TCP SYN Flooding Attack Lab

Contents

Lab Objectives and Setting

Overview

The Internet protocol suite, also known as TCP/IP suite, is the foundation of the modern Internet. This suite provides various protocols that lie upon the Internet Protocol (IP) for end-to-end data communications. Vulnerabilities in TCP/IP protocols may have serious effects on the upper layer protocols and applications and can endanger communication and data security. In this lab, we will look into TCP protocol, learn how it works and analyze its vulnerabilities, and show a way to attack it.

Objectives

Upon completion of this lab, students will be able to:

- Recognize the operation procedure and data format of TCP protocol;
- Explain how TCP SYN flooding attack works;
- Analyze the vulnerability of TCP and give feasible protection solution.

Prerequisites

- Practical experience with SSH and basic Linux commands;
- Basic knowledge of network protocols.

EZSetup

EZSetup is a Web application capable of creating various user-defined cybersecurity practice environments (e.g., labs and competition scenarios) in one or more computing clouds (e.g., OpenStack or Amazon AWS). EZSetup provides a Web user interface for practice designers to visually create a practice scenario and easily deploy it in a computing cloud, which allows for customization and reduces overhead in creating and using practice environments. End users are shielded from the complexity of creating and maintaining practice environments and therefore can concentrate on cybersecurity practices. More information about EZSetup can be found at https://promise.nexus-lab.org/platform/.

Environment Setting

In this lab, students can access three virtual machines (VM) from EZSetup, attacker, victim, and observer VM. The network topology is shown in Figure 1, the VM properties are listed in Table 1. Please refer to the EZSetup dashboard for the actual public IP addresses and passwords.

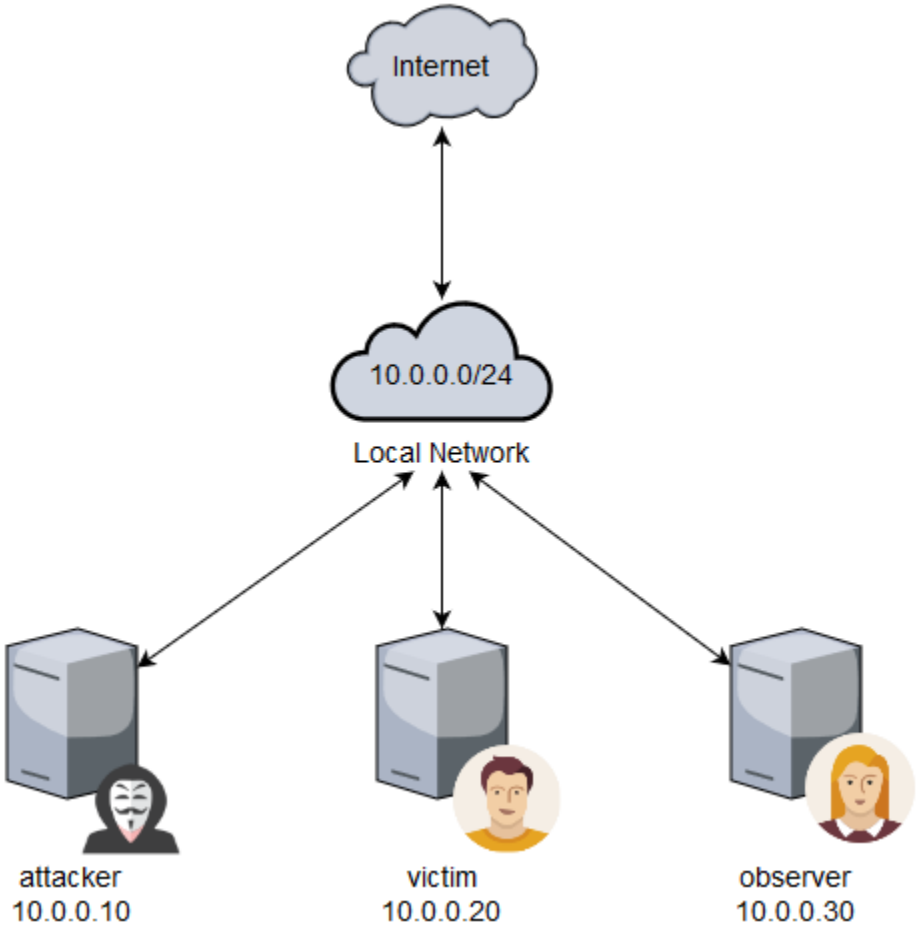

Figure 1 Lab network topology

Table 1 VM properties and access information

Name	Image	RAM	VCPU	Disk	Login account
ts-attacker	tcpipsecurity-attacker	2GB	2	40GB	See EZSetup
ts-victim	tcpipsecurity-victim	4GB	2	60GB	See EZSetup
ts-observer	tcpipsecurity-observer	2GB	2	40GB	See EZSetup

License

TCP SYN Flooding Attack

Transmission Control Protocol (TCP) is an essential protocol of the TCP/IP protocol suite. It is a transport layer protocol which provides highly reliable and ordered host-to-host communication in computer networks. Many Internet applications and high-level protocols rely on TCP, such as World Wide Web (HTTP), Email (SMTP) and Secure Shell (SSH). On the lower level, TCP uses the IP protocol, which is responsible for addressing hosts and routing data between hosts.

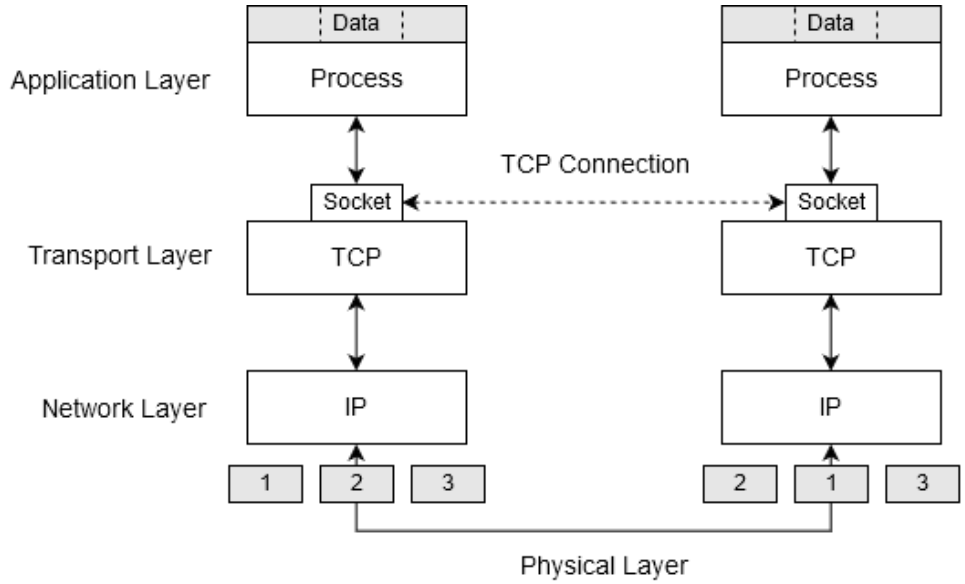

Figure 2 TCP Connection between two remote processes

TCP is connection-oriented, so applications using TCP need to establish a connection before exchanging data. TCP uses a socket, which is an IP address and TCP port pair, at each end of a connection so application processes TCP can be differentiated. TCP also supports full duplex, which means both sides of a TCP connection can send or receive data streams simultaneously. Figure 2 illustrates how a TCP connection between two processes of two different hosts works.

To understand how TCP operates to achieve a reliable communication, we should first take a look at the TCP message format, which is shown in Figure 3. The size of a TCP packet header is 20 bytes unless additional options are present. The **source port** and **destination port** fields are the sender and receiver TCP ports of a connection. The **sequence number** identifies the position of a packet's first octet in all the data of the current session. When SYN is set to 1, the sequence number is the initial sequence number and the corresponding acknowledgment number is this sequence number plus 1. The **acknowledgment number** is the sequence number that the sender expects for the next incoming TCP packet when ACK flag is set. **Header length**, or data offset, denotes the TCP header size in 32-bit words because the length of the options field is variable. The **reserved field** is for future use and is set to zero.

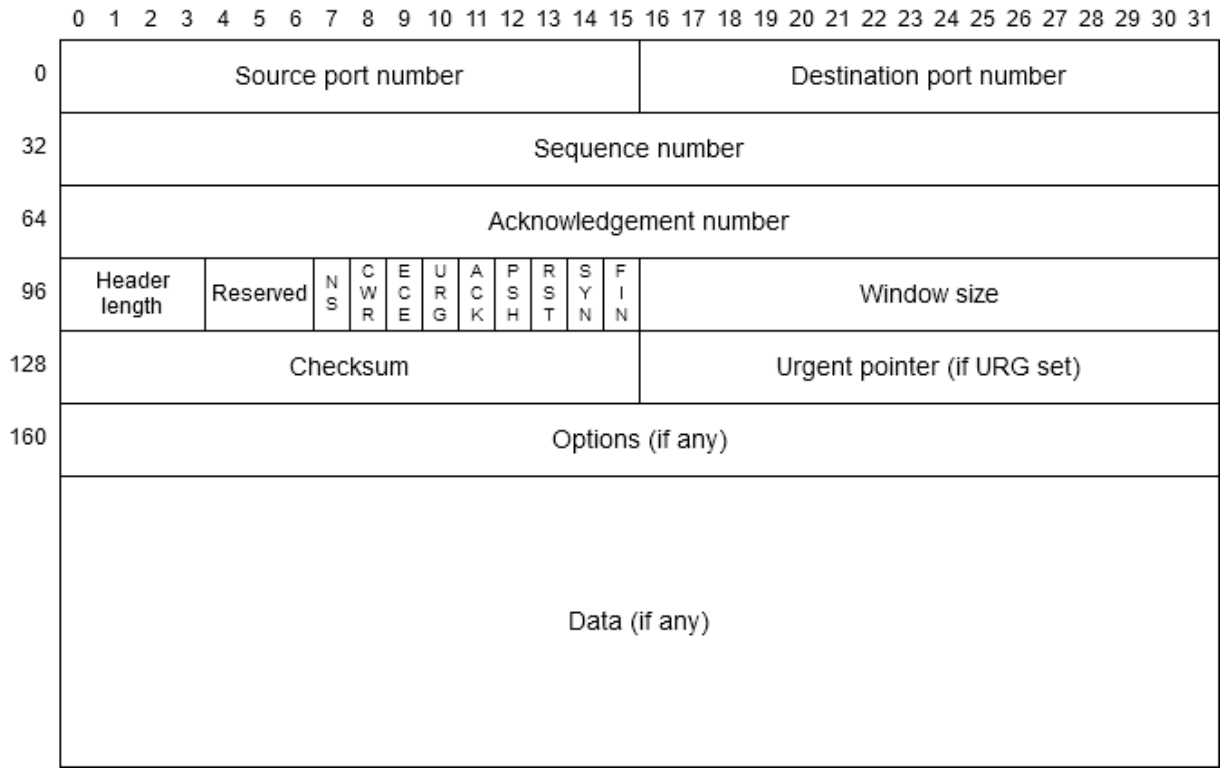

Figure 3 TCP message format

The following 9 bits are control flags. The **SYN** and **FIN** flags are set respectively when establishing and terminating a TCP connection. The **ACK** is always set after the initial SYN packet is received and indicates the acknowledgment field is significant. The **RST** flag means the sender wants to reset the connection. The **URG** flag signifies this TCP segment contains urgent data. If it is set, the **urgent pointer** indicates the offset of last urgent data byte from the sequence number. The **PSH** flag asks the receiver to push buffered data to its application. The **window size** field signifies the number of unacknowledged data octets the sender wants to receive, and its value usually depends on the amount of memory available to this connection. The **checksum** field is for error checking by the receiver. And the **options** field is optional, with zero paddings to make sure data starts on a 32bit boundary.

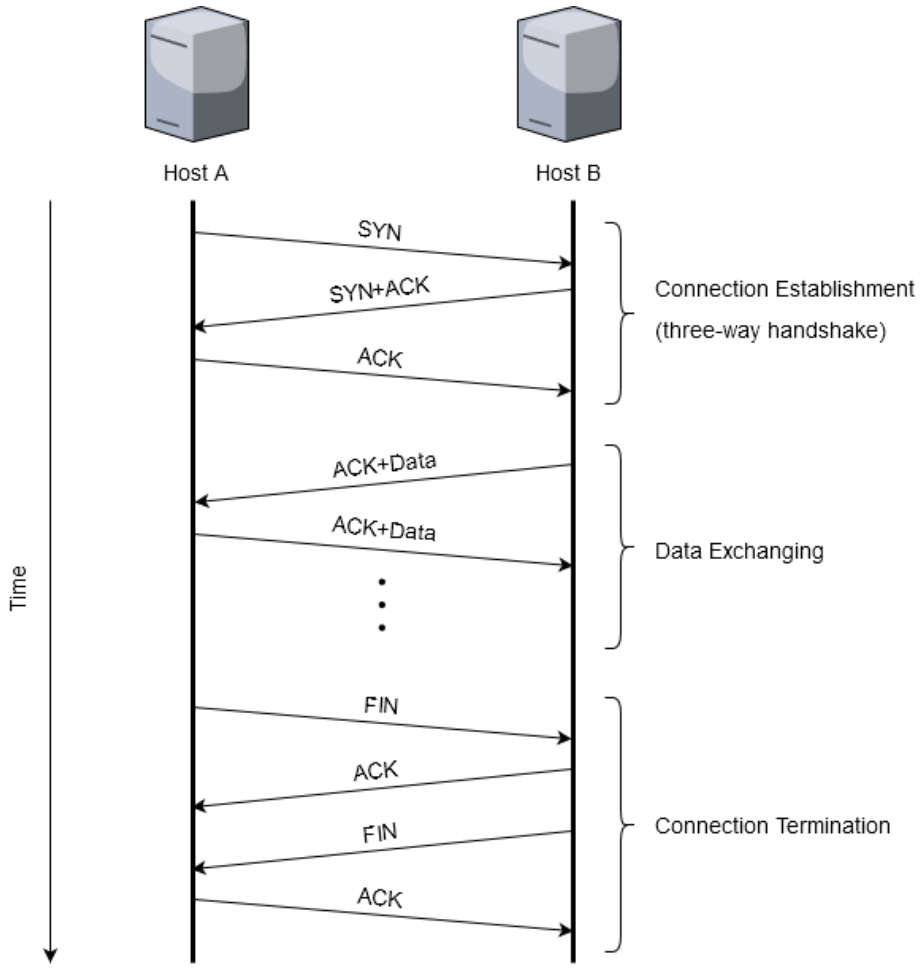

Figure 4 Packet flow in a TCP session

Figure 4 shows the simplified process of the packet flow during a TCP session. To establish a TCP connection, clients on both sides use a three-way handshake. The client that initiates the connection first sends a packet with SYN flag set to its receiving client (or TCP server for short), and the sequence number is set to a random number X. Then, the server replies an SYN-ACK to inform the client that connection request is received. The acknowledgment number is set to X+1, and the sequence number is chosen by the server randomly, i.e., Y. Finally, the client sends back an ACK to the server with a sequence number of X+1 and an acknowledgment number of Y+1. The three-way handshake is to make sure both sides of a connection are willing to connect and agree on the connection parameters. The connection termination uses a similar four-way handshake, which will not be covered in details in this guide.

In the TCP three-way handshake process, when the server receives the initial SYN, it will store the pending connection information in its memory. This is called a half-open connection because only the server side confirms the connection. We can easily create half-open connections by sending SYN to the server without replying to the received SYN-ACK. As there is only finite size of a queue in the memory to store TCP connection information, we can overflow it by intentionally creating many half-open connections in a short period. Once there is no more space for storing connection information, the server

will stop accepting new connections, eventually block other legitimate users from connecting to the server.

This is called TCP SYN flooding attack, which is a form of denial-of-service (DoS) attack. The process is illustrated in Figure 5. Although after a certain time the half-open connection data will be removed from memory, the attacker can always send SYN faster that the victim can process, thus making this attack very efficient. The key to a successful attack is to spoof random source IP addresses in the SYN packets so that this TCP traffic seems to come from legitimate users. Otherwise, the firewall may block our attack if too many SYN packets are sent from the same source. Also, if the host with a source IP address exists and it receives the SYN-ACK from the server, it will reply with an RST, which remove the half-open connection from the server's memory. Thus, the randomness of the source IP address is very important.

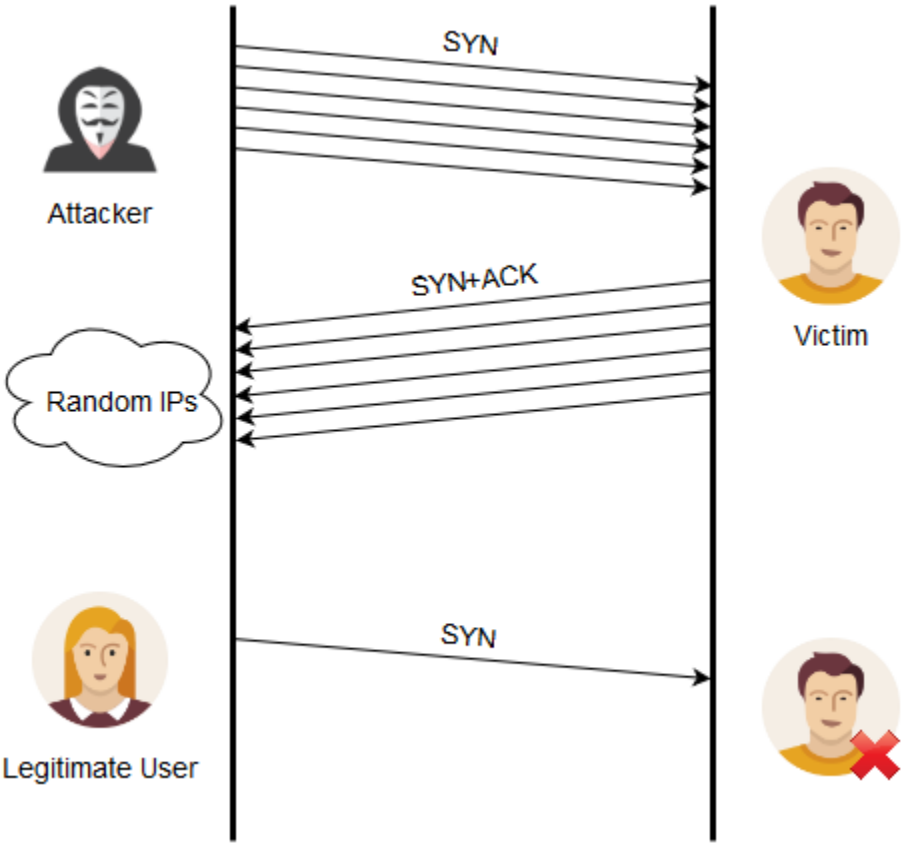

Figure 5 TCP SYN flooding attack process

To conduct a TCP SYN attack, you can use tools like hping3. hping3 is a command-line TCP/IP packet assembler and analyzer. It supports TCP, UDP, and raw IP protocols, and can be used for firewall testing, port scanning, TCP/IP attack auditing, etc. You can use the following command to start a TCP SYN flood attack:

$ sudo hping3 -d <packet_len> -S --flood --rand-source -p <target_port> <target_host>

The meanings of the options in the above command are listed below:

-c: number of packets to be sent
-d: packet data length in bytes

-S: set TCP SYN flag
--flood: send packets as fast as possible, hide incoming replies from the output
--rand-source: send out packets with random source IP addresses
-p: destination port

You need to specify a valid destination with an open port for this attack to work. For more details about hping3, please refer to the manpage:

$ man hping3
You can check all the TCP connections using "netstat -antp". This command will show the local and remote IP address and port of a connection, as well as the connection state and the program bound to this connection. The status field has a few possible states, including ESTABLISHED, LISTEN, and CLOSE. Please refer to the manpage of netstat for a complete description of these states.

Assignment

1. Here, we are going to capture the TCP traffic when you visit a website. Please start a traffic capturing using Wireshark with the following filter options:

   ip.src_host == 144.167.4.20 || ip.dst_host == 144.167.4.20

   and then make an HTTP request to the http://www.ualr.edu (not http://ualr.edu) using curl:

   $ curl www.ualr.edu

   Looking at your capturing result, what is the data length of the TCP packets during the three-way handshake? How many TCP packets do you capture (including HTTP packets)?

2. Log in to the attacker machine, launch a TCP SYN flooding attack on the victim's port 22. Use netstat tool check the connection status on the victim before and during the attack. What do you observe? To see if your attack works, try to SSH into the victim from the observer machine. Can you connect to the victim? If you are connected to the victim before the attack begins, will you be disconnected when the attack is going on?

3. There are several countermeasures for defending against TCP SYN flooding attack. One is called SYN cookie. It uses a specially crafted sequence number in the SYN+ACK and discards SYN queue entry in the three-way handshake. This defense is on by default. You can check your current settings using

   $ sudo sysctl -n net.ipv4.tcp_syncookies

   and turn it on by

   $ sudo sysctl -w net.ipv4.tcp_syncookies=1

3. Turn on SYN cookie and see if your attack still works. Please briefly explain why SYN cookie can protect the system from SYN flooding attack.

Complete all the tasks and save your answer (with screenshots) to each of the tasks into a PDF file. Submit the PDF file.