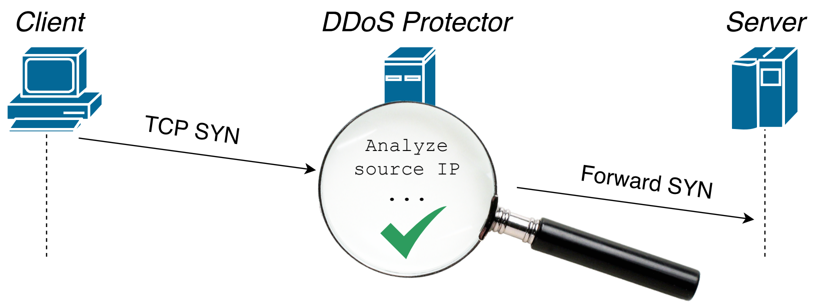
BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2020
http://excel.fit.vutbr.cz

Adaptive SYN Flood Mitigation Based on Attack Vector Detection and Mitigation Process Monitoring

Patrik Goldschmidt*



Abstract
TCP SYN Flood is one of the most widespread DoS attack types performed on computer networks nowadays. The attack comes in many possible forms and several different mitigation methods to deflect it also exist. This paper discusses mentioned security incidents, various mitigation approaches, and presents a mechanism able to choose the most suitable method to mitigate the attack. The suggestion is made according to network traffic and the properties of mitigation methods. After the suggested method is deployed, the algorithm also monitors its behavior and may suggest a different strategy when the one currently in use proves to be ineffective. Our experiments have shown that the mechanism is able to successfully detect several attack variants and suggest a suitable method to deflect them while trying to minimize the impact on the end-user as much as possible. On the other hand, the suggestion accuracy is heavily dependent on available mitigation methods and their properties, which need to be set manually before the system can be used.

Keywords: TCP SYN Flood — DDoS Mitigation — Adaptive DoS Protection

Supplementary Material: *N/A*

*xgolds00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

Note that Sections 1 and 2 are partially modified versions of my Excel@FIT 2019 paper [1], which also discussed SYN Flood problematics, and thus its introduction and theoretical background are relevant for this article as well.

1. Introduction

Transmission control protocol (TCP) is an integral part of the Internet protocol suite. As its importance is fundamental for the operation of the Internet, it is often misused to cause various cybersecurity threats. Data from the past several years show a strong trend towards TCP abuse to perform Distributed Denial of Service (DDoS) attacks. The report from Q4 2019 by Kaspersky Lab states that the most frequent target of a denial of service attacks was TCP, targeted by 90.5% of all the attacks [2] (Figure 1). According to Cisco, the number of DDoS attacks will double to 14.5 million p.a. by 2022 [3].

Figure 1 also shows that the majority of attacks on TCP are performed by SYN flooding, which poses a significant threat to many web services such as websites or e-mails. The purpose of this paper is to describe details of this weakness and its attack variants, present existing solutions to prevent it, and most im-

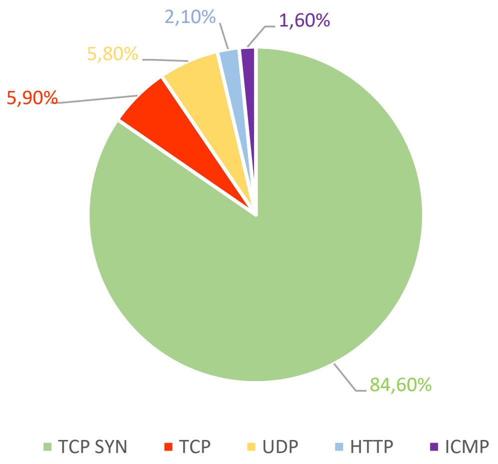


Figure 1. Distribution of DDoS attacks by type, Q4 2019 by Kaspersky Lab. [2]

portantly, introduce a system able to switch between these mitigation solutions in real-time according to network traffic patterns and effectivity of the currently employed mitigation strategy.

The mechanism was developed as a part of the CESNET's research project *DDoS Protector*[1], which currently contains 3 SYN Flood mitigation methods and requires a system to switch between them to enhance its mitigation capabilities. At the time of writing this paper, the mechanism is already implemented and tested, currently waiting for integration into the main system. However, results obtained so far are mostly experimental, and its fine-tuning is still in progress.

2. TCP Security Considerations

2.1 Known Vulnerabilities and Attacks

Threats that abuse TCP weaknesses can be classified as either flood or injection attack types. Flood attacks typically target a single host or a network. They aim to exhaust the target's resources by flooding bogus packets, making it inaccessible for regular clients, thus creating a denial of service. On the other hand, injection attacks are based on eavesdropping and injection of crafted segments into a TCP session. Injected data may contain malicious code, compromise the user's privacy [4], or reset the session [5].

The functionality of the most popular TCP attack – SYN flood depends on the process called TCP three-way-handshake, which is used to create a reliable channel for communication between two hosts. During this process, the server receives a request to set up a connection (SYN segment), responds with a confirming message (SYN-ACK segment), and waits until the client that originated the previous SYN confirms an establishment of the connection. The rationale behind a successful DoS assumes that the server (victim) allocates a new state for every received SYN segment, and that there is a limit of such states that can be stored. These are described in RFC 793 as Transmission Control Block (TCB) data structures. TCB structures are used to store necessary state information for an individual connection. They may be implemented differently among the operating systems, but the key concept is that a new memory chunk needs to be allocated upon every new TCP connection [6].

[1]https://www.liberouter.org/technologies/ddos-protector/

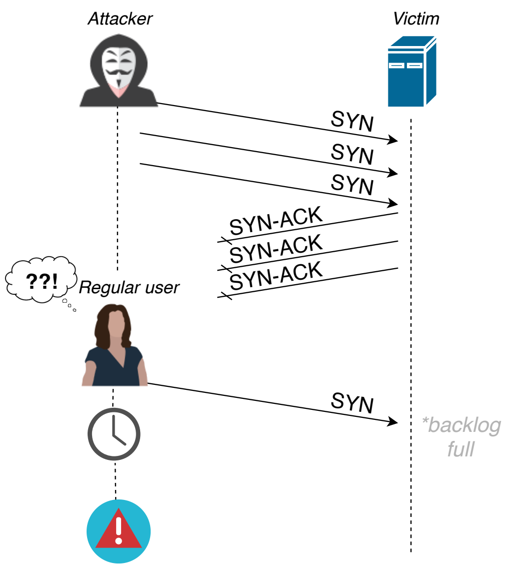


Figure 2. TCP SYN flood attack.

Operating system kernels typically try to protect host memory from getting exhausted by implementing a limit of contemporary TCB structures called backlog. When the backlog limit is reached, either incoming SYN segments are ignored, or uncompleted connections in the backlog are replaced. As illustrated in Figure 2, the primary goal of SYN flooding is to exhaust the target's backlog with half-open connections so that legitimate clients will be unable to connect. For this purpose, spoofed IP addresses that do not generate a reply to SYN-ACKs are often used.

More sophisticated variants of this attack include SYN/SYN-ACK floods, Fake session (SYN + SYN-ACK + FIN flood), which flood the server with various types of TCP segments to simulate the traffic of a real client. This is done to disguise their malicious intentions since pure SYN attacks are more likely to be detected. Another special technique – Session attack utilizes a botnet to create a lot of valid TCP connections at once and stretch them as long as possible, making the victim server inaccessible [7].

Some less sophisticated TCP attacks may rely on pure strength, typically flooding SYN-ACK, fragmented ACK, PSH-ACK, RST or FIN segments in enormous numbers. Regular stateless firewalls typically do not analyze and filter this type of traffic, therefore, it can reach its victim more easily than pure SYN flooding. From now on, only the SYN Flood attack and its mitigation will be considered.

2.2 SYN Flood Mitigation Techniques

Modern operating systems provide relatively large backlogs, thus being protected from big spikes in regular TCP traffic. However, this is not sufficient to cover flood attacks described in 2.1, so specialized methods are still required. Linux kernels historically provided robust security by implementing two end-host countermeasures – *SYN cookies* and *SYN caching* [8].

SYN cache method utilizes hashing to store a lightweight fingerprint of the IP address, port number and a secret for every incoming TCP connection. This way, the operating system does not need to allocate the whole TCB, but only a fragment of the original memory required. A device implementing this method is, therefore, able to queue more requests, becoming harder to exhaust. In the BSD kernel from 2002, this optimization reduced the size of the per-connection data by 78 % while allowing up to 15359 entries [8].

In contrast to SYN cache, SYN cookies method does not need to store any state information at all, requiring no memory per connection. Essential data defining the connection alongside with a timestamp and a secret are hashed into a 32-bit value representing the *SEQ* number of the SYN-ACK segment. Upon ACK response receipt, the server can reconstruct original SYN parameters and successfully establish a connection. The method is often highly effective, but its nature denies SYN-ACK retransmission and also restricts TCP options usage [9].

Other interesting approaches, like *TCP Random Drop* [10], which aims to replace a random half-open connection with another, also exist. However, the usage of end-host mitigation techniques is sometimes undesired (e.g., to preserve system resources), unsuitable in certain environments, or insufficient against some types of attacks. Therefore, other specialized methods need to be employed as well.

Various TCP extensions and modifications with anti-DoS capabilities like TCP Cookie Transactions [11] and TCP Fast Open [12] also exist. However, none of them is globally used mainly due to a lack of support from software companies and hardware vendors.

Other potential ways for TCP DoS protection include network-based countermeasure techniques. The simplest of them is *traffic filtering* [13] and its improved variant *reverse-path forwarding* [14]. Their fundamental idea is to deny all incoming traffic from IP addresses that do not match their source network prefix (packets intentionally crafted with false IP). This process would allow discarding all traffic from spoofed IP addresses, but is not reliable, because filtering policies would need to be deployed by a majority of Internet service providers, which cannot be generally relied on (2006) [15].

SYN flood attacks were historically mitigated by firewalls, proxies, or IDS/IPS systems, which usually used initiator SYN-ACK spoofing or listener ACK spoofing techniques [15] [16]. These mitigation practices are mostly present to this day, though many security solutions also utilize cloud technologies with virtualized IDS/IPS systems instead of traditional per-network defense [17].

In real-world situations, both end-host and network-based techniques are frequently employed, and they generally do not interfere when used in combination [15]. Newer trends in DDoS mitigation also employ artificial intelligence and machine learning principles such as [18]. These approaches are generally able to protect against a wider range of attacks but suffer from a poorer performance when compared to their previously mentioned counterparts.

As may be seen, the presented methods have different behavior and consume various amounts of system resources. They also act differently from the side of the end-user. Therefore, a switching mechanism between these methods able to turn on "the best" is desired. This algorithm should choose a strategy that is able to successfully deflect the attack and affect the end-user as little as possible.

3. CESNET's DDoS Protector Environment

As mentioned in Section 1, this project was developed as a part of the larger-scale research – *CESNET's DDoS Protector*. This system aims to provide DDoS protection for high-speed networks, using proprietary network interface cards based on FPGA technology with a custom firmware and drivers. The Protector was firstly designed to mitigate reflective and amplification DDoS attacks but gradually evolved into a system with different capabilities, such as a TCP SYN Flood mitigation. Currently, the following SYN Flood network-based mitigation strategies are supported:

- TCP SYN Drop
- TCP Reset Cookies
- TCP ACK Spoofing

Traditionally, the mitigation methods were strictly bound to the protected network's prefix by rules, which

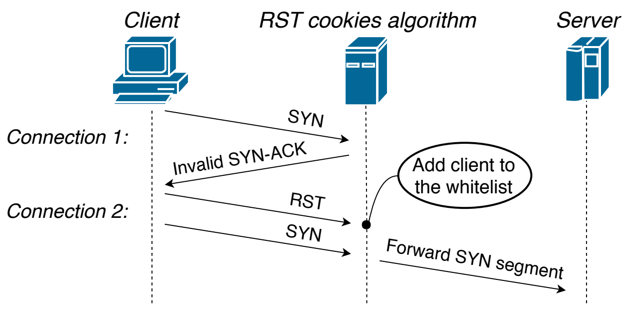


Figure 3. RST Cookies functionality.

had to be specified by the system's user. Therefore, the same mitigation principles were applied regardless of the properties of the attack. Also, changing the method required manual intervention as well. For this reason, a mechanism of *Adaptive SYN Flood Mitigation Modules*, as described in Section 4, was designed and developed. This section will further analyze the mentioned available mitigation methods.

3.1 TCP SYN Drop

TCP SYN Drop is a proprietary method developed especially for CESNET's DDoS Protector. Its functionality depends on soft and hard thresholds, which are used to limit the maximum throughput of SYN data allowed from a single client. Each client has an associated counter based on its IP address stored in the module's internal structures. If the client has not sent any ACK segment yet, the maximum number of SYN segments it can send is given by the soft threshold. If at least one ACK from the particular IP is received, SYNs are limited by the higher hard threshold. This mechanism effectively limits the number of connection requests from a single IP but can be easily fooled by injecting a random ACK segment into the session or by constantly spoofing source addresses [19].

3.2 TCP Reset Cookies

TCP Reset Cookies method utilizes the TCP three-way handshake mechanism to establish a security association with a client before forwarding its SYN data. As outlined in Figure 3, the association is established by intentionally crafting an invalid SYN-ACK response to the first SYN received from a client. According to the RFC 793, the client is expected to respond with the RST segment containing a specific sequence number (SEQ). The mechanism compares expected SEQ with the received one and adds IP address on the whitelist if they match. Connection requests from whitelisted clients are forwarded, whereas ones from non-whitelisted addresses are dropped. With the incorporated functionality of SYN Drop mechanism, this variant provides significantly better security, though

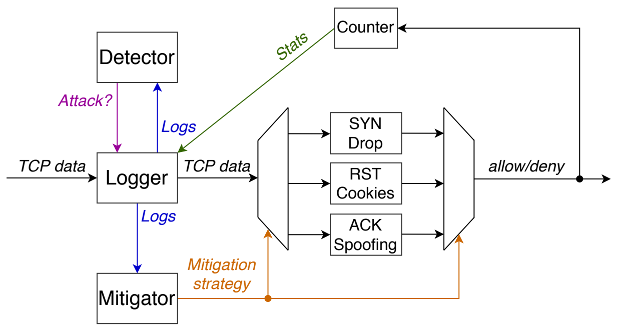


Figure 4. Adaptive SYN Flood Mitigation Modules overview.

at the cost of higher system resources utilization and bigger impact on the end-client [1].

3.3 TCP ACK Spoofing

The goal of the TCP ACK Spoofing approach is to prevent the exhaustion of the protected device's backlog by denying it to get overfilled. This process works on the principle of sending spoofed ACK segments to finish every half-open session and complete the three-way handshake. If the client does not generate an ACK segment within the specified timeout period, the ACK spoofing mechanism terminates the connection with an RST segment. If the expected ACK is received, the algorithm marks the connection as valid and does not interfere in the future TCP communication between the nodes. The method prolongs the server's ability to serve clients but does not mitigate an attack by itself. Therefore, it is not suitable against excessive volumes of SYN traffic, but can be worth a try when all other methods prove to be ineffective [19].

4. Adaptive SYN Flood Mitigation Modules

As outlined before, the mechanism for dynamic method switching requires to provide several features to work as desired. This functionality is provided by the following 3 separate submodules, collectively named as *Adaptive SYN Flood Mitigation Modules*:

- SYN Flood Detector
- SYN Flood Logger
- SYN Flood Mitigator

In short, these modules monitor live traffic (Logger), compare it to different thresholds and search for patterns (Detector), and choose the proper SYN flood mitigation method according to them (Mitigator) (Figure 4). Each of these submodules is discussed in more detail in the following subsections.

4.1 SYN Flood Logger

The main purpose of the SYN Flood Logger submodule is to gather and store statistics of the ongoing

TCP traffic for further analysis. This is achieved by directly processing IP and TCP headers of SYN, ACK and RST segments and incrementing their associated counters. To estimate the potential usage of whitelists and other data structures, the information about unique IP addresses needs to be collected as well. Since precious IP counting would be too expensive, the submodule utilizes the HyperLogLog (HLL) probabilistic data structure. Probabilistic data structures offer an ability to store vast amounts of data in relatively small structures (KiB to MiB) at the cost of precious results for approximate values. For our purposes, 512B big 9-bit HLL with a standard error of 4.6% was used.

Other statistics required to determine successful mitigation, such as the number of allowed SYNs, the number of denied SYNs, and the status of the attack (active/inactive) can not be collected directly, but the user or other submodules are responsible to provide them by appropriate function calls.

Collecting and monitoring the network statistics with respect to time is allowed by splitting the data collecting process into blocks with equal duration. Each block (time window) is represented by a single log structure, which contains all the statistics described above and adds a context about when its data have been collected. For mitigation purposes, the absolute time is not necessary, so we are working with the relative time considered from the current moment up to the N time windows to the past. When a new window is created, the logs are rotated, so the oldest log is automatically replaced. The recommended length of the window is several seconds, so the mechanism stays flexible and quick to react to potential attack vector changes.

Real-time environments such as CESNET's DDoS Protector are expected to have minimum packet delays and be able to handle tens of gigabits of traffic per second. For this reason, the traffic logging process needs to be parallelized into several threads, each running its own Logger instance. Results from all instances are then merged into a master-logger, which is used for querying and further data analysis. Since the process of merging logs takes some time, each thread needs to have 2 independent logger instances and support a mechanism to write only to Loggers currently not being merged. This way, the statistics can be collected and evaluated in mutex-free manner without data races.

4.2 SYN Flood Detector

SYN Flood Detector is a stateless submodule used to find out whether the SYN Flood attack is present or not. The current implementation of the Detector consists of the set of triggers and thresholds, which are fired when certain patterns and conditions are found in the statistics structure collected by the Logger submodule. Currently, active detection triggers include simple SYN messages count in the current time window, but also more advanced patterns, such as ratios between unique SYN addresses and unique pure ACK addresses and also historical thresholds, such as if the SYN ratio was consistently high in the past several time windows but the hard limit in the current time window was not reached, the possible attack may get reported anyway. As displayed in Figure 4, the results from the Detector are used to update window statistics in the Logger submodule. These data are then passed to the Mitigator instance for further analysis.

4.3 SYN Flood Mitigator

SYN Flood Mitigator submodule is responsible for mitigation methods management and decision-making which one is the most suitable to deflect the ongoing attack. Its principle is based on maintaining a list of available mitigation methods, choosing one of them according to their properties, and then determining whether its employment is effective against the attack or not. If the mitigation proves to be ineffective, the method is marked not to be used again during this particular attack, and another available strategy is tried.

The list of available methods is created by registration. This process involves specification of a mitigation method strategy structure, which is supposed to describe properties of the mitigation method, such as how much computer memory does it need, how many hashing functions it computes during the segment processing, or which types of TCP traffic does it process. There are currently 15 of these method parameters, aiming to provide a very detailed description of methods, but be as general as possible, so new methods may be easily added in the future. In other words, the whole mechanism is not limited only to 3 mitigation methods supported by the DDoS Protector.

Internally, the registered mitigation methods are stored in an array ordered based on their rating. The rating describes an impact the mitigation method has on the end-user. Processing frequently used TCP segments, such as ACKs, hashing, dropping SYNs, requiring retransmission, etc. negatively impacts the end-user, and hence increases the rating value. Mitigation methods with the lowest ratings are placed at the beginning of this array, so the method determination process considers them first. More heavyweight methods reside at the end, so they are not used until more serious conditions occur.

The optimal method is determined according to discovered traffic patterns – such as if the method can be effective and its whitelists will not get overfilled

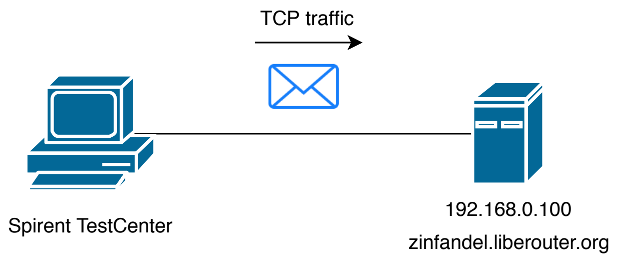


Figure 5. Experimental configuration testing environment scheme.

by many unique IP addresses. If more than one suitable method is found, the mechanism always chooses the one having a lower rating, thus having a lesser impact on the end-user. After the method is chosen, it is not changed for several time windows to gather enough mitigation statistics and prevent frequent method switching. If the method is determined to be ineffective (e.g., not dropping any SYN segments while the attack is ongoing), it is marked as unsuitable and not taken into account during the decision-making process for the several following time windows.

5. Evaluation and Results

According to the modular and highly configurable design, all of the submodules described in Section 4 can be customized and fine-tuned during both the compilation and execution stages using various macros and configuration structures. For example, the user may specify the sensitivity of the Detector thresholds, defining that an attack is triggered sooner or later. Weights for methods ratings can also be altered, so the user is in full control over which mitigation method will get prioritized when searching for the best match. As the reader may have noticed, the functionality and mitigation capabilities of the mechanism are particularly dependent on the configuration it is initialized with.

At first, an initial experimental configuration was created for mechanism evaluation purposes. This setup then was iteratively developed based on its performance in the performed tests. Our test environment consisted of a simple point-to-point network (Figure 5), established in CESNET's Liberouter project private network. The program using *Adaptive SYN Flood Mitigation Modules* was deployed on a server connected to a machine running *Spirent TestCenter* application[2]. This program provides a feature to forge custom packets at excessively high speeds, thus being an ideal choice to simulate a SYN Flood DDoS attack.

The current implementation of the SYN Flood detection in the CESNET's DDoS Protector is based only on a single threshold. Our tests have shown that adaptive approach described in this paper is much more sensitive and can react on wider variety of possible attack vectors. Apart from regular threshold detection, our mechanism successfully reports events like a sudden increase of the SYN traffic. For example, if the volume of SYN traffic increases by 20% over the past 30 time windows (e.g., 30 seconds), and at least 80% of the SYN threshold is reached, a possible attack is reported. This may, of course, produce a number of false positives such as in the morning when employees are arriving to work. Again, all these facts showcase the importance of the configuration.

[2]spirent.com/products/testcenter

Rank	Method	Rating
1.	TCP SYN Drop	127.014
2.	TCP RST Cookies	149.100
3.	TCP ACK Spoofing	491.927

Table 1. Mitigation methods Mitigator ratings

There are many more detection patterns, such as the ratio between unique IP addresses sending (pure) ACK segments and unique addresses sending SYNs, which may also signal a possible attack. Currently, a value we use for this threshold is 1.20, but again, it is highly dependable on the environment.

After initially filling the properties of the mitigation methods from Section 3, the ratings shown in Table 1 were obtained. This corresponds to our expectations since the processing of ACK segments is rather costly. This property results from the high share of ACK segments in regular TCP traffic (more than 90% according to our measurement between CESNET and ACONET networks in 2018). After running the algorithm with this setup upon regular pure SYN Flood attack, we observed a behavior which followed its theoretical functionality described in Section 4. The method with the lowest mitigation rating (SYN Drop) was chosen, and the mechanism did not change it until the pure SYN attack was in progress. As we begin mixing SYNs with several fake ACKs, the SYN Drop mitigation started to be ineffective, so the mechanism switched to RST Cookies and again, kept it active until we managed to fool it with regular TCP stack as well. In this case, the last resort method – TCP ACK Spoofing was employed. When we completely turned off the SYN generation, ACK Spoofing was held for several more time windows before stopping the mitigation completely. Several similar tests like this were also conducted with expected results corresponding to the module configuration.

As a result of these findings, we deduce that the mechanism is a step ahead from the current DDoS Protector's state and may work fine if configured properly. However, the trend of setting everything manually is unscalable and unsuitable in general. For this reason, we are trying to find a "good" universal configuration, that will work fine until the setup process could be done in a more automatized manner. The author of this document is already working on TCP segment classification using machine learning techniques, which may complement or replace the Detector's functionality in the future.

6. Conclusions

This paper has focused on the analysis of the TCP SYN flood attack and discussed several techniques for its mitigation. All of the examined methods have certain advantages and disadvantages, making them suitable for different variants of the attack. To tackle this issue, the paper has presented the *Adaptive SYN Flood Mitigation* mechanism, which aims to provide a way to determine the most viable mitigation method based on detected traffic patterns and mitigation statistics.

Our experimental results have proved that the mechanism can detect regular and more sophisticated SYN floods that the current DDoS Protector triggers were not able to react on. However, its mitigation capabilities are heavily dependent on the configuration, which is also partially dependent on the protected network properties and the environment the algorithm is deployed in. The "universal" configuration able to facilitate most of the environments has not been found yet, so we are continuing to tweak and experiment with the mechanism to achieve better performance and mitigation capabilities.

Although the described approach is officially finished, further improvements can still be made. Apart from finding its optimal and universal configurations, there is a huge potential in improving the detection capabilities and the mitigation method suggestion principles. The Logger submodule may also be slightly modified and reused for feature extraction for the DDoS mitigation machine learning project, which the author of this article is currently starting to work on.

Acknowledgements

I would like to thank my supervisor *Ing. Jan Kučera* for his friendly nature and support. Many thanks also to *Ing. Martin Žádník Ph.D.*, who introduced me to the CESNET's security research projects back in 2017.

This project was developed as a part of the research "Adaptive protection against DDoS attacks[3]" granted by Ministry of Interior, Czech Republic under the auspices of CESNET a.l.e. and its contractual research for students, in cooperation with Faculty of Information Technology at Brno University of Technology.

[3] starfos.tacr.cz/en/project/VI20192022137

References

[1] P. Goldschmidt. TCP Reset Cookies – a heuristic method for TCP SYN Flood mitigation. In *Excel@FIT 2019*, Apr 2019.

[2] Kupreev, Badovskaya, and Gutnikov. Ddos attacks in q4 2019. Technical report, Kaspersky Lab, February 2020. Available at: https://securelist.com/ddos-report-q4-2019/96154/.

[3] Cisco Systems. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper. Technical report, Jan 2017.

[4] B. Harris and R. Hunt. TCP/IP security threats and attack methods. *Computer Communications*, 22(10), 1999.

[5] Paul A. Watson. Slipping in the Window: TCP Reset Attacks. Jan 2004.

[6] W. Eddy. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987 (Informational), August 2007.

[7] Riorey, Inc. Taxonomy of DDoS Attacks. Accessed: 2019-08-26.

[8] Jonathan Lemon. Resisting SYN Flood DoS Attacks with a SYN Cache. In *USENIX Conference on the BSD operating system*, 2002.

[9] Daniel J. Bernstein and Eric Schenk. SYN Cookies proposal, Sept 1996. Accessed: 2019-08-26.

[10] L. Ricciulli, P. Lincoln, et al. TCP SYN Flooding Defense, 1999.

[11] W. Simpson. TCP Cookie Transactions. RFC 6013, Jan 2011.

[12] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP Fast Open. RFC 7413, Dec 2014.

[13] P. Fergusson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, May 2000.

[14] Baker, F. and Savola, P. Ingress Filtering for Multihomed Networks. RFC 3704, Mar 2004.

[15] Wesley M. Eddy. Defenses Against TCP SYN Flooding Attacks. *The Internet Protocol Journal*, 9(4), Dec 2006.

[16] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a Denial of Service Attack on TCP. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1997.

[17] Opeyemi Osanaiye et al. Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. *Journal of Network and Computer Applications*, 67, 2016.

[18] Chuanhuang Li, Yan Wu, Xiaoyong Yuan, Zhengjun Sun, Weiming Wang, et al. Detection and defense of DDoS attack-based on deep learning in OpenFlow-based SDN. *International Journal of Communication Systems*, 31(5), 2018.

[19] P. Goldschmidt. Heuristic Methods for the Mitigation of DDoS Attacks That Abuse TCP Protocol, May 2019. Bachelor thesis. Faculty of Information Technology, Brno University of Technology.