Lab - TCP SYN FLOOD ATTACK

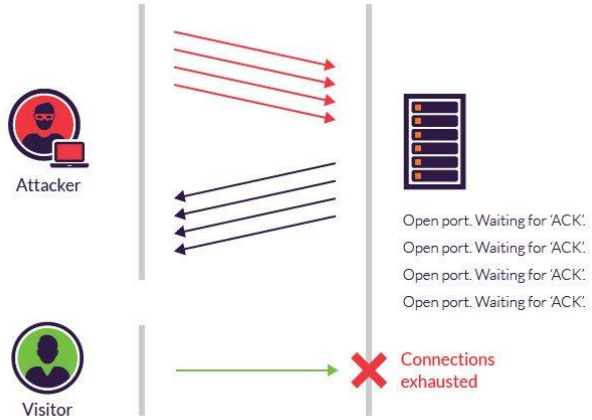
ATTACK DESCRIPTION

TCP SYN flood (a.k.a. SYN flood) is a type of Distributed Denial of Service (DDoS) attack that exploits part of the normal TCP three-way handshake to consume resources on the targeted server and render it unresponsive. Essentially, with SYN flood DDoS, the offender sends TCP connection requests faster than the targeted machine can process them, causing network saturation. When a client and server establish a normal TCP "three-way handshake", the exchange looks like this:

- Client requests connection by sending SYN (synchronize) message to the server.
- Server acknowledges by sending SYN-ACK (synchronize-acknowledge) message back to the client.
- Client responds with an ACK (acknowledge) message, and the connection is established.

In a SYN flood attack, the attacker sends repeated SYN packets to every port on the targeted server, often using a fake IP address. The server, unaware of the attack, receives multiple, apparently legitimate requests to establish communication. It responds to each attempt with a SYN-ACK packet from each open port.

The malicious client either does not send the expected ACK, or—if the IP address is spoofed—never receives the SYN-ACK in the first place. Either way, the server under attack will wait for acknowledgement of its SYN-ACK packet for some time.



Progression of a SYN flood

During this time, the server cannot close down the connection by sending an RST packet, and the connection stays open. Before the connection can time out, another SYN packet will arrive. This leaves an increasingly large number of connections half-open – and indeed SYN flood attacks are also referred to as "half-open" attacks. Eventually, as the server's connection overflow tables fill, service to legitimate clients will be denied, and the server may even malfunction or crash. While the "classic" SYN flood described above tries to exhaust network ports, SYN packets can also be used in DDoS attacks that try to congest your pipes with fake packets to achieve network saturation.

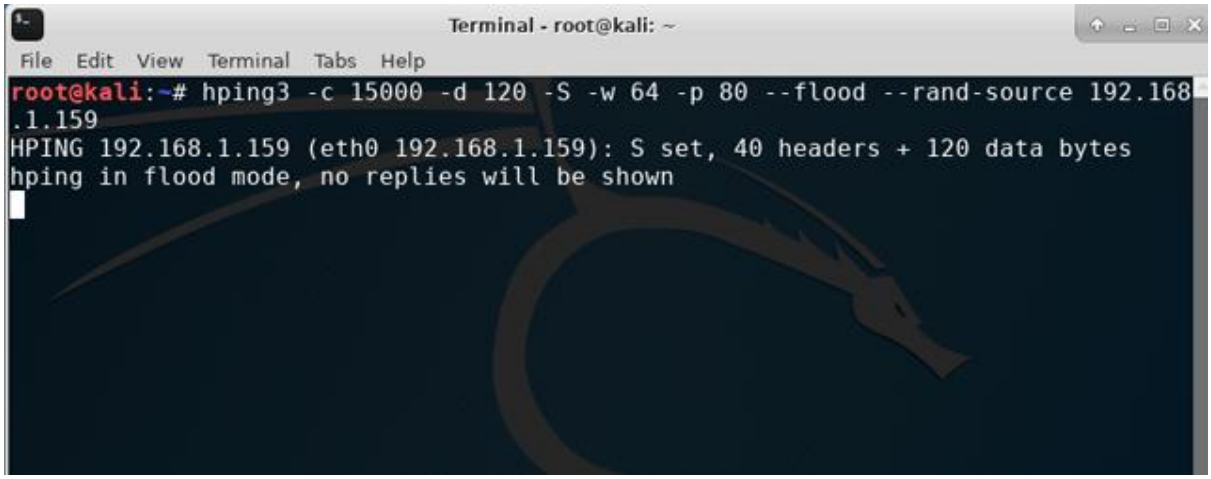
How to perform TCP SYN flood DoS attack & detect it with Wireshark, Ubuntu/Kali Linux and Hping3?

To test if you can detect this type of a DoS attack, you must be able to perform one. The simplest way is via a Kali Linux and more specifically the hping3, a popular TCP penetration testing tool included in Kali Linux. Alternatively, Linux users can install **hping3** in their existing Linux distribution using the command:

```
sudo apt-get install hping3
```

In most cases, attackers will use **hping** or another tool to spoof IP random addresses, so that's what we're going to focus on. The line below lets us start and **direct the SYN flood attack** to our target (192.168.1.159):

```
sudo hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source 192.168.1.159
```



Let's explain in detail the above command:

hping3	Name of the application binary
-c 15000	Number of packets to send
-d 120	Size of each packet that was sent to target machine
-S	I am sending SYN packets only
-w 64	TCP window size
-p 80	Destination port (21 being FTP port). You can use any port here.
--flood	Sending packets as fast as possible, without taking care to show incoming replies. Flood mode.
--rand-source	Using Random Source IP Addresses. You can also use -a or –spoof to hide hostnames.
192.168.1.159	Destination IP address or target machines IP address. You can also use a website name here.

So how do you know it's working? In hping3 flood mode, we don't check replies received (actually you can't because in this command we've used **--rand-source flag** which means the source IP address is not yours anymore). Took me just 5 minutes to completely make these machines unresponsive (that's the definition of DoS – Denial of Service). In short, if this machine was a Web server, it wouldn't be able to respond to any new connections and even if it could, it would be really slow.

Experiment

Objective:

To perform a TCP SYN flood DoS attack with Ubuntu/Kali Linux (hping3) and use the Wireshark network protocol analyser filters to detect it.

Create a web page with N (e.g. 10) embedded images. Each image should be of minimum 2 MB size. Note down the time taken to display the entire page before DoS attack. Perform DoS attack using hping3 command. Make a web request again and ensure that the cache is cleared before starting the request. Explain the response time differences.

Step 1: Store images in the server path

A html page consisting of 10 images having size > 2MB were placed and accessed by the client. This html page should be stored in the location: /var/www/html/filename.html

Step 2: Prepare web page as html file to add 10 images

Step 3: Access web page on client side

The client could access the file as: 172.16.10.1/filename.html where --> 172.16.10.1 is server's IP address

Note:

1. The wireshark should capture the packets between the client and the server while the file is accessed.
2. The images in the HTML page should have all the permissions specified through the server for the proper access.

Step 5: Use wireshark

Open wireshark in the server computer while client is trying to access the server's local host webpage. Apply http filter and note the time to capture all the 10 images.

Additional recommendations:

UDP Flood

UDP is a protocol which does not need to create a session between two devices. In other words, no handshake process required. A UDP flood does not exploit any vulnerability. The aim of UDP floods is simply creating and sending large amount of UDP datagrams from spoofed IP's to the target server. When a server receives this type of traffic, it is unable to process every request and it consumes its bandwidth with sending ICMP "destination unreachable" packets. **hping3** can be used for creating UDP floods:

```
hping3 --flood --rand-source --udp -p TARGET_PORT TARGET_IP
```

TCP FIN Flood

A TCP packet with FIN flag enabled is only accepted when a client established a TCP connection with a server. Otherwise, packets will be simply dropped. If the attacker just floods server without establishing TCP connections, FIN packets will be dropped as expected. But the server still requires some resources to process each package to see if the package is redundant. These types of attacks are easy to execute because it is just generating junk FIN packets and sending them. To perform FIN floods, **hping3** can be used:

```
hping3 --flood --rand-source -F -p TARGET_PORT TARGET_IP
```

PUSH and ACK Flood

By flooding a server with a bunch of PUSH and ACK packets, the attacker can prevent the server from responding to the legitimate requests. In order to perform PSH+ACK attack you can use **hping3** with this parameter.

```
hping3 --flood --rand-source -PA -p TARGET_PORT TARGET_IP
```