Memo TCP SYN Flood

Vinci Nicolò

22 October 2021

1 TCP SYN flood

A TCP SYN flood attack consists on corrupting the TCP three-way handshake. In the TCP three-way handshake, the client first sends a SYN packet to the server. The server receives it and responds with a SYN ACK. Moreover, it allocates in memory a Transmission Control Block for the connection. At the end, the client sends an ACK to complete the TCP three-way handshake. However, if the client does not send the last ACK, the TCP connection is half-opened and the server has allocated a useless TCB in memory. So, if an attacker sends a lot of SYN packets, the server will allocate a lot of TCBs in memory wasting resources. At a certain point, this leads to a Denial Of Service, because the server ends the memory allocating a lot of TCBs. The attacker can send a huge amount of SYN packet without spoofing his IP address. The server will respond to the attacker with a huge amount of SYN ACK and the throughput can be insufficient before the server exhausts its memory. So, the attacker can spoof his IP address. In this way, the server sends the huge amount of SYN ACK packet to a fake IP address. However, if there is a legitimate machine at the fake IP address, it will responds with RST packet for each SYN ACK packet received. The server frees the TCB in memory when it receives a RST packet. To sum up, the fake IP address should be not respond to SYN ACK packets.

2 TCP SYN cookies

The main problem is that the server allocates immediately a TCB in memory even if the three-handshake is not completed. The server can use TCP SYN cookies to wait the three-handshake completion for allocating resources in memory. When the client sends a SYN to the server to start the handshake, the server will respond with a SYN ACK crafting a special sequence number called SYN cookie. Then, the client sends back an ACK packet with an Acknowledgement number SYN cookie + 1. Now, the server can substract 1 from the Acknowledgement number and check the SYN cookie. If it is correct the server will allocate resources in memory for that specific TCP connection Otherwise, it will drop the packet without allocating resources. However, the server has to craft the SYN cookie that can be computationally expensive. Moreover, SYN cookies does not reduce traffic and an attacker can still saturate the bandwidth sending a lot of TCP SYN packets.

3 Legitimate client script

The legitimate client script is called *leg_traffic.sh* and it can be found in the folder *scripts/traffic/*. It sends a *get* request in background with curl in an infinity while loop. It can be launched in this way:

```
$ ./scripts/traffic/leg_traffic.sh
```

4 Attack command

The attack command with flooder is embedded in the *attack.sh* script placed in the folder *scripts/traffic/*. The user can choose to launch flooder spoofing the attacker IP address or not.

```
$ ./scripts/traffic/attack.sh <spoof, nospoof>
```

The flooder is launched with the server IP destination 5.6.7.8 at port 80 with an highrate of 200 packets per second.

5 Results

First of all, the task is performed disabled SYN cookies on the server. The resulting graph is reported at figure 1.

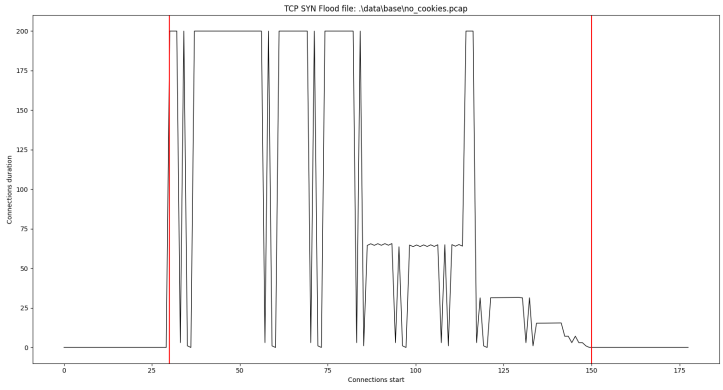

Figure 1: Task with no SYN cookies

The graph reports the TCP client connections towards the server. On x-axis there is when the connection starts and on y-axis the connection duration. If a TCP client connection is not be able to conclude, it has a duration of 200 seconds. The attacker sends a lot of SYN packets between the two red lines. Indeed, a lot of TCP client connections are not be able to conclude between 30

seconds and 150 seconds. The server was not able to process all the incoming requests from the client, because the attacker was overloading the server with a huge amount of TCP SYN packets. Hence, the server allocated resources in memory for a lot of half-opened connections.
Then, the experiment has been repeated with the SYN cookies enabled on the server.

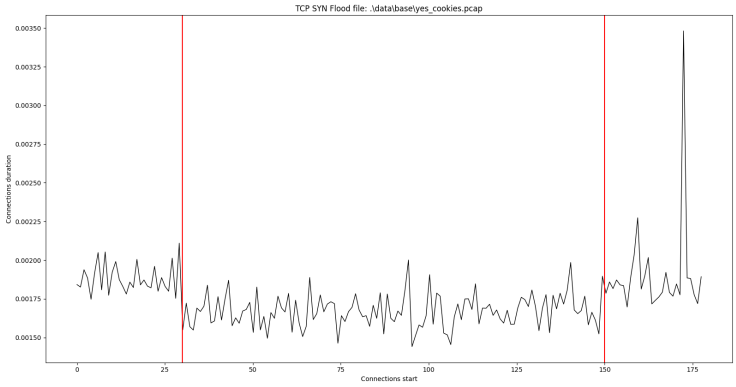

Figure 2: Task with SYN cookies

Now, the server was able to satisfy any incoming TCP client connection. Indeed, no connection has 200 seconds as duration. This because the server allocates memory resources with SYN cookies only if the attacker completes the TCP three-way handshake.

6 Extra 1

The experiment has been repeated without SYN cookies and without spoofing the attacker IP address. The graph at 3 is similar to 2.

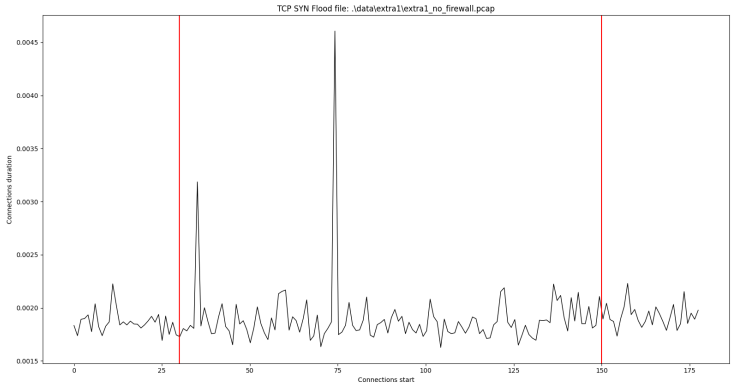

Figure 3: Extra 1 with no firewall rule

So, the client is able to complete any TCP connection towards the server. When the attacker sends a TCP SYN with the flooder tool without spoofing the IP, the server responds with a SYN ACK packet to him. Then, the attacker replies with a RST packet, because it did not recognize that the first SYN packet has been sent by him with the flooder command. Hence, the server receives the RST packet and frees the memory. However, the attacker can again perform a TCP SYN flood attack dropping any incoming SYN ACK packet from the server. An iptables rule has been designed for this purpose.

```
sudo iptables -I INPUT -s 5.6.7.8 -p tcp \
--tcp-flags URG,SYN,PSH,RST,FIN SYN -j DROP
```

The attacker will drop any incoming TCP packet with the SYN flag to 1 and with flags URG,SYN,PSH,RST,FIN set to 0. So, it is able to drop a TCP packet with both SYN and ACK flag set to 1. Indeed, the resulting graph is similar to graph 1 repeating the experiment setting the rule to the attacker.

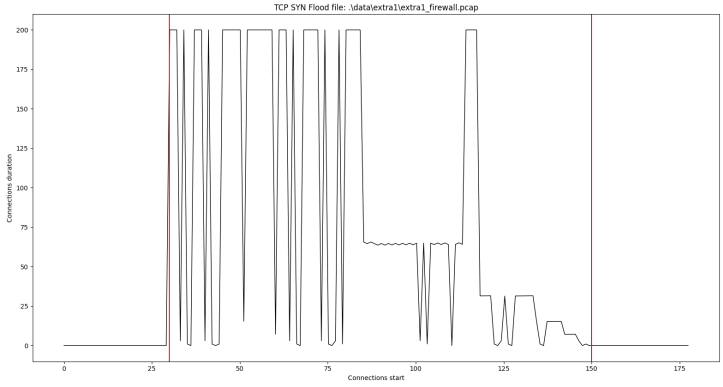

Figure 4: Extra 1 with firewall rule on attacker

A lot of TCP client connections end with 200 seconds, it means that the attacker is able to perform a Denial Of Service again.

7 Extra 2

The network topology has been changed for this extra task. The router has been removed and there are only two point-to-point connections. One between server and client and the other between server and attacker. The new NSfile has the following line:

```
set link1 [$ns duplex-link $server $client 1000Mb 0ms DropTail]
set link2 [$ns duplex-link $server $attacker 1000Mb 0ms DropTail]

tb-set-ip-link $server $link1 11.12.13.1
tb-set-ip-link $client $link1 11.12.13.2

tb-set-ip-link $server $link2 22.23.24.1
tb-set-ip-link $attacker $link2 22.23.24.2
```

The experiment has been repeated with SYN cookies disabled and without spoofing the attacker IP address.

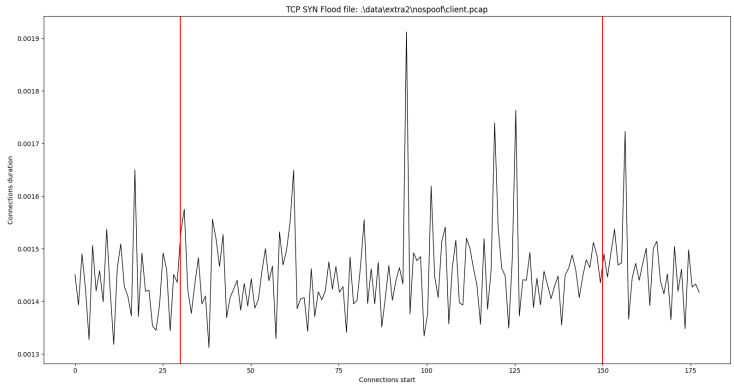

Figure 5: Extra 2 without spoofing

The client is able to end in a proper way any TCP connection, because the situation is similar to the section 6 with no firewall rule. The attacker sends a SYN packet, the server is able to respond with SYN ACK packet and the attacker replies with a RST packet. The server can respond to the attacker because they are on the same point-to-point link. The experiment has been

also repeated with SYN cookies disabled and spoofing the attacker IP address. Now, some TCP client connections are not be able to conclude in a proper way. It looks like that the attacker is able to perform the TCP SYN flood attack. Looking at the traffic on the server interface towards the attacker, the server is not be able to reach the attacker spoofed IP address. However, the server responds with SYN ACK packets in broadcast opening the TCP connection halfway and allocating resources in memory.

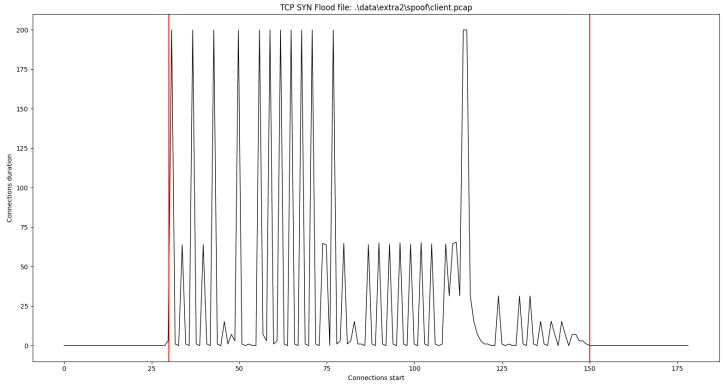

Figure 6: Extra 2 with spoofing