Attacks on the TCP Protocol

N6.1. This problem is based on following TCP client program. (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 `sendall()` calls in this client program, will each call trigger a separate TCP packet?

```
#!/bin/env python3
import socket

tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp.connect(('10.0.2.69', 9090))

tcp.sendall(b"Hello Server!\n")
tcp.sendall(b"Hello Again!\n")

tcp.close()
```

N6.2. This problem is based on the following TCP server program. (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?

```
#!/bin/env python3
import socket

tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcp.bind(("0.0.0.0", 9090))
tcp.listen()

conn, addr = tcp.accept()
with conn:
    print('Connected by', addr)
    while True:
        data = conn.recv(1024)
        if not data:
            break
        print(data)
        conn.sendall(b"Got the data!\n")
```

N6.3. 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.

N6.4. 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?

N6.5. Does a SYN flooding attack cause the victim server to freeze?

N6.6. In the SYN flooding attack, why do we randomize the source IP address? Why can't we just use the same IP address?

N6.7. What will happen if the spoofed source IP address in a SYN flooding attack does belong to a machine that is currently running?

N6.8. 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`?

N6.9. Can we launch a SYN flooding attack from a computer without using the root privilege?

N6.10. 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.

N6.11. 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.

N6.12. 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.

N6.13. 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?

N6.14. Are TCP Reset attacks effective against encrypted connections, such as SSH?

N6.15. Is UDP communication subject to reset attacks?

N6.16. There is an active connection between a Telnet client (`10.0.2.5`) and 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.

N6.17. 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?

N6.18. Can we launch a TCP session hijacking attack against an SSH connection?

N6.19. There is an active TCP connection (a telnet session) between two computers, and we have captured the most recent packet sent over this connection. Please construct a packet to hijack this session, so you can run the `"/bin/rm -f /"` command on the server. Due to a technical error in the sniffer program, the last two digits of the acknowledgment number in the captured packet are corrupted. That is why you see `**`.

There is no need to write a program. Showing the essential fields of the packet should be sufficient. Brief explanation should also be included.

```
▶ Internet Protocol Version 4, Src: 10.9.0.7, Dst: 10.9.0.6
▼ Transmission Control Protocol, Src Port: 23, Dst Port: 45634
    ...
    Source Port: 23
    Destination Port: 45634
    Sequence number: 2737422009
    Acknowledgment number: 7185323**
    ...
▶ TCP Data, length: 20 bytes
```

N6.20. 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.

N6.21. UDP services can be used for amplification attacks. Why can't TCP be used for the same attack?

N6.22. 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 the 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.

N6.23. Machine A (its IP address is `10.9.0.5`) inside an organization runs a TCP server using `"nc -l 9090"`. Due to the firewall, no outside machine can send data to this TCP server, except machine B (its IP address is `192.168.60.5`). Namely, for a packet from the outside to reach this TCP server, the source IP address must be B. You have noticed the following facts: (1) machine A always uses a fixed number `0x1000` as its initial sequence number during the TCP three-way handshake protocol, and (2) machine B is currently offline.

You, as a remote attacker, want to send a message to this TCP server, but you have no access to machine B, nor can you sniff any packet from machine A. Please write a Python program to achieve this goal. A sample Python code is listed in the following, showing how to create a TCP packet.

```
# Create an IP object
# src: source IP, dst: destination IP
ip = IP(src="1.1.1.1", dst="2.2.2.2")

# Create a TCP object
# sport: source port,      dport: destinatin port
# seq:   sequence number,  ack: acknowledge number
# flags: see the notes below
tcp = TCP(sport="33333", dport="23",
          seq=8000, ack=5000, flags="SA")

# Here are a few examples of the TCP flags
# flags = "A" : Set the ACK bit only
# flags = "S" : Set the SYN bit only
# flags = "SA": Set the SYN and ACK bits only

# Construct and send the packet
pkt = ip/tcp
send(pkt, verbose=0)
```