Aifaz Dhuka
UCID: 30069823

Attacks report

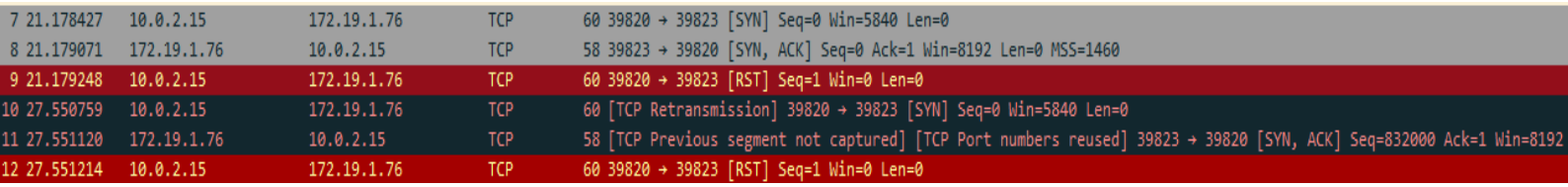
Attack1

1.
    i. syn packet number = 3
    ii. syn ack packet number = 4
    iii. ack packet number=5
2. Server sequence number: 12032001
3. Client sequence number: 790224699

Attack 2

1. Spoofed syn packet number = 1
2. Spoofed source ip = 10.0.2.15 and port=39820
3. It sends back syn ack. And then the OS sees that and does not recognize it as it did not send it. So it sends back RST packet to reset the connection.
4. Spoofed sequence number = 0
5. Server's sequence number = 124416001
6. I think the server does consider this as a new connection as it replies back with syn ack. Although, the (relative) sequence number seemed to have gotten messed up on the second reply. Also, some devices could store syn cookies leading them to know if a syn was previously sent or not and track the activity. Thus using this they could start ignoring packets (probably for a certain period) if they find its activity suspicious. To solve this, using random source ip and ports could avoid this syn cookie detection. This could also probably avoid the tcp segment not captured error.

   Column 10 is same packet retransmitted, and column 11 is the response from the server.



Attack 3

1. Reset message is sent to Server before it sends data back to the client. This is done when the server is sleeping.
2. Yes, since the server disconnects from the current client while the client still thinks they are connected.
3. rst packet at column number 6 is the spoofed packet.
4. Client connected from ip = 10.0.2.15 and port = 53776
5. The client continues waiting for the response from the server until it is terminated forcefully (packet 10 => by pressing ctrl-c or because of timeout if timeout feature is implemented) as it still believes that it is connected to the server when it is in fact not connected.

Attack 4

1. Psh ack in column number 6 is the spoofed packet sent that contains a different data.
2. Client connected from ip = 10.0.2.15 and port = 33972
3. The server sends fin ack and when it doesn't receive any reply it keeps on sending again and again. Then the actual message (packet #29) is sent to the server by the client which gets considered as duplicate and gets ignored. Client also sends a fin ack (packet #31). Then when the retransmitted fin ack (packet #32) from the server is received (for the spoofed message), the client thinks its fin ack got a fin ack back and sends an ack (packet #33) back. And the line disconnects.