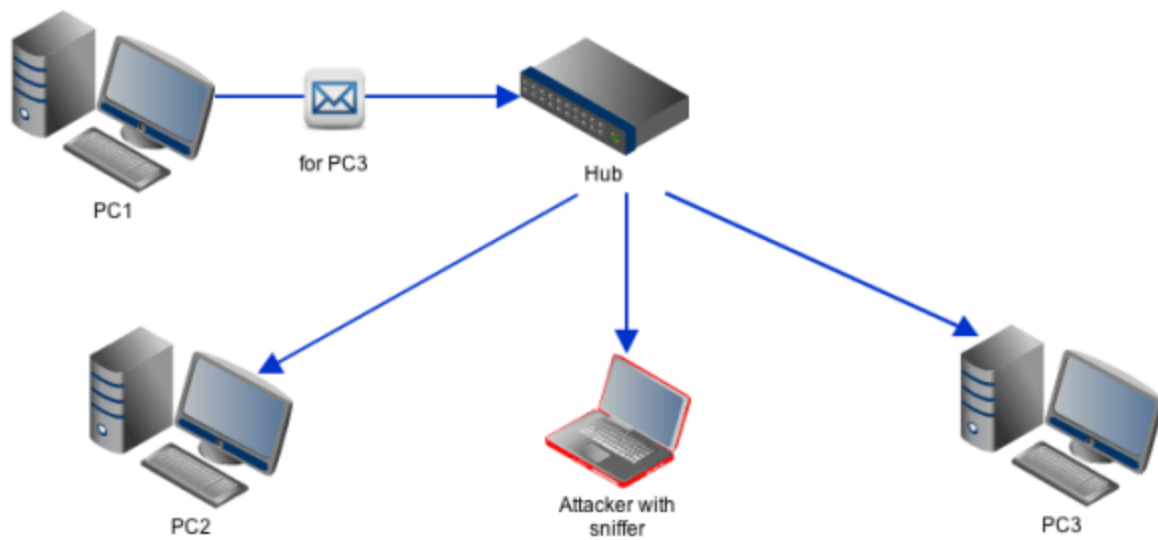
Packet sniffing attack and sniff HTTP/telnet passwords

Packet Sniffing Attack:

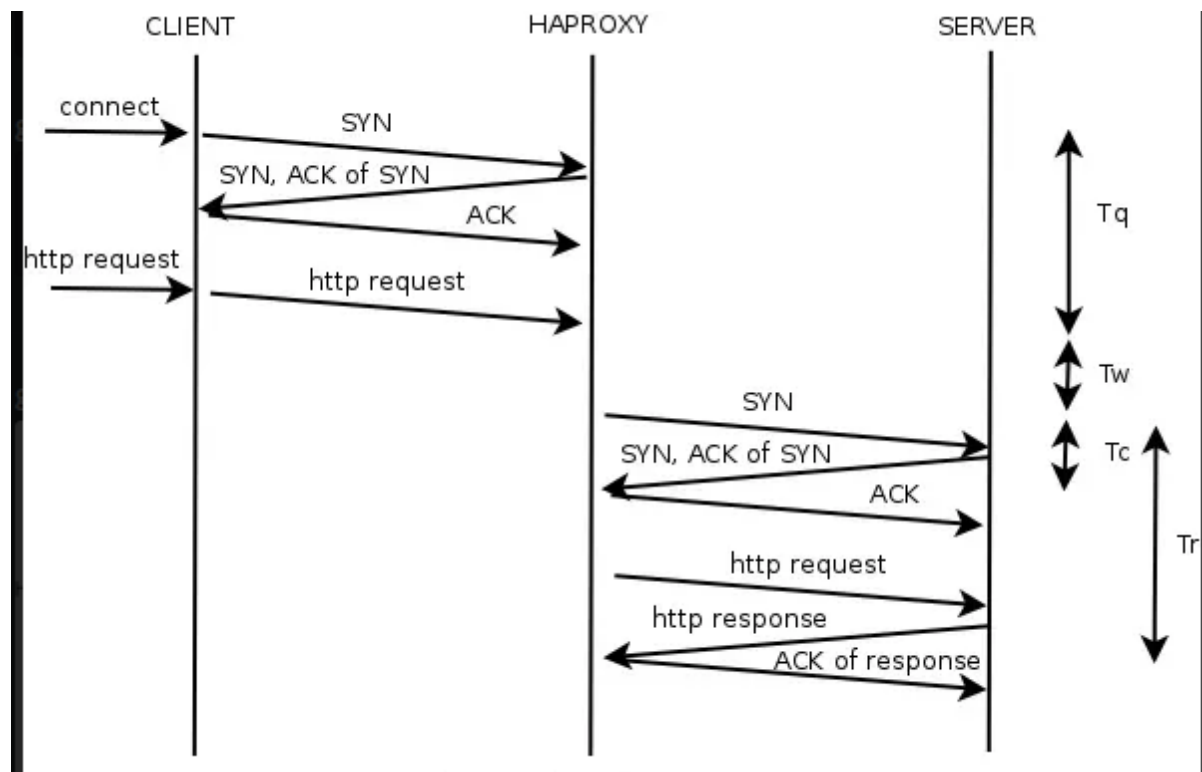
In the context of network security, a sniffing attack or a sniffer attack corresponds to theft or interception of data by capturing the network traffic using a packet sniffer program. An attacker (commonly interpreted as some entity who wants to do some harm to the clients) can use a sniffer program to analyze and corrupt network traffic and read the data communications between them.

Topology Diagram:





Timing Diagram:



Attack Strategy:

Every device has **Network Interface Cards(NIC)**. Machines in a network communicate among each other via NIC cards. Every NIC card has a unique MAC address. NICs read all the network frames in the traffic. When an Ethernet frame arrives from the network, it is copied into the memory inside the NIC. If the destination address in the header matches the NIC's MAC address, then the frame is copied into the Ring Buffer(A kernel buffer). Otherwise, the frame is discarded. If a NIC is configured in **Promiscuous Mode**, NIC passes every frame received from the network to the kernel, regardless of whether the destination MAC address matches with the card's address or not.

Using this concept we can design our attack. The step by step process might be like the following:

1. At first, create two virtual machines(VM).
2. Configure one as the victim and the other as the attacker.
3. Establish a LAN connection between the attacker and the victim.
4. Configure the attacker machine in **Promiscuous Mode**.
5. Using **Pcap API** or **raw sockets**, capture Ethernet Frames from the victim's HTTP post request. After capturing the Ethernet frame, the Frame header, IP header, TCP header must be removed. Thus we will get the HTTP Post request content.

Packet / Frame details:

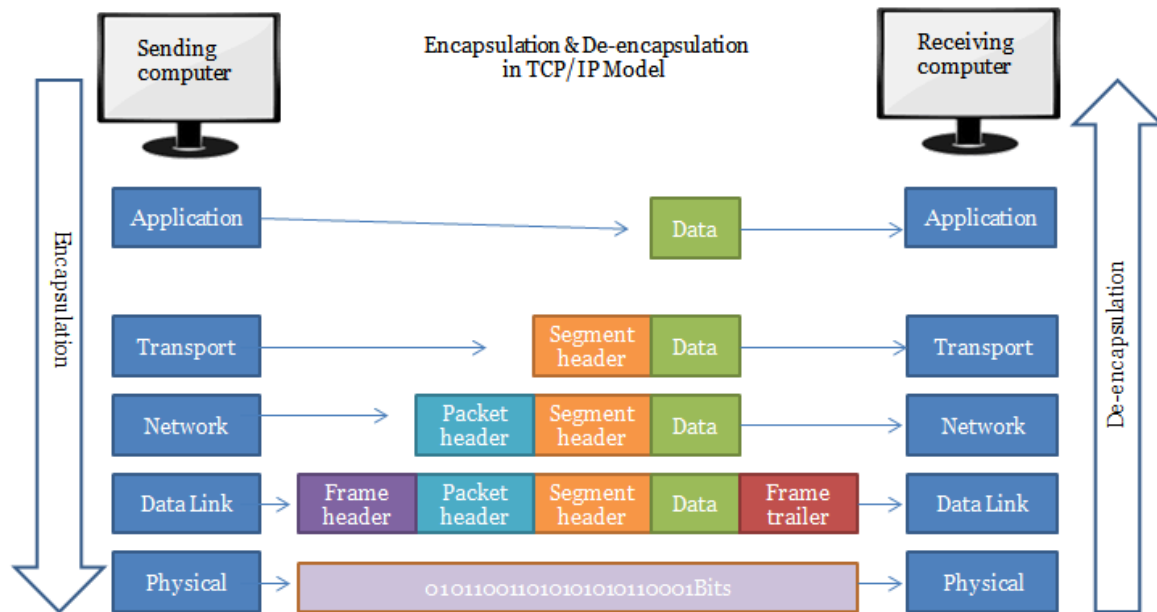
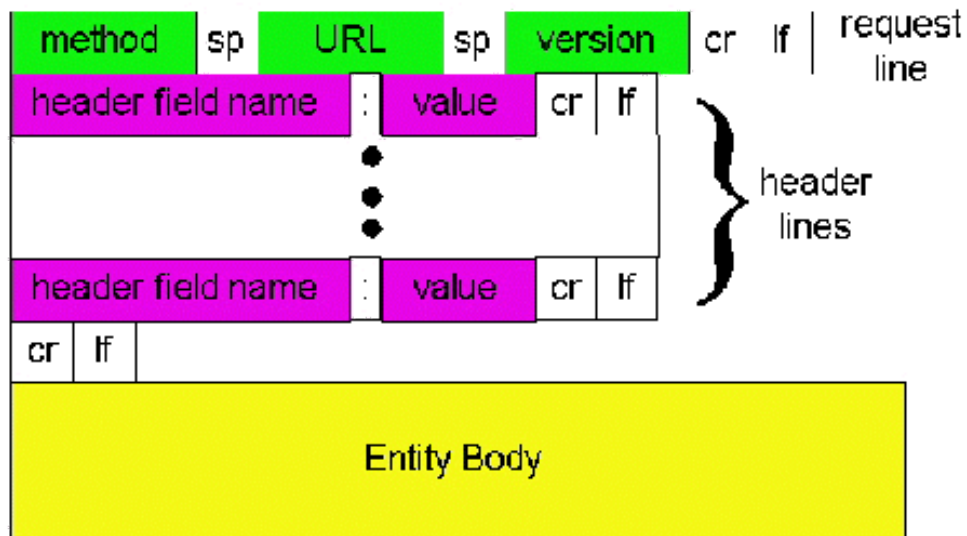
Ethernet frames will be captured by the sniffer program.

Subsequent removal of Frame header, IP header, TCP header gives us HTTP POST Request content of the victim.

Justification: Why my design should work:

The design should work because after setting a machine's NIC in promiscuous mode, it can read all the ethernet frames from the internet. We can always check in our code whether the retrieved packet is of the type IP packet and whether it contains HTTP Post request content.

http request message: general format



16.

Optimistic TCP ACK Attack (Streaming Server)

Definition Of The Attack :

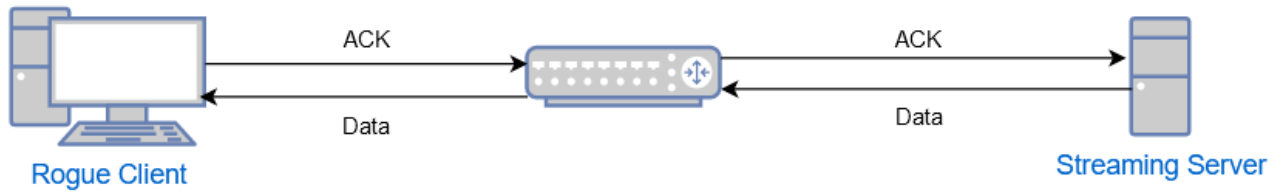
Optimistic TCP ACK attack is a DoS (Denial Of Service) or DDoS (Distributed Denial Of Service) attack. It makes the TCP congestion control mechanism work against itself. So, first we look into TCP congestion control mechanism.

TCP congestion control mechanism : In a TCP communication session, a concept of congestion window is used. When a connection is established (details will be discussed later), server sends Data to client. And client sends ACK (a packet) to server. This ACK indicates that Data was received without any loss. As a server receives ACK from a client, it dynamically adjusts the congestion window size to reflect the estimated bandwidth available. So, *Congestion Window* fix number of TCP packets allowed to be sent. The window size grows when server receives ACKs, and shrinks when segments arrive out of order or are not received at all - which indicates Data was missing or was not received by client. This congestion control nature of TCP automatically adjusts as network conditions change, shrinking the congestion window when packets are lost and increasing it when they are successfully acknowledged. More ACK comes to the server, it increases sending Data to client.

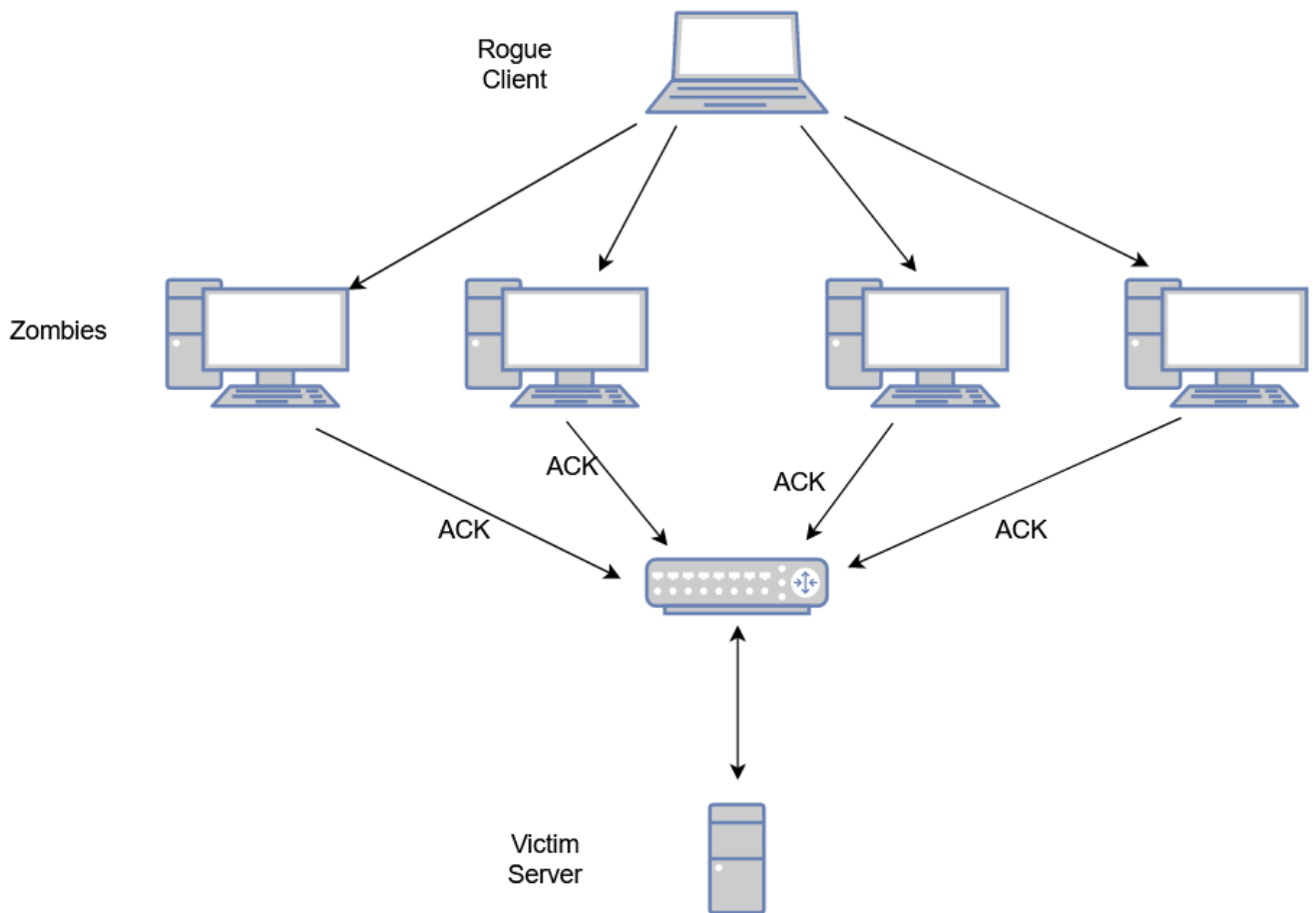
Optimistic ACK attack : Here, a rogue client tries to increase server's sending rate until it's whole bandwidth is covered and can't serve anymore. An *optimistic* ACK is an ACK sent by a client for a Data segment that it has not yet received. The attack is done by the client sending ACKs to Data packets before they have been received. The aim of the client is to acknowledge "in-flight" packets, which have been sent by the server but have not yet been received by the client. As a result, the server believes that the transfer is progressing better than it actually is and may increase the rate at which it sends packets.

At some point, server can't send any packets anymore because it's bandwidth is full. That's why Optimistic ACK is a DoS attack. The rogue client can use a group of machines to do the job. A distributed network of compromised machines ("zombies") can exploit this attack in parallel to bring about wide-spread, sustained congestion collapse. And then it can be called DDoS attack.

Topology Diagram :



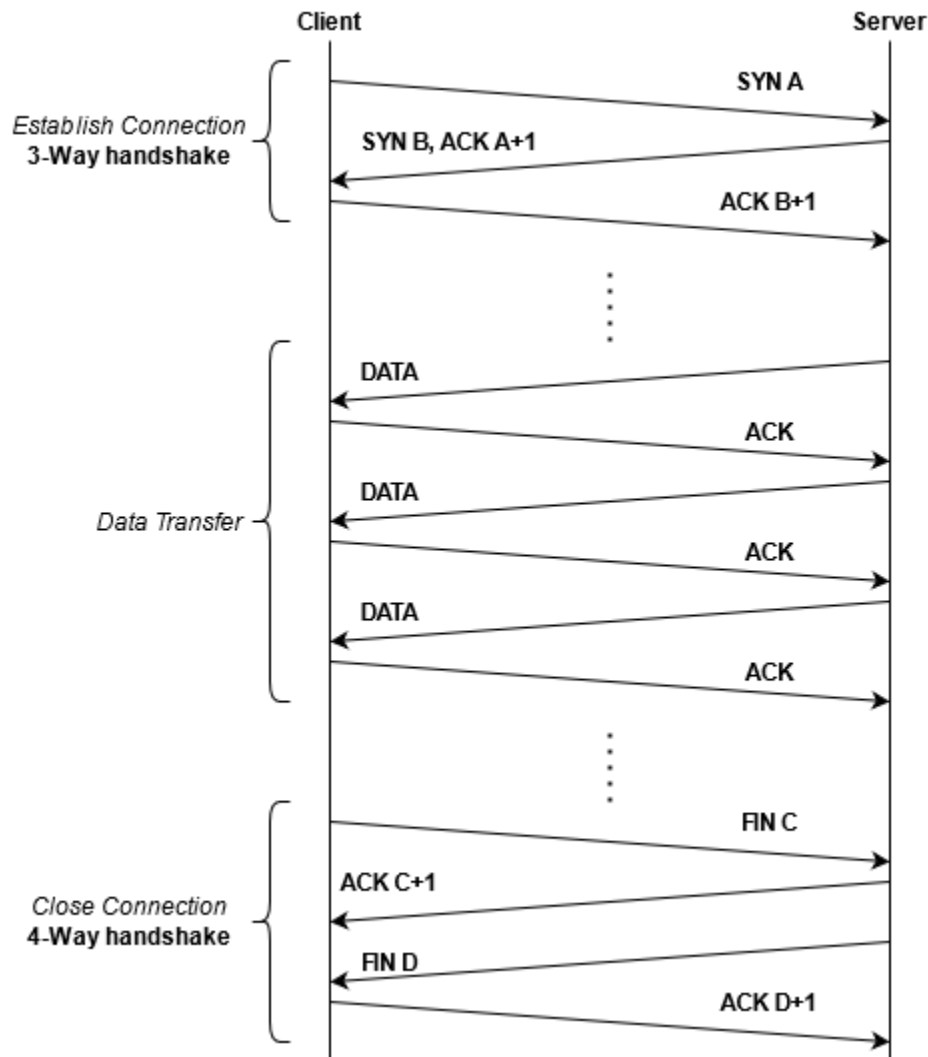
DoS Topology



DDoS Topology

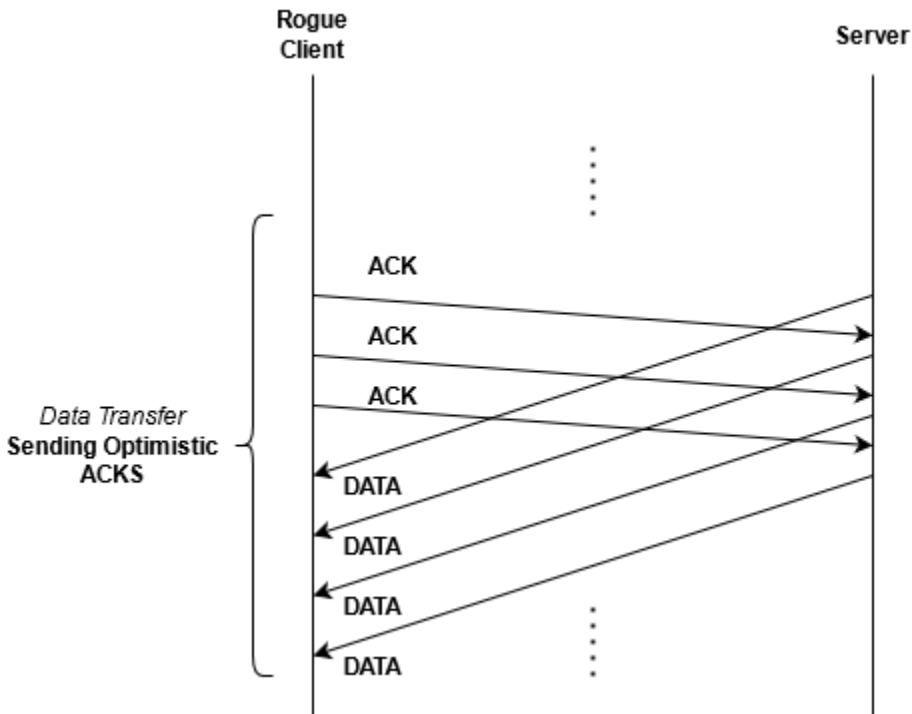
Timing Diagram :

TCP communication has 3 phases : Establish Connection, Data Transfer, Close Connection. Connection Establishment is a 3-way handshake method and Connection Closing is a 4-way handshake method.



In a normal TCP communication, we can see, in the *Data Transfer* Phase, that the server sends Data and then client sends ACK. That means client never sends ACK before receiving Data. Upon receiving ACKs the congestion window is adjusted. More ACKs is received, more increment in window size, more Data is sent, more bandwidth is covered.

So, the attack strategy is sending ACKs of some Data packets to server which are sent from server, but not yet received. When server receives those ACKs it makes an observation that sent packets are received, which is not true. Then, it starts sending more packets. At a point, when server bandwidth is full, it can't send anything. Results in a Denial of Service. The attack timing diagram is like this :



Packet Details :

Here, our concern is ACK packets which are sent to server. ACK is short for "acknowledgement." An ACK packet is any TCP packet that acknowledges receiving a message or series of packets. The technical definition of an ACK packet is a TCP packet with the "ACK" flag set in the header. Below is a TCP packet header :

0	8				16				24				32		
Source Port								Destination Port							
Sequence Number															
Acknowledgment Number															
Data Offset		Reserved		C W R	E C E	U R G	A C K	P S H	R S T	S S Y N	F I N	Window Size			
Checksum								Urgent Pointer							
Options											Padding				

So, If the "ACK" flag is set then its a ACK packet. ACK packets are part of the TCP handshake, when establishing and closing a connection. Also, upon receiving Data (even before receiving when having bad intensions), the client sends ACK.

But, the crucial part here is to determine the Acknowledgement Number. The attacker must produce a seemingly valid sequence of ACKs. For an ACK to be considered valid, it must not arrive before the victim (the server) has sent the corresponding packet. Otherwise it server will consider the ACK as a missing data and decrease the congestion window size. Thus, the attacker must estimate which packets are sent and when, based only on the stream of ACKs the attacker has already sent. At first this might seem a difficult challenge, but the victim's behavior on receiving an ACK is exactly prescribed by the TCP congestion control algorithm.

So, to exploit the attack, we need to

1. Create two virtual machines (VM).
2. Configure one as the victim and other one as the attacker.
3. Establish a LAN connection between the attacker and the victim.
4. Build ACK packets, in the header the ACK bit will be set. And also we need to guess the next ACK number properly. These packets will be sent to streaming server(victim).
5. Use Raw Socket for the task

Justification:

Our design is typical Optimistic ACK attack design. When doing implementation, various challenges may arrive, like correct guess of ACK numbers. But as mentioned before, TCP congestion control mechanism or algorithm itself provides to exploit the Optimistic ACK attack. The concept of window for Data passing is the main reason that the attack is possible, because growing the window size will gradually cover the whole bandwidth of the server. One time, server will unable to respond any more which is our goal. If correct implementation is done, it is not impossible at all that the attack will work fine.

Attack No 10

ICMP Ping Spoofing Attack

Definition of the Attack:

Ping is a measurement of how much time it takes for a small dataset to reach a destination and come back to the initial computer. It is usually measured in milliseconds.

ICMP(Internet Control Message Protocol) is a special supporting protocol of IP(Internet Protocol).

ICMP ping spoofing is basically IP spoofing. An attacker sends ping requests with false ICMP packets to random addresses, but with faking its own address. Rather it changes the source address of the ping request. So when a response comes for the ping, it doesn't come to the attacker, and it goes to the victim machine, whose address was given as the source.

So when lots of responses come to the victim machine, the channel goes overloaded. Then the victim cannot send or receive any other requests. So this is a distributed denial of service or DDOS attack.

Topology Diagram:

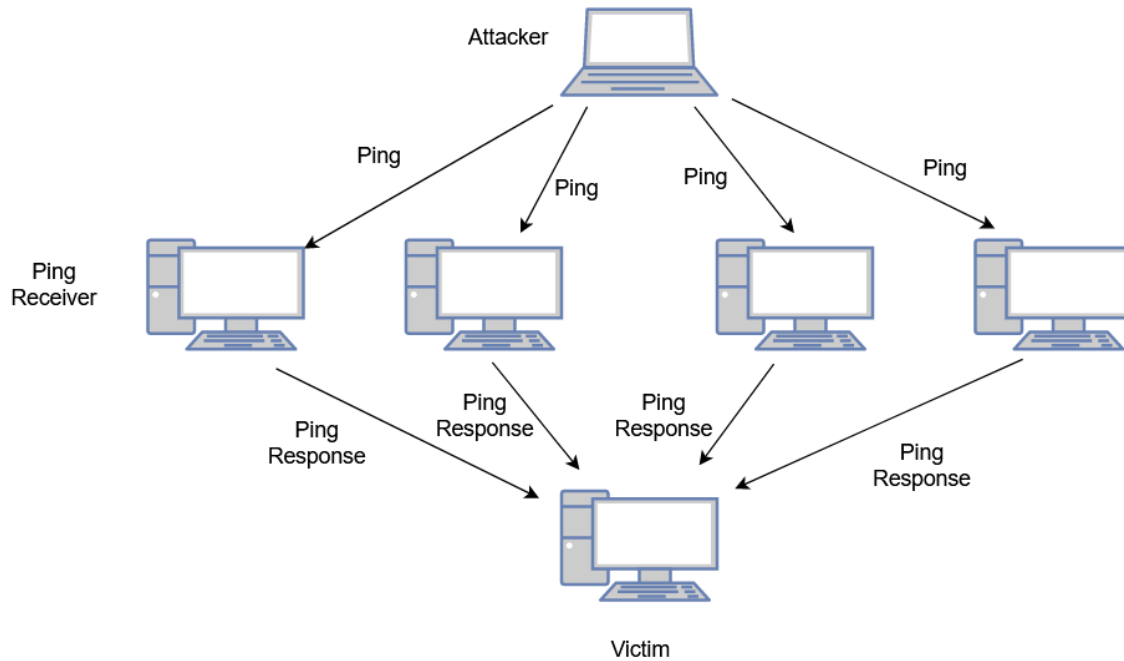


Figure: Ping Spoofing

Timing Diagram:

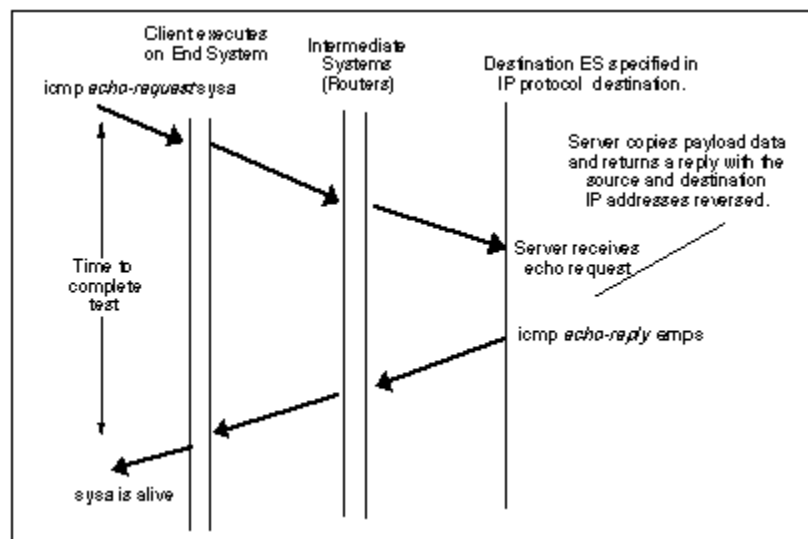



Figure: ICMP ping timing diagram

Packet Details :

Bit Offset	0-3	4-7	8-15	16-18	19-31
0	Version	Header length	Service Type	Total Length	
32	Identification			Flags	Fragment Offset
64	Time to Live		Protocol	Header Checksum	
96	Source Address				
128	Destination Address				
160	(Options)				
160+	Data Data Data Data Data Data Data Data Data Data Data Data ...				



Over-write source address with a different IP address

Figure: Source modification for spoofing attack

Justification :

The machines receiving ping requests get to know the address from the source address field of the packet header received there. But if that field is modified with some different source, the server has no clue where it is coming from. And when the responses overload a victim, it will be very difficult to know who actually sent them. So if implemented correctly, this should work most of the time.

ICMP Redirect Attack

Definition Of The Attack:

ICMP redirection is a feature of IP to find any available better routes for a request going outside LAN. But it has some drawbacks.

When an attacker is connected in the same LAN of a victim pc, it can redirect a request from the victim. It does so by updating the routing table of the victim host. When a pc tries to access an address outside the LAN, it has to go through a gateway. But if the attacker somehow manages to update the routing table of the host, it can tell the victim that there is a better gateway for the destination. If the attacker tells the victim the new gateway, the victim will always give packets to the new one. And hence the packet is redirected by someone in the middle. So this is a Man In The Middle or MITM attack.

Topology Diagram:

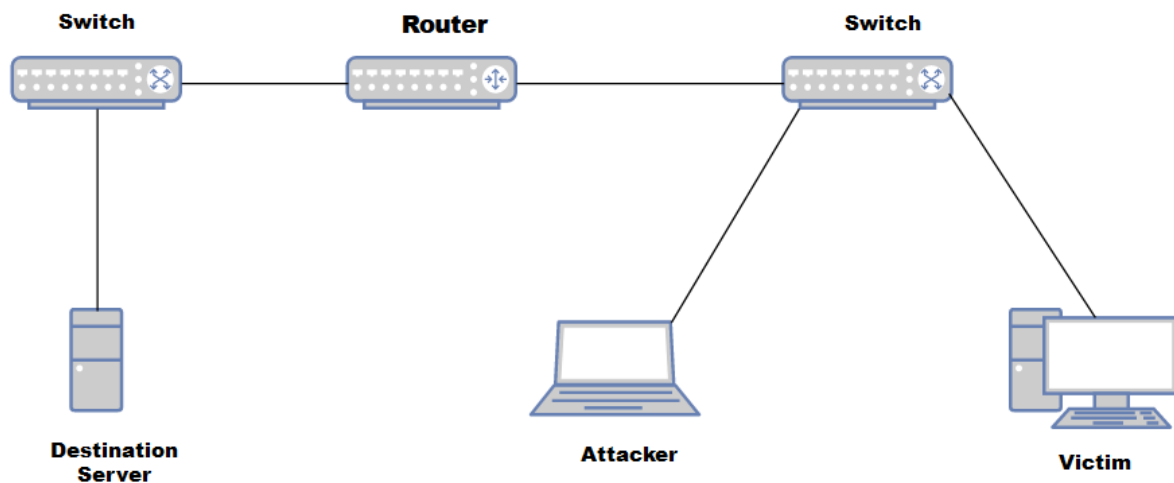


Figure : ICMP redirect attack topology

There is no packet or frame header modification for this attack, the packet remains the same, only gets redirected to new gateways.

Justification:

Redirecting ICMP messages is a feature of IP. It helps the packets take shortest routes to the destination. But it also can harm when someone intentionally fakes shorter available routes. If an attacker is able to update the routing table, by telling the host that the shortest route to its destination is via another gateway, the host or victim will definitely send the packets to the new updated gateway. So this will work.