Investigating TCP SYN Flood Mitigation Techniques in the Wild

Julian Villing
Technical University of Munich, Germany
Email: julian.villing@tum.de

***Abstract*—TCP SYN flood Denial-of-Service (DoS) attacks exploit a weakness in the TCP specification. By initiating many incomplete connections, the servers' backlog is filled with the states of the half-open ones. Once there is no more space in the backlog, the server is unable to handle legitimate requests and the attack has been successful. It is important to prevent this problem as the targeted machine cannot offer its services anymore due to being unreachable. This paper introduces and compares several TCP SYN flood mitigation techniques as well as discussing challenges of their detection.**

***Index Terms*—tcp syn flood, syn cookies, syn authentication, syn flood mitigation**

1. Introduction

When establishing a new TCP connection, the server stores the corresponding state in a transmission control block (TCB) which is needed to establish the connection. The minimum size for said block is 280 bytes whereas Linux uses 1300 bytes per block. A new TCB is created and stored in the backlog for each TCP SYN packet received. This information persists during the three way handshake until the connection is either established or a timeout occures as explained by Eddy in [1].

TCP SYN flood attacks exploit this weakness: they initiate many half-open connections in a short time but never complete the handshake. The flood of connections forces the server to keep many unnecessary TCBs for those connections in the backlog which is eventually filled. Upon exhaustion, which is reached with 100 to 1000 connections, no new connections can be established to handle legitimate requests and the DoS attack has been successful [1].

Mitigating this problem is important since not being able to handle such floods can for instance lead to outages. As an example, starting in October 2012, a large US bank was a target of such a flood but unable to mitigate the attacks for more than 5 months. This caused the bank's online service to be unusable every now and then according to [10]. It is desirable that servers under attack are still capable of delivering their services. Mitigation techniques are designed for this usecase, aiming to protect the server from being unreachable.

The rest of the paper is structured as follows in Section 2, the TCP three way handshake including its weakness is explained. Different countermeasures with their advantages and disadvantes are introduced in Section 3. Section 4 covers why it is difficult to detect which of those techniques is currently in use. In the conclusion these techniques get compared in terms of memory and computing immunity as well as effectivity.

2. TCP Three Way Handshake

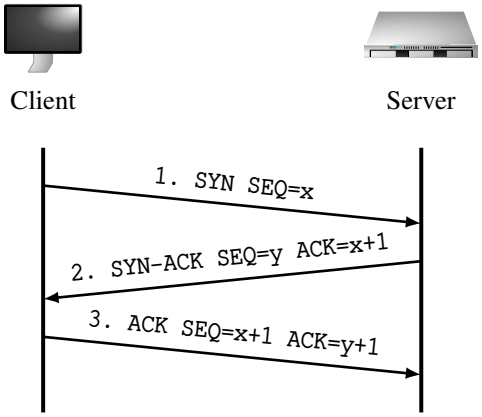

Figure 1: TCP Handshake

As seen in Figure 1, a TCP handshake consists of three packets being exchanged between the client and the server. Those are used to inform both endpoints about each other's sequence number and the corresponding acknowledgments.

When the server receives the SYN packet, a TCB is created and stored in the backlog, primarily to keep track of the options selected / requested in the header. It contains, among others, the source / destination addresses and ports as well as the sequence numbers received from and sent to the client [1].

Once the handshake is completed, the TCB is removed from the backlog and does not put stress on the bottleneck anymore. Therefore the limited resource is consumed starting with the reception of the first packet until the third packet is received or a timeout occured [1].

In case of a SYN flood, the second packet is ignored by the attacker and therefore the third packet will never be sent. As a result the TCBs use up memory until the timeouts occur [1].

3. Mitigation Techniques

This section describes and compares the advantages and disadvantages of different approaches used to decrease the vulnerability by the SYN flooding attack.

3.1. Filtering

When a router receives a packet, it verifies that the source address specified in the received packet is actually

doi: 10.2313/NET-2019-06-1_14

reachable. A reasonable check to use is strict reverse path forwarding. This method only forwards packets "if the packet is received on the [network] interface which would be used to forward [...] traffic to [the packets' source address]" as explained by Baker and Savola in [9]. The dropped packet must have a spoofed source IP and did not come from a legitimate host in the network [1] as shown by Salunkhe et al. in [3].

Advantages: Sending spoofed packets is much more difficult and no changes to TCP are necessary [1].

Disadvantages: Filtering only works for spoofed IP addresses which are not reachable and is therefore ineffective against an attacker which controls multiple hosts. Global deployment of filters is also neither guaranteed nor likely [1].

3.2. Increased Backlog

This technique solves the problem that the backlog is filled too quickly by simplyincreasing its size. The obvious result is that more connections can be stored [1] [3].

Advantages: More connections can be stored, therefore filling the backlog takes a bit longer.

Disadvantages: The backlogs' implementation is not designed to scale, e. g. the search algorithm used is not trimmed for efficiency. It can also be circumvented simply by increasing the attack rate [1].

3.3. Reduced SYN-RECEIVED Timer

Instead of increasing the backlog, this approach reduces the duration for which the connections can occupy the backlog by reducing the SYN-received timer [1] [3].

Advantages: Incomplete connection attempts get removed earlier from the backlog.

Disadvantages: If the process of establishing a connection takes longer than normal, e. g. due to a slow network connection, legitimate connections might not get established [3]. It is equally ineffective for the same reason: an increased attack rate can easily make up for the lower timeout [1].

3.4. Recycling the Oldest Half-Open TCB

This technique mitigates the attack by recycling the oldest half-open connection once the backlog is exhausted [1].

Advantages: It works if the time to fully establish a connection is lower than the time needed to fill the backlog [1]. This means that the attack rate must be low or the backlog must be big enough.

Disadvantages: The approach fails if the backlog gets filled too quickly [1] since slow connections can be evicted.

3.5. SYN Cache

Each new connection is stored in a minimized TCB which in turn is stored in a hashmap with a limited bucket size. The bucket in which to store the TCB is selected by hashing the IP adresses, ports and secret bits from the header which the server chose beforehand and the oldest entry is dropped if the bucket is full. Generating the hash from secret bits prevents malicious clients from overflowing a specific bucket and therefore dropping legitimate connections [1] [3]. It was "the most effective and the most used" technique back in 2008 according to Oncioiu and Simion in [8].

Advantages: The secret bits prevent attackers from overflowing buckets and dropping legitimate connections [1].

Disadvantages: Because the complete TCB is not stored, some information is left out and must be retransferred once the connection gets established [1].

3.6. SYN Cookies

As the value of the initial sequence number (ISN) used in the handshake can be chosen at random, the server can give this number a special meaning e. g. by encoding data.

SYN Cookies use this very number to encode the state which would otherwise be stored as a TCB in the backlog and therefore prevent the latter from filling up. The value consists of three parts which get concatenated: the slowly increasing timestamp the server keeps track of, the maximum segment size as well as a hash of the client's ISN as well as the source / destination address and port. When the client's acknowledgement is received, the server substracts one from the acknowledgement number and compares it to the encoded state using the last few timestamps. If the encoded states match, it is a legitimate client and the TCP handshake is completed successfully. The use of a timestamp prevents replaying the packet at a later time. A TCB is created using the state and directly stored as an established connection, therefore never allocating resources in the backlog [1] [3] as presented by Lemon in [5], Ricciulli et al. in [6] and Liu and Sheng in [7].

Advantages: No memory is consumed to store the state because it is encoded in the server's initial sequence number instead [1].

Disadvantages: The sequence number is smaller than the TCB, therefore, not the whole state can be stored and data retransmission may be necessary. The SYN-ACK cannot be resent because the state is not stored on the server. This behavior breaks the TCP semantics. This technique is only effective in a low degree SYN flood attack as space is traded for processing time [1].

3.7. TCP SYN Authentication

Legitimate clients can be identified if they follow the TCP specification unlike attackers who just flood the server with SYN packets.

SYN Authentication uses a special mitigation device between the client and the server. If a client wants to establish a connection to the server, it actually does the TCP handshake with that device. When the intermediate device receives a SYN packet, it responds with a SYN+ACK packet containing an invalid acknowledgment number. The client has to respond with a RST packet as defined in the TCP specification. If it does, the client is authenticated beause attackers do not handle

doi: 10.2313/NET-2019-06-1_14

Packet	Counter	Action	Pass?
SYN	None	Move to C-1	×
SYN	C-1	Move to C-3	✓
SYN	C-2	None	✓
SYN	C-3	Add to C-3	○
ACK	None	None	×
ACK	C-1	None	×
ACK	C-2	None	✓
ACK	C-3	Move to C-2	✓

✓ (pass the packet), × (drop the packet), ○ (pass is less likely)

TABLE 1: Handling of SYN and ACK packets

invalid packets and may not even receive it, e. g. if they use spoofed IP addresses. Since the client is now authenticated, the mitigation device allows direct communication between the client and the server as shown by Nagai et al. in [2]. In other flavors the server may send a reset and also track the hop count.

Advantages: The server does not allocate resources for incomplete connections as the mitigation device ensures that the connection is legitimate [2].

Disadvantages: The client needs to do the handshake twice [2]. This slightly increases the connection establishment time in times where RTTs should be low.

3.8. Three Counters

Attackers flood the server with many SYN packets without responding which can be distinguished from legitimate requests. Because the latter follow the TCP specification, they can resend packets which were lost and correctly reply to the server's responses.

This technique makes use of three counters, C-1 to C-3, in which different packets will be stored. The first counter records initial SYN packets, the second stores SYN packets of established connections, and the last records any other SYN packets. These counters are typically used after a flood has been detected. Their usage can be seen in Table 1 and is explained in the following paragraphs [4].

A 4-tuple consisting of the source / destination addresses and ports is extracted from each received SYN packet and queried against the three C-s. If it is not found, the tuple resembles a new connection and is added to C-1 while the packet is being dropped. If it is in C-1 or C-2 the packet is passed and the tuple is moved to C-3 in the first case. Otherwise the packet must be in C-3 and it is forwarded with a propability p, decreasing for an increasing number of packets received. This is achieved by adding the tuple to C-3 multiple times and querying the counter for the total amount [4].

Received ACK packets are handled in a similar manner using the same 4-tuple. If the packet is in C-2 or C-3, the packet is passed and moved to C-2 if not done yet because the connection is now completed. In any other case, the packet will be dropped because a SYN packet must be received before an ACK as explained by Gavaskar et al. in [4].

Advantages: This technique is effective against many identical packets as they are less likely to be passed the more often they are received.

Disadvantages: Every SYN packet has to be sent twice. Duplicating every SYN packet is a problem because the latter will be forwarded to the server for sure. Furthermore following the two SYNs with an ACK, without the need to actually listen for a SYN-ACK response, renders this countermeasure ineffective while not affecting other approaches because the sequence numbers are not tracked. Lastly, flooding the server with a spoofed SYN packet, essentially preventing that very packet from being forwarded can cause problems if a legitimate client attempts to connect using the same SYN packet.

3.9. Random drop

Similar to technique 3.4, a connection is dropped once the backlog is full. The difference is that the connection chosen at random and the client is informed with a TCP Reset (RST) [6].

Advantages: As most of the entries in the backlog are from the attacker, this mitigation technique has a high chance to drop the malicious connections [6].

Disadvantages: There is a small propability that legitimate connection attempts get denied [6].

3.10. SYN Agent

Instead of doing the handshake with the server, the client does that with the SYN agent instead. After the handshake is completed, there are two ways to continue depending on the kind of agent.

The first option is that the agent does the handshake with the server imitating the client. Once this is completed, the difference between the agent's and the server's sequence number has to be remembered. It is applied to each message the agent receives from either side and modified before it is forwarded.

The other option is that the agent informs the server about the successful handshake by sending an ACK packet to the server with the reserved bit set to one. This includes the sequence number the agent used while establishing the initial connection. The advantage of this method is that the agent just forwards the messages without touching them. As a result, the agent does not need to store the difference and the computational effort is also decreased [7].

Advantages: The server is guaranteed to only get to know serious connection attempts. The latter option also reduces the load on the agent [7].

Disadvantages: An extra agent is needed which has to store information about half-open connections [7]. If the first option is chosen, the agent is also given additional computational effort.

4. Detection

The detection and identification of the mitigation techniques in use is difficult because they are only active

doi: 10.2313/NET-2019-06-1_14

while the server is being flooded with excessive SYN packets. Some countermeasures are hard to identifiy from the outside even if they are currently active. To make if even more difficult, not every technique has an unique measurable outcome.

Sending SYN packets alone does not yield any information about the protection mechanisms used because the necessary information is contained within the received SYN-ACK packet. A SYN-ACK which is not sent yields additional information as well. For these information to be collected a legitimate client or tool is required which actually listens for those responses.

An **increased backlog** or the **reduced timeout** are practically undetectable because they are essentially configuration options which could also just be configured large or short respectively. Additionally, these measures simply delay the point of exhaustion by allowing more half-open connections to be stored at the same time or removing them earlier.

Recycling is mostly undetectable as the oldest half-open connection is resetted. The attack rate must be high enough for this technique to get noticable by preventing legitimate connections from being established.

Besides being quite similar, **random drop** is even harder to detect because the amount of dropped connections is equal however the legitimate clients have a decreased chance to be chosen. This technique is therefore even more difficult to be identified.

Filtering is partly detectable because SYN packets with spoofed addresses do not get forwarded and because of that no SYK-ACKs for those can be captured.

Caching is difficult to detect as it can be identified if information needs to be retransmitted. This technique should not be identifiable by dropped connections as the probability of a real one being reset is rather low.

SYN Cookies are not detectable since the initial sequence number is a hash value.

SYN Authentication is identifiable by the first SYN-ACK which is always invalid.

Three counters are detectable as the first SYN will always be dropped for a new connection. In addition sending many packets in the name of a valid client might block it from connecting.

The **agent** cannot be identified as it duplicates and imitates the server.

5. Conclusion

In order to prevent exploitation of the weakness in the TCP specification, many TCP SYN flood mitigation techniques have been developed. They are summarized in Table 2.

Some techniques like increasing the backlog or reducing the timeout focus on delaying the exhaustion of the backlog while others try to prevent it from getting full, like SYN agent or SYN cookies. Additionally, recycling and random drop take no precautions to prevent the backlog from being filled, instead, they simply drop a connection from the backlog if it is full based on the respective heuristic algorithm.

SYN Cookies are the best choice if the migitation technique must be ready for immediate use because this method is included in Linux. They can therefore be used even if no countermeasures have been taken beforehand. A SYN Agent should be used when the server must be protected from flooding attacks under all circumstances however a separate machine with enough memory handling the incoming traffic is needed. SYN Authentication can be used instead of SYN Cookies if correctly following the TCP specification is more important than the response time as the intermediate device can resent the SYN-ACK packets however the handshake has to be done twice.

Technique	Guarantee	Memory Immunity	Computing Immunity	Robustness	Good Performance
Filtering	○	✓	×	✓	×
Increased Backlog	×	×	×	✓	×
Reduced Timeout	○	×	✓	✓	✓
Recycling	○	✓	✓	✓	✓
SYN Cache	✓	✓	×	✓	×
SYN Cookies	✓	✓	×	×	✓
SYN Authentication	✓	✓	✓	✓	×
SYN Agent	✓	✓	✓	✓	✓
Three Counters	○	×	×	✓	×
Random Drop	×	✓	✓	✓	✓

✓ (fulfilled), × (not fulfilled), ○ (depends on the attack)
This table further extends the one from [6].

TABLE 2: Comparison of Mitigation Techniques

References

[1] W. Eddy, "TCP SYN Flooding Attacks and Common Migitations", RFC 4987, August 2007

[2] R. Nagai, W. Kurihara, S. Higuchi, T. Hirotsiu, "Design and Implementation of an OpenFlow-based TCP SYN Flood Mitigation", 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, 2018

[3] H. S. Salunkhe, Prof. S. Jadhav, Prof. V. Bhosale, "Analysis and Review of TCP SYN Flood Attack on Network with Its Detection and Performance Metrics", International Journal of Engineering Research & Technology, January 2017

[4] S. Gavaskar, R. Surendiran, Dr. E. Ramaraj, "Three Counter Defense Mechanism for TCP SYN Flooding Attacks", International Journal of Computer Applications, September 2010

[5] J. Lemon, "Resisting SYN flood DoS Attacks with a SYN Cache", USENIX Association, February 2002

[6] L. Ricciulli, P. Lincoln, P. Kakkar, "TCP SYN Flooding Defense"

[7] P.-E. Liu, Z.-H. Sheng, "Defending Against TCP SYN Flooding with a new Kind of SYN-Agent", Proceeding of the 7th International Conference on Machine Learning and Cybernetings, July 2008

[8] R. Oncioiu, E. Simion, "Approach to Prevent SYN Flood DoS Attacks in Cloud"

[9] F. Baker, P. Savola, "Ingress Filterin for Multihomed networks", RFC 3704, March 2004

[10] radware Inc, "Operation Ababil", Online, 2013, last visited 2018-12-06, https://security.radware.com/WorkArea/DownloadAsset.aspx?id=848

doi: 10.2313/NET-2019-06-1_14