Attacks on the TCP Protocol

16.1. This problem is based on the TCP client program shown in Listing C.1 (C is the chapter number of the TCP Attack chapter; its actual value depends on which version of the book you are using). (1) To get responses from the server, the TCP client program should register for a source port number, but in the program, this step seems to be missing. Without this port number, how can the client program get responses? (2) Which line of the code triggers the three-way handshake protocol? (3) There are two `write()` calls in this client program, will each call trigger a separate TCP packet?

16.2. This problem is based on the TCP server program shown in Listing C.2. (C is the chapter number of the TCP Attack chapter; its actual value depends on which version of the book you are using). (1) Does the program get blocked when invoking `listen()` until a connection comes? (2) What is the purpose of the `accept()`? (3) Why does the `accept()` call create a new socket? Why cannot we use the same one that is used in the `listen()` call?

16.3. We have two machines, A and B. (1) Two TCP client programs on machine A send their data to a TCP server that is listening to port 8023 on machine B. Will the data from these two client programs be mixed together on the server side? Please explain. (2) Two UDP client programs on machine A send their data to a UDP server that is listening to port 8023 on machine B. Will the data from these two client programs be mixed together on the server side? Please explain.

16.4. A program wants to send many pieces of data to a server, each piece will be sent via a separate call. The server needs to know the boundaries among these pieces. (1) If the program uses UDP, how does the server know where the boundaries are? (2) What if the program uses TCP?

16.5. Does a SYN flooding attack cause the victim server to freeze?

16.6. In the SYN flooding attack, why do we randomize the source IP address? Why cannot we just use the same IP address?

16.7. What will happen if the spoofed source IP address in a SYN flooding attack does belong to a machine that is currently running?

16.8. An attacker launches a SYN flooding attack against the `telnet` server on a target machine. This particular `telnet` server listens to two ports, port `23` and port `8023`. The attack is only targeting the default `telnet` port `23`. When the attack is undergoing, can people still be able to telnet to the server using port `8023`?

16.9. Can we launch a SYN flooding attack from a computer without using the root privilege?

16.10. Why do we choose to fill up the memory used for half-open connections, why cannot we directly target the memory used for holding full connections? The latter requires more memory, so the resource is much easier to exhaust.

16.11. If TCP always uses a fixed sequence number (e.g., zero) in its `SYN + ACK` packet during the three-way handshake protocol, please describe how you can conduct a denial-of-service attack on the TCP server. Your objective is different from the SYN flooding attack; you want to cause the server to establish connections with many non-existing computers, and thus exhausting the server's resources, especially its memory.

16.12. All the information that a server needs to know about a connection is not only contained in the `SYN` packet, but also in the final `ACK` packet from the client. Therefore, information-wise, there is no need to allocate a buffer to save the information about half-open connections. If we get rid of this buffer, the SYN flooding attack will not be effective any more. Do you agree with such a statement or not. Please justify your answer.

16.13. To reset a connection between two remote machines, i.e., we will not be able to see the packets between these two machines, what are the main challenges?

16.14. Are TCP Reset attacks effective against encrypted connections, such as SSH?

16.15. Is UDP communication subject to reset attacks?

16.16. There is an active Telnet connection from a client (`10.0.2.5`) to a Telnet server (`10.0.2.9`). The server has just acknowledged a sequence number `1000`, and the client has just acknowledged a sequence number `3000`. An attacker wants to launch the TCP session hijacking attack on the connection, so he can execute a command on the server. He is on the same local area network as these two computers. You need to construct a TCP packet for the attacker. Please fill in the following fields:

- Source IP and Destination IP
- Source port and Destination port
- Sequence number
- The TCP data field.

16.17. In a TCP session hijacking attack, if the server is waiting for data starting from sequence number `X`, but we used `X + 100` in our attack packet. Will our attack succeed or fail?

16.18. Can we launch a TCP session hijacking attack against an SSH connection?

16.19. ★
The Mitnick attack is a variation of the TCP session hijacking attack. This attack involved two computers (we will call them A and B) in San Diego Supercomputer Center. B trusted A, so if somebody logs in from A, no password would be asked. Kevin Mitnick wanted to log into B, but he did not know the password, and he had no access to A either. He could only do that remotely. To get in, he would have to fool B to believe that his login request was from A.

Before the login program runs, a TCP connection needs to be made first. Therefore, Mitnick needed to forge a TCP connection request from A to B first. If the connection is established successfully, Mitnick would have all the parameters about the connection, including the port numbers and sequence numbers. He could then use this connection to log into B, and steal information from there.

To simplify the scenario, let us assume that computer A was not even running; only B is running. Please describe how Mitnick would get B to establish a connection with A. In

those days, TCP's initial sequence numbers were not randomized, and they were quite predictable.

16.20. UDP services can be used for amplification attacks. Why cannot TCP be used for the same attack?

16.21. In the past, we wrote a SYN flooding program using Python, but we could never get the attack to work; there is nothing wrong in the program. After a close look at the attack, we found out that he speed of our Python program is too slow: it can only send out a few spoofed packets in a second. We also found out that there are many Rest packets coming back to the victim machine. Based on this observation, please explain why our Python program could not get the attack to work.