# Benefits and Drawbacks of SYN Cookies in Linux Kernel

**Ali Ghaffarian**

**2024, December**

# Contents

# 1. Abstract

For a TCP connection to be established, a three way handshake must be done, SYN floods target a state of this process in which the server waits for the client to respond. Handling handshakes by sending SYN cookies is one of the most used defenses against SYN floods, which enables us to accept connections statelessly, with no one other than the server itself noticing the use of SYN cookies.

In this research i tried to determine if SYN floods are still viable in the absence of any direct defenses and if not, how the SYN cookie approach impacts our system in various ways.

## 2. Introduction

In the following sections, the essential concepts are presented that are needed to understand how SYN cookies work and why they are needed.

### 2-1. TCP

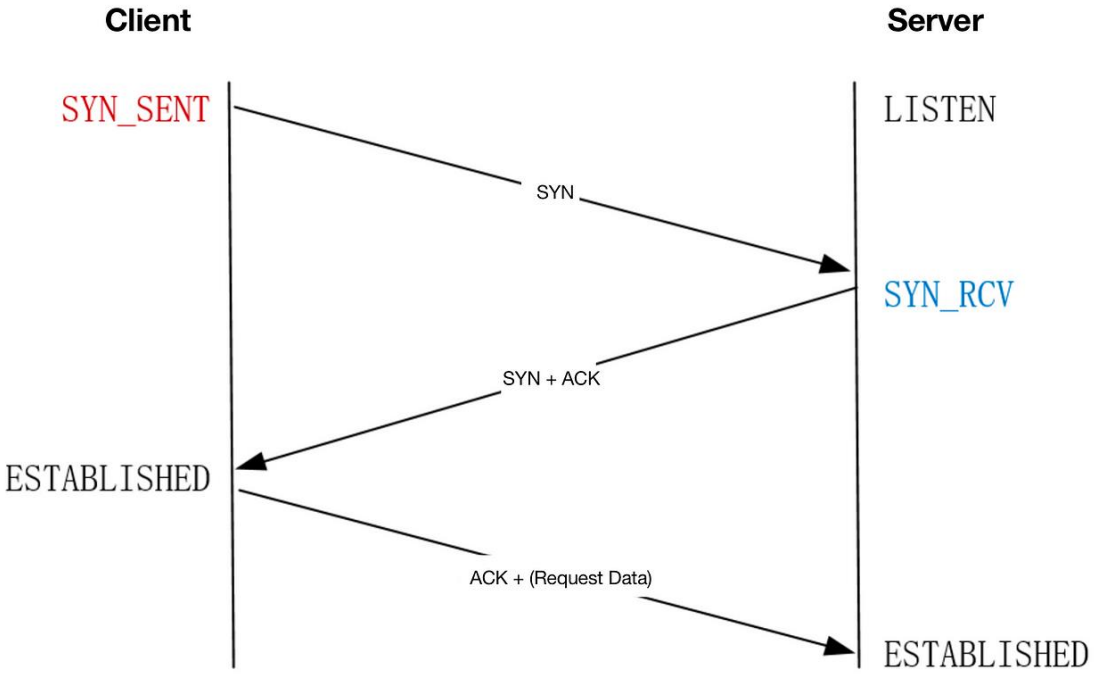
Transmission Control Protocol is the most used reliable data transfer protocol in TCP/IP networks and is considered a complex one.

<table>
<tr><td colspan="5">Source Port</td><td colspan="5">Destination Port</td></tr>
<tr><td colspan="10">Sequence Number</td></tr>
<tr><td colspan="10">Acknowledgement Number (meaningful when ACK bit set)</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">Window</td></tr>
<tr><td colspan="5">Checksum</td><td colspan="5">Urgent Pointer (meaningful when URG bit set)</td></tr>
<tr><td colspan="10">(Options) If present, Data Offset will be greater than 5.<br>Padded with zeroes to a multiple of 32 bits, since Data Offset counts words of 4 octets.</td></tr>
<tr><td colspan="10">Data</td></tr>
</table>

TCP fields[1]

## 2-2. Three Way Handshake

The two ends of the TCP connection must know each other's initial sequence number before the actual reliable data transfer can begin, one of the responsibilities of a TCP three way handshake is exactly this.



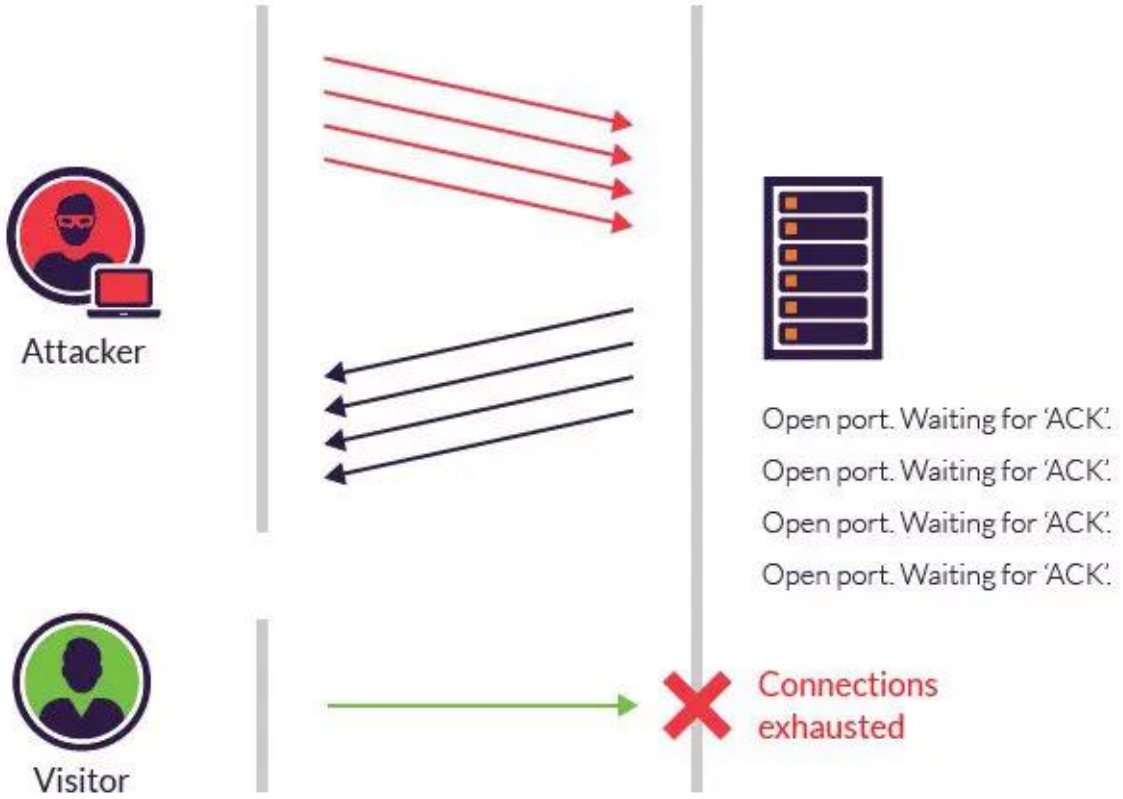

three way handshake[2]

## 2-3. SYN Floods

TCP SYN flooding is a DoS attack that leverages the reserved resources for pending connections. Because TCP servers normally reserve such resources only when a connection is initiated, the attacker can flood the server with SYN requests that include spoofed source IP addresses.

This way the backlog of the TCP server will eventually fill up, making the server always waiting for the non-existent clients to complete their connections.

When using the server's timeout and backlog as the attack vector, the cost of the attack drops dramatically, as a typical TCP server will wait several seconds until it determines that the client's not going to complete the connection.




SYN flood[3]

## 2-4. SYN Cookies

SYN floods were so successful because of the stateful nature of the TCP protocol. One way to stop SYN floods from being so cheap and effective is to handle the three-way handshake differently while preserving the rest of the protocol.

In other words, handle the handshake statelessly without breaking the protocol, that is, forgetting the client ever initiated a connection, but having a method to recover all connection's crucial data when the client completes the connection.

SYN cookies do this by carefully encoding the connection's data in the server's initial sequence number (ISN) when acknowledging the client's SYN request. This way when the client sends the final packet of the three-way handshake all of the required data to reconstruct the connection will be recoverable.

### TCP Crucial Fields

The data a TCP server needs to be able to reconstruct the connection includes:

- client's ISN
- server's ISN
- client's IP address
- server's IP address
- client's port number
- server's port number
- max segment size (MSS)

### Encoding TCP options

TCP is an old transport layer protocol, it was designed for networks of its time that tended to lose packets frequently and have lower bandwidth, in which devices reside with small buffers to dedicate to the connection, because of this certain TCP options are needed to be negotiated so that modern devices can use all of their offered load while using TCP as the transport layer protocol.

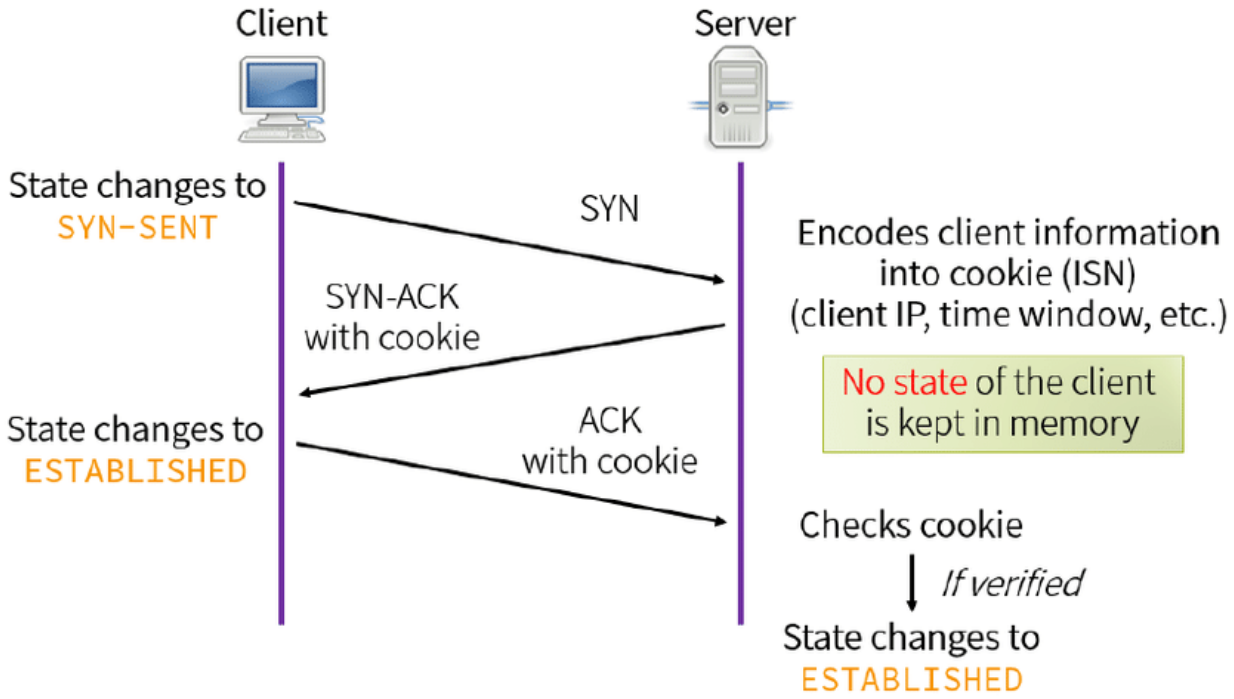
The most important of which includes:

- Window Scale
- Selective Acknowledgement (SACK)

**Window Scale:** each device advertises a window scale value, indicating each declared window size needs to be multiplied by 2^window scale.

**Selective Acknowledgement (SACK):** when using SACK as the loss recovery each device can declare the sequence number of the lost data and continue the data transfer without needing to slow down.

Linux encodes these two options in the timestamp field of the SYN-ACK packet only if both devices support timestamps [5].




SYN cookie[4]

## 3. Testing Environment

The role of the device will probably vary based on the experiment.

**Device A:**

```
OS: Arch Linux x86_64
Host: ASUS TUF Gaming F17 FX706HC_FX706HC 1.0
Kernel: 6.11.5-arch1-1
CPU: 11th Gen Intel i5-11400H (12) @ 4.500GHz
Memory: 15732MiB
Network controller: MEDIATEK Corp. MT7921 802.11ax PCI Express Wireless
Network Adapter
```

**Device B:**

```
OS: Android 13 aarch64
Host: POCO 2201116PG
Kernel: 5.4.259-qgki-gcb8e1baa6b45
CPU: (8) @ 1.804GHz
Memory: 5437MiB
Network controller: Not root on this device, can't check
```

**Device C:**

```
OS: Debian GNU/Linux 12 (bookworm) x86_64
Host: Aspire A315-58G V1.26
Kernel: 6.1.0-22-amd64
CPU: 11th Gen Intel i3-1115G4 (4) @ 4.100GHz
Memory: 11739MiB
Network controller: Intel Corporation Wi-Fi 6 AX201
```

# 4. Related Work Study

How to generate, issue, and verify SYN cookies is still an ongoing topic because of the tradeoffs of different implementations, some users might redefine how their host handles SYN cookies based on their needs [6].

## 4-1. Cookie generation in Linux Kernel

The encoded information (client's ISN and MSS) is combined with two cryptographically hashed values (like HASH (secret, saddr, sport, daddr, dport, secret)) secret keys and a count value that is incremented every minute to make the cookie and sent to the client [5][7].

Cookie validation is done by extracting the encoded MSS from the client's Acknowledge number and checking if it's out of range [7].

## 4-2. Minisocks

Linux stores half opened connections via the tcp_request_sock (also known as minisock[5]) structure which is 96 bytes compared to a regular tcp_sock which is a 1616 structure [5]. Considering how smaller the minisock is, the possibility of running out of memory during a SYN flood is extremely low.

## 4-3. Benefits of SYN Cookies

- No need to track any half-opened connections.
- Higher cost of DoS attacks that target the stateful nature of TCP three way handshake as the attack vector, like Syn floods.

## 4-4. Drawbacks of SYN Cookies

- Limited support for TCP options.
- SYN cookies are ineffective against SYN flood attacks that target the victim's bandwidth.
- In an effective connection spoofing scenario, firewalls focusing on filtering connections by SYN flag will be bypassed.
- Needs more CPU resources to be computed and validated.

### Packet Loss

On 3rd March 2022, Kevin Graham stated that because the server forgets what the client's ISN was it could lose data that was going to be arrived along with the final packet of the three way handshake [8]. Other operating systems might be suffering from the same

problem as well, but due to the nature of close source software, we can't state anything unless special experiments are done.

### Connection Spoofing Attacks

As a Linux TCP server in SYN cookie mode accepts connection after receiving a valid SYN cookie from the client, connections can be made by guessing SYN cookies. On 13th August 2013, Jakob Lell provided a method to guess a valid SYN cookie in around 8 minutes with a gigabit connection to the target [9].

# 5. Running the Experiment

When doing the actual experiments, I encountered so many difficulties that i was unable to count, like unexpected behavior of the device, losing some logs, and most importantly, lack of knowledge.

The events are presented in the sequence they happened, meaning a technical difficulty is from where it is introduced until the end of the experiment (or a fix is applied).

## 5-1. Availability of a TCP Server Under SYN Flood With SYN Cookies Disabled

I made an automated laboratory with three devices, each waiting for a connection to start their assigned tasks, and measured the server's response time with and without SYN cookies.

Role of each device is as follows:

- Device A (attacker):
    - Floods the Server with SYN Requests and logs the sending rate.
- Device B (client):
    - Access point of the wireless network.
    - Tries to reach the server every one second and log the response time.
- Device C (server):
    - Listens for a single TCP connection, after the connection is closed listens for connections again.

My implementation of SYN flood was inspired heavily by, Jakob Lell's approach[9], that is, to write the SYN requests on a packet capture file(pcap) instead of sending them, then utilize tcpreplay to send the packets at the desired rate, this way no computation time is required for each packet.

The type of physical network being wireless added complexity, and i had to disable any encryption (thus any pre-shared password) to be able to transmit the pcap as it was. To remove layer 2's complexity i actually sent the SYN requests on layer 3 and captured them using Wireshark.

To test if i had setup the SYN flood attack correctly, i replayed the generated pcap two times, the first and second replays went with 99 and 258 packets per second, which was not enough at all to have any impact on the target's resources (except for the connection backlog that depends on the user's configurations), but still due to the noise i injected into the network, my ssh session was closed and i needed to physically interact with the server.

**Server Availability, SYN Cookies Disabled**

While flooding the server with the rate of 151 SYN requests per second, the client device although being the access point of the network, failed to establish a TCP connection with the server, after i stopped the flood, the client finally managed to connect with the successfully (we assume that it was the backlog that was full, as the server remained unresponsive even when the SYN flood stopped for a while).

**Server Availability, SYN Cookies Enabled**

The server detected the SYN flood almost instantly, sending SYN cookies afterward.

The server was still unresponsive. i suspected that changing kernel parameters needed a reboot to take effect, but i highly doubted it, my suspicion was proven right after rebooting the server and connections could be made from client to server easily.

Since i didn't know how to get rid of my OS's fairness, i spawned 5 tcpreplay processes which replayed the same pcap to boost the sending rate and this approach resulted in 248 packets per second sending rate.

No difference in the response time of the servers was observed during the flood (sadly i lost the logs), the reason for this could be:

- Limitation of my network
- Fairness of OS
- My lack of knowledge

At this point, i needed to change the server's OS to Manjaro Linux due to sanctions preventing me from using the tools i needed to continue the experiments.

The server now running Manjaro Linux unexpectedly closed the port that was being flooded making me unable to reproduce the logs that i lost, and it was too late to change the OS for a second time.

### 5-2. Packet Loss of SYN cookies

Role of each device being the same as the section *Availability of a TCP Server Under SYN Flood with SYN Cookies Disabled*, I decreased the SYN flood rate to only make the TCP server to go into SYN cookie mode and then made a manual connection which initiates the three way handshake and tries to complete it with the wrong sequence number, to test whether the server can verify client's sequence number.

Unlike how is demonstrated in [7], the server can check if the client's sequence number is valid or not, as i investigated more, i realized this problem was patched by encoding the client's initial sequence number in the SYN cookie [6].

## 6. Conclusion

With the different technical difficulties i encountered, it was hard to be sure of anything, but in the experiments i've done, i observed that with the defenses against SYN floods being disabled, the server indeed became completely unresponsive (if enough bandwidth was available for the client to reach the server, which i'm not sure about, since i was limited to wireless networks, and i have no deep understanding of such). Packet loss in the Linux kernel while handling handshakes using SYN cookies is fixed by encoding the client's initial sequence number in the SYN cookie.

## 7. References

[1] Transmission Control Protocol, Wikipedia

[2] ByteCook, How To Improve the Performance of TCP Three-Way Handshake, Dev Genius

[3] What is a TCP SYN Flood, Imperva

[4] Miroslav Ďulík, Mechanism of SYN cookies, Download Scientific Diagram

[5] Patrick McManus, Improving syncookies, LWN.net, April 9, 2008

[6] Kuniyuki Iwashima , Linux Plumbers Conference : SYN Proxy at Scale with BPF , November 15, 2023

[7] https://github.com/torvalds/linux/blob/master/net/ipv4/syncookies.c

[8] Kevin Graham, SYN cookies ate my dog, breaking TCP on Linux, WP Bolt, March 3, 2022

[9] Jakob Lell, Quick Blind TCP Connection Spoofing with SYN Cookies, Jakob Lell's Blog, August 13, 2013

## 8. Attachments

Source codes, proposal, etc.:
https://github.com/AliGhaffarian/university_thingies/tree/main/research_and_presentation