DDoS On Sketch: Spoofed DDoS attack defense with programmable data planes using sketch in SDN

Kairo Tavares[1], **Tiago Ferreto**[1]

[1]PPGCC - Graduate Program in Computer Science
PUCRS - Pontifical Catholic University of Rio Grande do Sul

kairo.tavares@acad.pucrs.br, tiago.ferreto@pucrs.br

Abstract. *Distributed Denial of Service (DDoS) attacks are still a major issue in today's Internet. Over the last few years, we have observed a dramatic escalation in the number, scale, and diversity of these attacks. Among the various types, spoofed TCP SYN Flood is one of the most common forms of volumetric DDoS attacks. Several works explored the flexible management control provided by the new network paradigm called Software Defined Networking (SDN) to produce a flexible and powerful defense system. However, these works usually present an increase in all clients connection time, or vulnerability to buffer saturation attacks. In this work, we propose the use of sketch-based solutions to improve the data plane Safe Reset anti-spoofing defense mechanism. We implemented our solution in P4, a high-level language for programmable data planes, and evaluate our solution against a data plane Safe Reset technique on an emulated environment using Mininet.*

Resumo. *Os ataques de negação de serviço distribuído (DDoS) ainda são uma questão importante na Internet nos dias de hoje. Nos últimos anos, observamos um aumento significativo no número, escala e diversidade desses ataques. Entre os vários tipos, o Spoofed TCP SYN Flood é um dos tipos mais comuns de ataques de DDoS volumétricos. Diversos trabalhos exploraram o controle flexível de gerenciamento fornecido pelo novo paradigma de rede chamado SDN (Software Defined Networking) para produzir um sistema de defesa flexível e poderoso. No entanto, esses trabalhos geralmente apresentam um aumento no tempo de conexão para todos os clientes ou vulnerabilidade a ataques de saturação de buffer. Neste trabalho, propomos o uso de soluções baseadas em sketch para melhorar o mecanismo de defesa anti-spoofing Safe Reset no plano de dados. Implementamos nossa solução em P4, uma linguagem de alto nível para planos de dados programáveis, e avaliamos nossa solução em relação a uma técnica de Safe Reset no plano de dados em um ambiente emulado usando o Mininet.*

1. Introduction

Despite the extensive industrial and academic efforts, Distributed Denial of Service (DDoS) attacks are still a major issue in today's Internet. Over the last few years, we have observed a dramatic escalation in the number, scale, and diversity of DDoS attacks. These attacks can be described as a malicious attempt to make the victim unavailable by depleting its resources, such as CPU cycles, memory, and network bandwidth, or exploring design flaws.

Among the various DDoS attacks, TCP SYN Flood [Wang et al. 2002] is one of the most common forms of DDoS attacks. In fact, according to a Radware report [Radware 2016], 40% of the organizations experienced a TCP-SYN Flood attack in 2016. It is a volumetric attack that generates high-volume traffic and aims to exploit an implementation characteristic of the Transmission Control Protocol (TCP) to make server processes incapable of answering a legitimate client application's requests for new TCP connections. Any service that binds to and listens on a TCP socket is potentially vulnerable to TCP SYN Flood attacks. There are two main causes [Sun et al. 2009] of TCP SYN Flood attacks: i) the inherent asymmetry feature in TCP three-way handshake protocol, which enables an attacker to consume substantial resources at the server, while sparing its resources; ii) the fact that the server cannot control the packets it receives, especially SYN packets, can easily reach the server without its approval.

A common form of TCP SYN Flood attack uses an spoofed IP address to mask the attacker's identity. It is a more complex method where the attacker also needs to be able to create and inject raw IP packets with valid IP and TCP headers. Nowadays, popular libraries exist to aid with raw packet formation and injection, so attacks based on spoofing are fairly easy. An important consideration for this attack variation to succeed is the address selection, since the machines at the spoofed source addresses must not respond to the SYN-ACKs that are sent to them in any way. Therefore, attackers may select one or multiple addresses based on the prior knowledge that the source address is unreachable by the machine under attack.

To address this problem, DDoS defense mechanisms have been deployed in the network infrastructure to detect and mitigate the attacks. Usually, these defenses are enforced by using expensive and proprietary hardware appliances [Dzurenda et al. 2015, Martinasek 2015] that are fixed in terms of placement, functionality, and capacity. First, they are typically deployed at fixed network aggregation points (e.g., a peering edge link of an ISP). Second, they provide fixed functionality with respect to the types of DDoS attacks they can handle. Third, they have a fixed capacity with respect to the maximum volume of traffic they can process [Fayaz et al. 2015].

Ideally, a DDoS defense architecture should provide the flexibility to seamlessly place defense mechanisms where they are needed and the elasticity to launch defenses as needed depending on the type and scale of the attack. To address those requirements, several works [YuHunag et al. 2010, Giotis et al. 2014, Xing et al. 2013, Braga et al. 2010] explored the flexible management control provided by the new network paradigm called Software-Defined Networking (SDN) [McKeown 2009, Kreutz et al. 2015]. Among them, data plane based solutions [Shin et al. 2013, Afek et al. 2017] combined with the recently flexibility of programmable switches aims to leverage the hardware speed defense against Spoofed Flooding attacks. Usually, they implement anti-spoofing mechanisms that rely on making the switch interact and authenticate the client using techniques such as TCP Proxy, TCP Reset, and Safe Reset. These techniques allow the infrastructure to authenticate the TCP SYN request client first, before sending the request to the destination server. However, those mechanisms have several limitations. First, due to the required interaction to authenticate the client, they penalize all clients connection time even without an ongoing attack. Second, they have limited capacity to detect a valid client ACK or RST, and finally, they are vulnerable to a buffer saturation attack due to

limited data plane resources used to store a whitelist with authenticated users.

Even with this flexibility and speed, solutions in the data plane are restricted due to the limited capacity of hardware resources in terms of CPU, memory, and storage. As such, streaming algorithms, or sketches, have been used into network elements (e.g., routers or middleboxes) in high-speed networks to handle a high volume of traffic while maintaining a low consumption overhead of CPU and memory. Several sketches have been proposed to detect TCP-SYN Flood attacks [Sun et al. 2009, Dodig et al. 2017]. They rely on keeping tracking of the TCP Handshake to detect flooding attacks.

In this paper, we propose SACK3, a sketch-based solution implemented for programmable data planes, that aims to detect ongoing TCP SYN Flood attacks and only apply anti-spoofing defenses, e.g., Safe Reset proxy, on client requests to servers under attack, thus reducing the overhead on legit client connections. Our proposal not only leverages the hardware speed but also keeps the resources usage within hardware limitations, with the cost of a fixed false positive probability. We implemented our solution in P4, a high-level language for programmable data planes, and evaluated our solution against a data plane Safe Reset technique on an emulated environment with Mininet.

In summary, this paper presents the following contributions: (i) a technique, called SACK3, to detect and mitigate TCP SYN Flood attacks using anti-spoofing techniques while reducing the overhead of legit clients, (ii) a implementation of the improved version of SACK2 [Sun et al. 2009] that uses dual counting Bloom Filters [Dodig et al. 2017] for programmable data planes in P4, and (iii) a comprehensive evaluation of SACK3, which confirms its effectiveness against TCP SYN Flood attacks. Moreover, the experiments have shown a significant reduction of connection time against a pure Safe Reset proxy implemented in the data plane.

The rest of this document is organized as follows. Section 2 presents related work relevant to our proposal. The SACK3 technique is presented in Section 3. Section 4 presents experimental results. Finally, Section 5 concludes the paper.

2. Related Work

In SDN-based networks, several defense mechanisms against TCP SYN Flood attacks have been proposed over the years. They mainly focus on two security problems: (i) the improvement of solutions to detect and mitigate attacks against services on the network, and (ii) the security vulnerabilities introduced by the SDN architecture that can be exploited by flooding attacks, such as control channel saturation and OpenFlow table exhaustion. In this section, we classify these mechanisms into three groups: (i) control plane solutions, (ii) data plane solutions, and (iii) sketch-based solutions.

2.1. Control plane solutions

SPHINX [Dhawan et al. 2015] is a framework which has been proposed to detect security attacks in SDN. SPHINX is implemented on the control plane and can detect TCP SYN Flood attacks by investigating the rate of packet-in messages which correspond to the new SYN requests. If the rate of a new SYN request is above the administrator-specific threshold, SPHINX raises an alert. The downside is that SPHINX does not investigate the SYN requests to distinguish the legitimates from non-legitimate requests and only strictly controls SYN requests with a predefined threshold.

OPERETTA, proposed by Fichera et al. [Fichera et al. 2015], is an OpenFlow-based approach implemented in the controller that investigates new SYN requests for detecting and rejecting malicious requests. It acts as a proxy between the TCP client and server. This solution listens for TCP SYN packet-ins from the switch and sends a forged SYN-ACK to the client. If the client sends the ACK message, therefore being authenticated, the controller sends an RST message to the host and installs a rule between the client and server on both sides. In this approach, the first attempt to establish a TCP connection is always rejected even for legitimate requests. Moreover, an adversary might flood the controller with complete TCP handshakes and impose the controller to install a new rule.

Finally, SLICOTS [Mohammadi et al. 2017] was proposed as an alternative to OPERETTA that does not penalize legitimate connections. It is implemented as a lightweight OpenDayLight (ODL) extension, and it monitors all ongoing TCP connections in the network for detecting and preventing SYN flood attacks. For each TCP connection request, SLICOTS installs temporary forwarding rules during the TCP handshaking process, and after validation of a request, it installs permanent forwarding rules between the client and server. Moreover, SLICOTS blocks the attacker that makes a large number of half-open TCP connections. However, it is vulnerable to table saturation in cases of spoofed attacks that generate a significant number of different client IP addresses that need to be routed or blocked.

2.2. Data plane solutions

AVANT-GUARD [Shin et al. 2013] has been proposed as a solution for detection and prevention of TCP SYN Flood attack. AVANT-GUARD uses a connection migration mechanism in data plane switches as a proxy for incoming SYN packets and limits the effect of the control plane saturation attack. Although AVANT-GUARD is beneficial in a general case, it causes a long delay in establishing new connections for legitimate TCP requests. Moreover, it is vulnerable to the buffer saturation attack [Ambrosin et al. 2017] and limits the number of connections that a data plane switch can proxy to the number of available TCP port numbers. It also requires upgrading the data plane equipment to support AVANT-GUARD features.

LineSwitch [Ambrosin et al. 2017] is an efficient solution that removes the limitations of AVANT-GUARD and tackles buffer saturation attack. Similar to AVANT-GUARD, LineSwitch uses the SYN proxy technique in data plane switches. However, LineSwitch implements probabilistic blacklisting of network traffic for mitigating the buffer saturation attack. The use of probabilistic blacklisting decreases the size of needed memory for storing the state of ongoing connections. Since LineSwitch mediates a minimum number of connections as compared to AVANT-GUARD, the limitation of the number of connections is reduced. LineSwitch is an effective solution against control plane saturation attack, but as well as AVANT-GUARD, it also requires an upgrade on data plane switches to be applied.

An anti-spoofing data plane solution proposed by Afek etal. [Afek et al. 2017] aims to explore the switch data plane to reduce the CAPEX, latency, and complexity of traditional DDoS anti-spoofing scrubbers that require dedicated middleboxes in the network. It starts showing that the current SDN OpenFlow match-and-action model is

rich enough to implement a collection of anti-spoofing methods. Secondly, they utilize advanced methods for dynamic resource sharing to distribute the required mitigation resources over a network of switches. The anti-spoofing methods include TCP Proxy, HTTP Redirect, TCP Reset, Safe Reset, and DNS Spoofing Cookie.

Among the anti-spoofing techniques implemented, the Safe Reset is the only one that can be applied to all TCP clients connections without compromising the TCP options required for performance, with no application-level awareness or requiring modifications on the client side. It is described as the weak version of the SYN cookie, where the switch responds to the initial SYN message with an SYN-ACK packet containing a bogus ACK number. If the client is compliant with RFC 793 (TCP), it immediately responds to the bogus ACK number with a RST packet containing the bogus ACK number as its sequence number (thus being authenticated). Then, after one second, the client initiates a new connection by sending a new SYN request.

2.3. Sketch-based solutions

Kompella et al. [Kompella et al. 2004] introduced a modified counting Bloom Filter called Partial Completion Filter (PCF) to keep track of the differences between SYN and FIN packets. However, a spoofed FIN RST packet can obstruct the algorithm.

Chen and Yeung [Chen and Yeung 2006] proposed using SYN-ACK pairs with the counting Bloom Filter. They did support differentiating between ACK and Data ACK packets. Nevertheless, a spoofing SYN and ACK packets may still obstruct the algorithm.

An accurate and efficient way of using counting Bloom Filters to keep track of TCP SYN Flood attacks, even from a spoofed source, is proposed in SACK2 [Sun et al. 2009]. SACK2 exploits the behavior of the SYN/ACK-CliACK pair to identify the victim's server and the TCP port being attacked, where a SYN/ACK packet is sent by a server when receiving a connection request, and a CliACK packet is the ACK packet sent by the client to complete the three-way handshake. Experiments shown that the memory cost of SACK2 for a 10Gbps link is 364KB and can be easily accommodated in modern routers. An improved version of SACK2 was proposed in [Dodig et al. 2017] that is able to use dual counting Bloom Filter (DCBF) to decrease false detection of matching packets.

3. Proposed Solution

As previously discussed, data plane anti-spoofing mechanisms that rely on making the switch to interact and authenticate the client have several limitations. We choose to focus on the Safe Reset proxy because it is the technique, among the others implemented in the data plane by [Afek et al. 2017], that can be applied to all TCP client connections, without compromising the TCP options required for performance, with no application-level awareness or modifications on the client side. This technique has limitations as follows: (i) it penalizes all clients connection time, even without an ongoing attack, (ii) it has limited capacity to detect a valid client ACK or RST, and (iii) it is vulnerable to a buffer saturation attack due to limitations in memory capacity to store the whitelist of authenticated users.

In this paper, we propose SACK3, which uses sketch-based solutions to address those limitations while leveraging the hardware speed and maintaining a low usage of

resources in the data plane. We implemented it in the P4 language for programmable data planes. It detects ongoing spoofed TCP SYN Flood and applies anti-spoofing techniques only for clients that try to connect to servers under attack. As mentioned before, due to the advantages of the Safe Reset proxy method, we choose this anti-spoofing technique to authenticate clients in SACK3 as well. Figure 1 presents a flowchart with the functioning of SACK3.

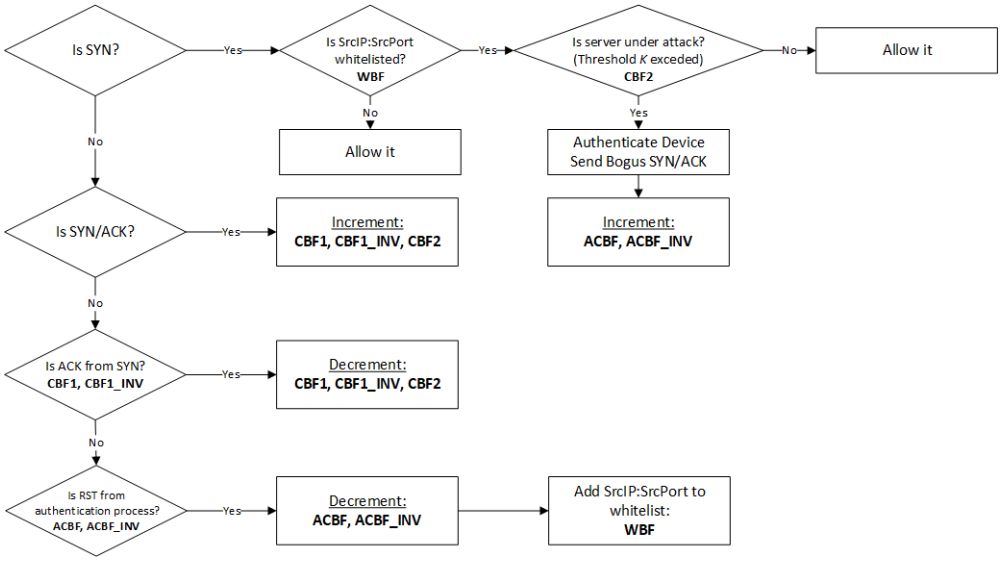

Figure 1. SACK3 flowchart

To solve the first limitation, we added a sketch-based algorithm to detect spoofed TCP SYN Flood on the data plane in order to choose suspicious clients to be authenticated. From the algorithms to detect ongoing spoofed TCP SYN Flood attacks listed in Section 2, we choose to implement an improved version [Dodig et al. 2017] of the SACK2 algorithm due to its effectiveness, low resource usage, and simplicity. We implemented the DCBF (Dual Counting Bloom Filter), composed by CBF1 and CBF1_INV, to track the SYN-ACK/CliACK pair.

DCBF uses as input the hashing of 6-tuple called *Psyn/ack* (1) to increment the filter counters. CBF1 utilizes the 6-tuple as it is, while CB1_INV uses the inverted 6-tuple values as input. This 6-tuple input structure consists of Source IP (SIP), Destination IP (DIP), Source Port (SP), Destination Port (DP), Sequential Packet Number (SEQ), and an Acknowledge Sequential Packet Number (ASEQ). This data is obtained from a SYN/ACK packet header and is hashed as follows:

$$Psyn/ack =< SIP, DIP, SP, DP, SEQ, ASEQ > \quad (1)$$

Detecting a matching pair of SYN/ACK and ACK packets starts with the detection of the ACK packet. For each detected ACK or DACK packet, a new hash of the 6-tuple named *Pack* (2) is used to decrement the filter counts. Different from increment, this sketch will only decrement the CBF1_INV when a matching pair in CBF1 is detected. This 6-tuple consists of the same members stored in a different order:

$$Pack =< DIP, SIP, DP, SP, ASEQ - 1, SEQ > \quad (2)$$

In order to detect which server is being attacked, every time a server answer a SYN/ACK, the CBF2 uses a 2-tuple hash named *Pserver* (3) to increment the filter counters. It will only decrement if the DCBF receives a valid CliACK. Therefore, CBF2 will have a counting estimation of how many unanswered SYN/ACK the server has. The minimum value of all counters can be used to compare against a threshold K to detect an ongoing attack. Notice that this technique will only detect ongoing TCP SYN Flood attacks that impact a real server. If the attacker is generating TCP SYN packets to a server that does not exist, there will be no SYN/ACK to increment and therefore no detection. This approach aims to authenticate only client requests that impact a real server, reducing the use of anti-spoofing techniques resources.

$$Pserver =< ServerIP, ServerPort > \quad (3)$$

If the threshold K is exceeded, a bogus SYN-ACK is generated to authenticate the client. The bogus SYN-ACK is made by setting ASEQ to a bogus number. It is sent back to the same port that it arrived. To keep track of the RST that is expected due to the bogus SYN-ACK that was sent, we proposed and implemented a similar DCBF only for authentication requests. It uses the hash of the SYN-ACK 5-tuple named *Pforged* (4) as input to increment this custom Authentication DCBF (ADCBF), composed by ACBF1 and ACBF1_INV. Once we receive the RST from the client that matches the forged SYN-ACK, we decrement the Authentication DCBF filter counters using the SYN-ACK 5-tuple named *Prst* (5) as input. Finally, once the client is successfully authenticated, we add the 2-tuple named *Pclient* to the whitelist Bloom Filter (WBF). The connection is allowed and all subsequent packets are forwarded. *Pforged*, *Prst*, and *Pclient* (6) are presented as:

$$Pforged =< SIP, DIP, SP, DP, ASEQ > \quad (4)$$

$$Prst =< DIP, SIP, DP, SP, SEQ > \quad (5)$$

$$Pclient =< ClientIP, ClientPort > \quad (6)$$

In summary, the proposed solution can be described as follows:

1. It uses an improved version of the SACK2 algorithm to detect the victims address and port under attack. We only authenticate clients that interact with victims' servers, thus adding a time overhead only to suspect clients.
2. We implement a similar sketch to the SACK2 algorithm to improve the capacity to detect valid ACK or RST that authenticate clients.
3. We use a space-efficient data structure, a Bloom Filter, to query for authenticated users, therefore reducing the chances of a buffer saturation attack.

4. Evaluation

In this section, we conduct a comprehensive simulation experiments and analyze the results to evaluate the performance of SACK3 against Safe Reset proxy switch and a basic switch. Using only the authentication mechanism as a Safe Reset proxy penalizes all clients connection time. We show that SACK3 can improve the connection time to other servers not under attack while maintaining the same level of protection.

4.1. Experimental Setup

We implemented SACK3 using the P4 programming language targeting the second version of the P4 software switch, also known as behavioral model or BMV2. BMV2 also provides a framework that allows developers to implement, deploy, test, and debug P4 switch implementations. To create our testbed for the experiments, we used BMV2 with Mininet [min], a popular SDN emulation tool. We ran our experiments on a dedicated server running an Ubuntu Server 16.04 with the following configuration: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz 32 Cores, 128GB of RAM, and 1TB of disk space.

As mentioned in 2, the Safe Reset proxy is a state-of-art solution that has been implemented in the data plane as an anti-spoofing mechanism to mitigate spoofed TCP SYN Flood attacks. Since SACK3 is also implemented in the data plane, we compare it against a switch implementation with a Safe Reset proxy defense mechanism. We implemented both SACK3 and Safe Reset on P4_16 for the BMV2 switch using the v1model. In order to make a fair comparison, we used the same Safe Reset implementation used in SACK3 but applied it to all new TCP connections. Nonetheless, to better compare the overhead caused by both implementations, we also implemented a basic switch implementation that parses Ethernet, IPv4, and TCP, to be used as the baseline that only does packet routing and does not present any DDoS protection. For the implementation of each sketch we choose to use 3 different hash function among the functions already supported in the P4_16 data plane: *crc16*, *csum16*, and *crc32*. The parameters chosen for each sketch (k is the number of hash functions, n is the number of elements, m is the amount of bits used in the sketch, b is the amount of bits used in each sketch counter, and ϵ is the false positive rate) and its total memory usage are listed in Table 1. We choose this parameters to not exceed the maximum false positive rate of 5% while using less the 1MB of memory space for all application.

Table 1. Sketches Parameters

Sketch	k	n	m	b	ϵ	Total Size (KB)
DCBF	3	50000	400000	4	3,05	400
CBF2	3	10000	80000	4	3,05	40
ADCBF	3	50000	400000	4	3,05	400
WBF	2	100000	800000	1	4,89	100

According to Figure 2, the testbed network is composed of one BMV2 switch. The servers are connected to the network using 1 Gbps link, while the other host is connected to the network with 100 Mbps links. In addition to the two servers, there are 31 hosts with 30 potential attackers and one host being the legitimate or benign user. Furthermore, the benign host requests a 10Kb Web page from the server as soon as it finishes the previous

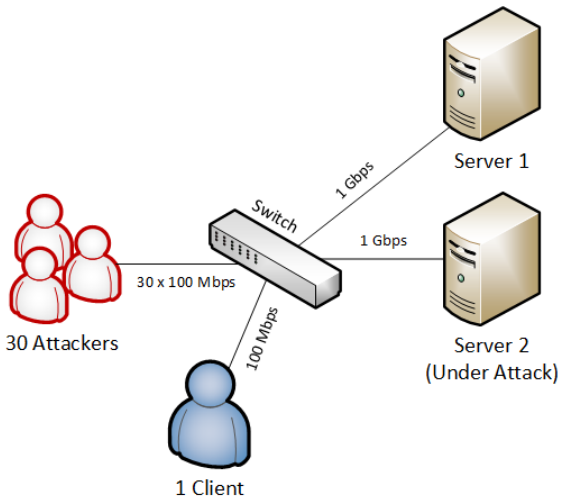

Figure 2. Topology

request. One of the main parameters which affect the efficiency of SACK3 in attack detection is the threshold K. In fact, choosing a proper value for threshold K can improve the efficiency of SACK3, since setting a larger value for K increases the attack detection time. On the contrary, setting a small value might also cause detecting a legitimate user as an attacker. Therefore, we fixed the threshold parameter K to 100, since we do not want to compare how fast the solution can detect a DDoS attack, but instead we want to compare the impact of the anti-spoofing mechanism on the client connection.

Mininet uses a real network stack with small and isolated process that can run real applications to emulate the hosts. Therefore, we define the following setup to the hosts: servers use a simple python server (*SimpleHTTPServer*) to serve a 10Kb Web Page; clients use the *Curl* tool to make legitimate HTTP requests and log the connection measurements; and attackers use *hping3* to generate a flood of spoofed TCP SYN requests to the servers. It is worth to mention two additional configurations in the *curl* tool: (i) we configured the connection timeout, maximum time allowed to get a SYN-ACK response, to 15 seconds in order to run the experiment faster when the server has become unavailable, and (ii) we used *curl* version 7.52 in order to have more precision in the time measurements, from 3 to 6-digit precision, due to the small values of connection time in the basic switch measurements. The testbed topology and tools used in this experiment setup were based on the experiments of OPERETTA [Fichera et al. 2015] and SLICOTS [Mohammadi et al. 2017].

Table 2. Experiments summary

Scenario	Type	Servers	Attackers	Rate (pps)
S1	Normal	1	0	0
S2	DoS	1	1	1000
S3	DDoS	1	30	1000
S4	Variable pps DDoS	2	30	1 to 100000

Table 2 lists each experiment and its parameters. We conducted simulations in four different scenarios using all three implementations of the switch as follows:

- Scenario *S1*: using only one server and one legitimate client, is meant to measure the overhead of each implementation in normal conditions when there are no

ongoing attacks.

- Scenario *S2*: using only one server, one legitimate client and only one attacker generating spoofed TCP SYN packets with a rate of 1000 pps (packets per second), aims to measure how each implementation behaves under a simple DoS attack, that is not enough to make the server unavailable.
- Scenario *S3*: similar to scenario S2, but with 30 hosts as attackers - this scenario is meant to measure how each implementation behaves under a DDoS attack capable of making the server unavailable.
- Scenario *S4*: similar to scenario S3, but varying the spoofed attack rate from 1 to 100000 packets per second (pps) in increments of 10 times the previous value. This scenario is meant to evaluate how each implementation behaves with different DDoS attack rates.

We evaluate the experiments using the following performance metrics: *Connection Time* denotes the time for initiating a TCP connection. More specifically, it denotes the time between a TCP SYN request and the TCP SYN-ACK response; *Download Speed* denotes the overall connection speed in Kbps when downloading a Web Page or a file.

4.2. Experimental Results

In this section we discuss and analyze the results for the two scenarios under different number of attackers and attack rates. First of all, we utilize scenario S1 under normal conditions to measure the overhead of each implementation. Figure 3-A shows the connection times difference, in log scale, among the three implementations. Safe Reset proxy implementation imposes a significant overhead due to the additional delay of one second fixed by the implementation, in comparison to the basic and SACK3 switch implementations. Since scenario S1 is composed of only legitimate client requests, SACK3 does not detect or trigger any defense mechanism that would penalize the legitimate client request. In contrast, Figure 3-B shows a close comparison of the connection times in seconds between the basic switch and SACK3. Results have shown an overhead of 115% in the connection time due to the TCP SYN Flood sketch-based detection mechanisms. Basically, it is the overhead caused by the *Lookup* and *Insertion* operations in the standard and Counting Bloom Filters. However, even with this additional overhead caused by the detection, it is still 890 times faster than Safe Reset.

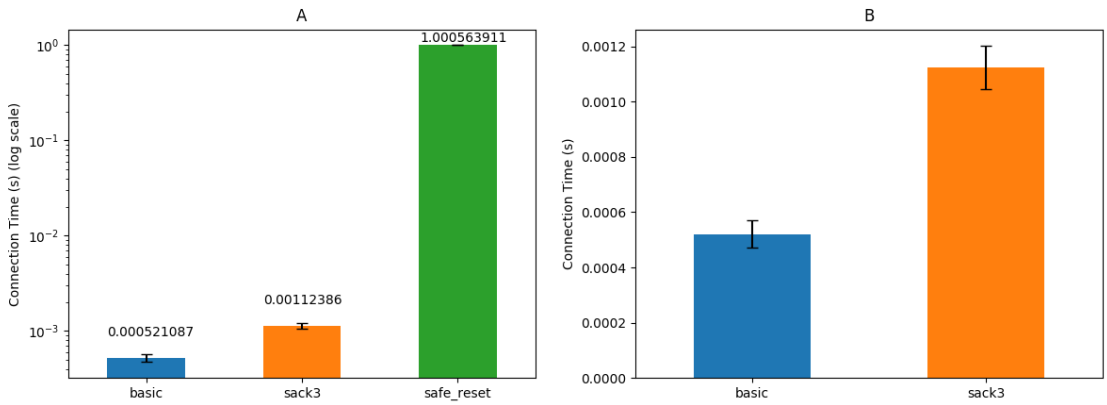

Figure 3. Connection time baseline without an ongoing attack

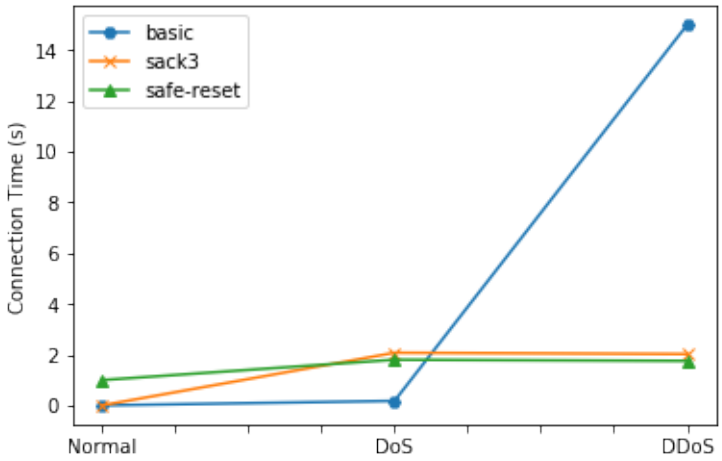

Figure 4. Connection Time in scenarios S1, S2, and S3

Figure 4 illustrates the connection times in scenarios S1, S2, and S3. The measurements were made using a request made to the server under attack. It is possible to notice that, even under the DoS scenario with no protection with the basic switch implementation, the server can still go unaffected. Although, when we scale the number of attackers to 30, making it a DDoS attack, we observe that all connection times goes to the connection timeout of 15 seconds as specified in the *curl* tool. Meaning that, under scenario S3, we can successfully attack the server making it unavailable. SACK3 presents a similar performance to Safe Reset, since in scenarios S2 and S3 it is able to successfully detect the ongoing attack to the server and apply the counter-measurement, in this case, authenticating the client with the same Safe Reset proxy mechanism. Therefore, SACK3 presents better connection times for legitimate users while maintaining the same level of protection.

In scenario S4 we measured legitimate client connection times for two different servers: 1) a server under attack as the previous experiments, and 2) a normal server not under attack while there is an ongoing attack in parallel. With this experiment we want to validate that SACK3 will have a better performance than Safe Reset for servers not under attack, while an ongoing attack on other servers is happening in the network. Figure 5 shows the connection time difference, in log scale, between SACK3 and Safe Reset when under a small DDoS attack. When clients connect to a normal server not under attack while an attack is ongoing in the network, SACK3 can maintain a small overhead similar to scenario S1 where there are no ongoing attacks.

In order to evaluate how the overhead in each switch implementation change over different attack rates, we also variate the rate in pps of Spoofed TCP SYN packets that each host generates. In Figure 6 it is possible to verify that the server under attack starts to become unavailable when the attack reaches 1000 pps. While the basic switch implementation does not provide any protection, SACK3 and Safe Reset maintain similar connection times over the increase in the attack rates. A spike may be observed when reaching the maximum attack rate. It happens due to the overhead of the detection mechanism that becomes more significant at higher attack rates. Although, since all experiments are performed using emulated software switches, this overhead may not be significant when deployed in a hardware data plane. Nevertheless, SACK3 always performs better than

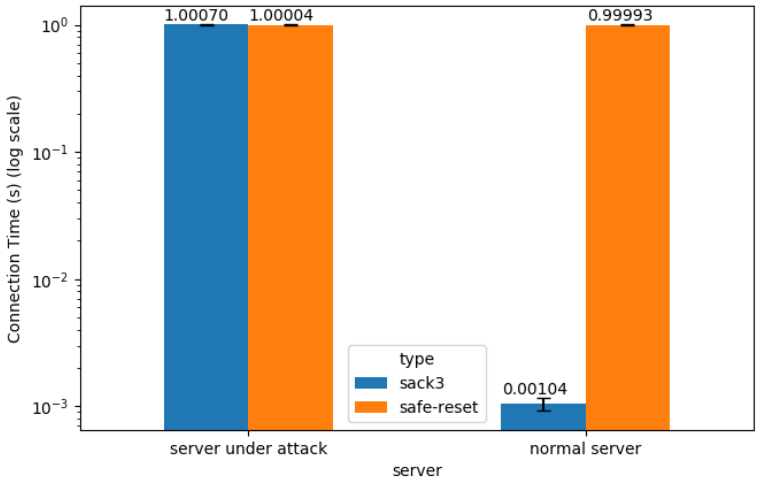

Figure 5. Connection time over 1 pps attack

Safe Reset for legitimate client connections to a normal server, or even with an ongoing attack on the network.

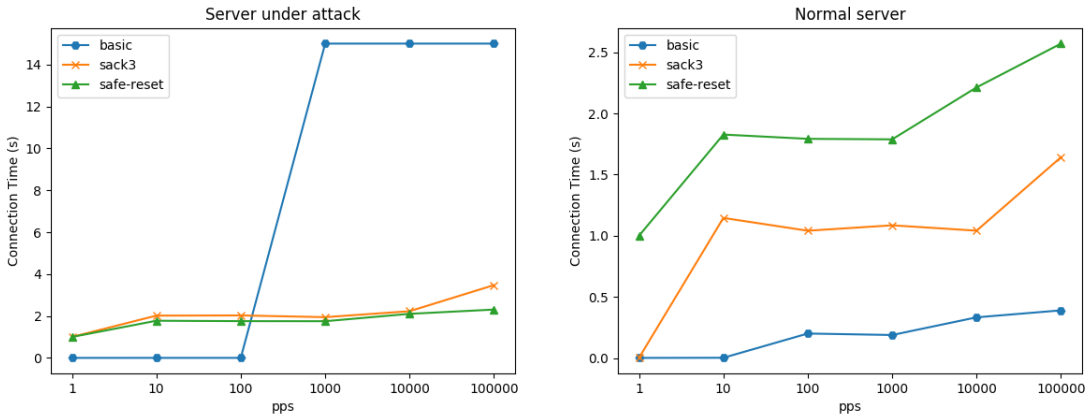

Figure 6. Connection time over different rates of DDoS attack

Finally, to better understand the impact of each switch implementation on the overall connection, we measured the download speed of a fixed 10Kb Web Page during scenario S1. Figure 7 shows that SACK3 has better performance if compared to Safe Reset. The significant decrease in download speed on the Safe Reset implementation is mainly due to its initial connection delay.

5. Conclusion

Several works explored the flexible management control provided by the SDN architecture to produce a flexible and powerful defense system against TCP SYN Flood attacks. Among them, data plane based solutions aim to leverage the hardware speed and the recent flexibility of programmable switches. Due to the limited resources on the data plane, defense system implements techniques that rely on authenticating the client before forwarding the request to the server. However, these techniques, such as Safe Reset, comes with an additional overhead to the client's connection even without an ongoing attack.

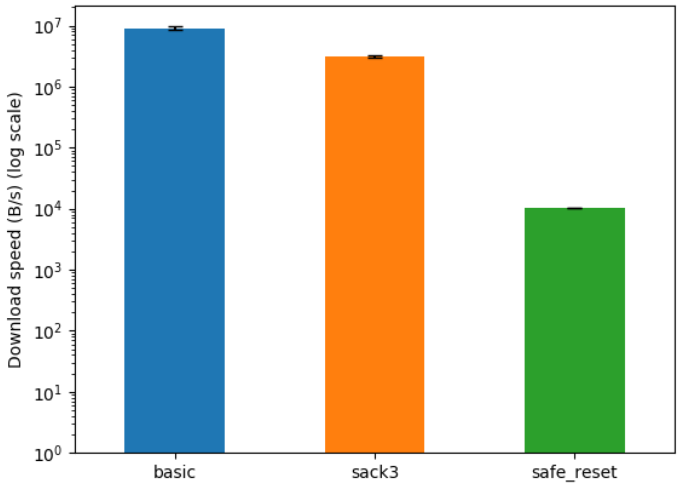

Figure 7. Download speed without an ongoing attack

To solve this problem, we proposed SACK3 that relies on sketches, probabilistic data structures, to detect ongoing attacks with a small usage of the hardware resources. By detecting the attacks, we can only apply these authentication techniques to suspicious clients that connect to servers under attack. Results have shown that our solution allowed clients that connect to normal servers to have only a small overhead on the connection time and download speed if compared to Safe Reset. While in experiments under DoS and DDoS attacks it presents a performance similar to the Safe Reset defense technique. Future work includes the use of other sketch-based algorithms to track suspicious clients to reduce resource consumption and increase accuracy, validate SACK3 on P4 hardware platforms (e.g. NetFPGA), and explore other tests scenarios for TCP SYN Flood attacks.

References

Mininet. http://www.mininet.org/. (visited on Mar. 10, 2018).

Afek, Y., Bremler-Barr, A., and Shafir, L. (2017). Network anti-spoofing with sdn data plane. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE.

Ambrosin, M., Conti, M., De Gaspari, F., and Poovendran, R. (2017). Lineswitch: tackling control plane saturation attacks in software-defined networking. *IEEE/ACM Transactions on Networking*, 25(2):1206–1219.

Braga, R., Mota, E., and Passito, A. (2010). Lightweight ddos flooding attack detection using nox/openflow. In *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, pages 408–415. IEEE.

Chen, W. and Yeung, D.-Y. (2006). Defending against tcp syn flooding attacks under different types of ip spoofing. In *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, pages 38–38. IEEE.

Dhawan, M., Poddar, R., Mahajan, K., and Mann, V. (2015). Sphinx: Detecting security attacks in software-defined networks. In *NDSS*.

Dodig, I., Sruk, V., and Cafuta, D. (2017). Reducing false rate packet recognition using dual counting bloom filter. *Telecommunication Systems*, pages 1–12.

Dzurenda, P., Martinasek, Z., and Malina, L. (2015). Network protection against ddos attacks. *International Journal of Advances in Telecommunications, Electrotechnics, Signals and Systems*, 4(1):8–14.

Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. (2015). Bohatei: Flexible and elastic ddos defense. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 817–832.

Fichera, S., Galluccio, L., Grancagnolo, S. C., Morabito, G., and Palazzo, S. (2015). Operetta: An openflow-based remedy to mitigate tcp synflood attacks against web servers. *Computer Networks*, 92:89–100.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., and Maglaris, V. (2014). Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Computer Networks*, 62:122–136.

Kompella, R. R., Singh, S., and Varghese, G. (2004). On scalable attack detection in the network. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 187–200. ACM.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Martinasek, Z. (2015). Scalable ddos mitigation system for data centers. *Advances in Electrical and Electronic Engineering*, 13(4):325.

McKeown, N. (2009). Software-defined networking. *INFOCOM keynote talk*, 17(2):30–32.

Mohammadi, R., Javidan, R., and Conti, M. (2017). Slicots: an sdn-based lightweight countermeasure for tcp syn flooding attacks. *IEEE Transactions on Network and Service Management*, 14(2):487–497.

Radware (2016). 2017-2018 global application network security report. *URL https://www.radware.com/ert-report-2017*. (visited on Dec. 10, 2017).

Shin, S., Yegneswaran, V., Porras, P., and Gu, G. (2013). Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 413–424. ACM.

Sun, C., Hu, C., Zhou, Y., Xiao, X., and Liu, B. (2009). A more accurate scheme to detect syn flood attacks. In *INFOCOM Workshops 2009, IEEE*, pages 1–2. IEEE.

Wang, H., Zhang, D., and Shin, K. G. (2002). Detecting syn flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1530–1539. IEEE.

Xing, T., Huang, D., Xu, L., Chung, C.-J., and Khatkar, P. (2013). Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, pages 89–92. IEEE.

YuHunag, C., MinChi, T., YaoTing, C., YuChieh, C., and YanRen, C. (2010). A novel design for future on-demand service and security. In *Communication Technology (ICCT), 2010 12th IEEE International Conference on*, pages 385–388. IEEE.