A SYN FLOODING DDoS ATTACK DETECTION IN P4-BASED PROGRAMMABLE NETWORKS

Calin-Marian IURIAN, Daniel ZINCA, Iustin-Alexandru IVANCIU,
Tudor-Mihai BLAGA, Virgil DOBROTA
Communications Department, Technical University of Cluj-Napoca, Romania
Marian.Iurian@com.utcluj.ro; Daniel.Zinca@com.utcluj.ro; Iustin.Ivanciu@com.utcluj.ro;
Tudor.Blaga@com.utcluj.ro; Virgil.Dobrota@com.utcluj.ro

Abstract: Recent studies related to Distributed Denial-of-Service (DDoS) attacks were focused on determining how to leverage data plane programmability, enabled by P4 language, to identify intrusions directly in network switches, without the involvement of Software-Defined Networking (SDN) controllers. This facilitates the detection of TCP SYN flooding, for instance, without changing the flow paths through a central controller, therefore increasing the speed for managing workloads. In this paper, we propose a novel mechanism by introducing the capability of P4-based data planes to implement real-time detection and to respond effectively to attacks that require stateful functionalities (e.g., port scan).

Keywords: *DDoS, P4, SDN, TCP SYN flooding, TCP SYN port scan.*

I. INTRODUCTION

The more available information related to a wide area of activity domains becomes, the more complex the information systems and Tactics, Techniques and Procedures (TTPs) used by the attackers are.

Denial-of-Service (DoS) attacks have increased significantly since a higher number of devices access the Internet of Things (IoT) and the businesses now support the remote connectivity technologies to complement existing infrastructure. No matter their size, organizations frequently disregard asset and inventory management guidelines that would enable them to fully comprehend their security vulnerabilities. Furthermore, IoT devices commonly use default passwords and do not possess strong security policies, leaving them open to hacking and to other forms of exploitation. Users are typically unaware that gadgets are infected, and an attacker may simply access hundreds of thousands of these devices to launch a significant attack.

As presented in [1], Distributed Denial-of-Service (DDoS) have grown in volume, length, and complexity during the past few years. According to the Q2 2022 DDoS report, SYN flood attacks accounted for 53% of all network-based attacks, thus remaining the primary attack method today. Their functionality is based on exploiting the stateful TCP handshake's initial connection request. During the initial request, servers do not have any information about the new TCP connection, and without conventional protection they may find it difficult to mitigate a flood of initial connection requests. As long as the "Three-way Handshake" procedure is not terminated or canceled when it reaches the second phase, the connection is considered to be "Half open" (the server is awaiting the last ACK). A "Half open connection" occurs when the destination machine has already transmitted the SYN+ACK segment and is awaiting the arrival of the last ACK segment in order to establish the connection. The destination will wait a certain amount of time for the last ACK segment to arrive. During this time, the information about the upcoming connection will be registered and processed. This step will require a particular amount of memory and processing resources of the destination system.

Software-Defined Networking (SDN) is regarded as a next-generation network architecture. The main feature of SDN is that it employs a centralized controller to decouple the data plane from the control plane. The security of the network and the traffic rules can be easily implemented using a centralized controller, which acts as a network administrator who monitors the incoming and outgoing traffic. As a result, the controller has a global view of the network and, with ready-to-use resources, it is possible to build complex algorithms for network management (e.g., network security, congestion control, load balancing). The rapid and accurate identification and mitigation of abnormal behavior, such as DDoS attacks, is critical for network security. Nonetheless, traffic must be routed through a controller equipped with a DDoS attack detection mechanism. In addition to a substantial demand of processing resources for DDoS attack detection techniques, in general, classifying all flows in a controller will surpass the usable processing power, especially for flooding-based attacks [2].

The P4-based programmable data plane (Figure 1) can prevent large-scale attacks with low overhead, it can detect attacks per packet, and it can react to constantly changing threats with strong performance programmable hardware switches. This facilitates DDoS attack detection without changing the flow paths through a central controller, and thus enhancing speed and workload management.

In this work, we propose a novel approach for DDoS attack detection of TCP SYN flooding, by leveraging the potential of a TCP SYN (half-open) scan on the target host. Since this will respond with a RST segment if the port is closed, or with a SYN+ACK segment if the port is open, it is not necessary to finish the three-way handshake with the

Manuscript received October 29, 2022; revised November 30, 2022

final ACK reply. As a result, just half of the connection to the target port is made in order to determine its status.

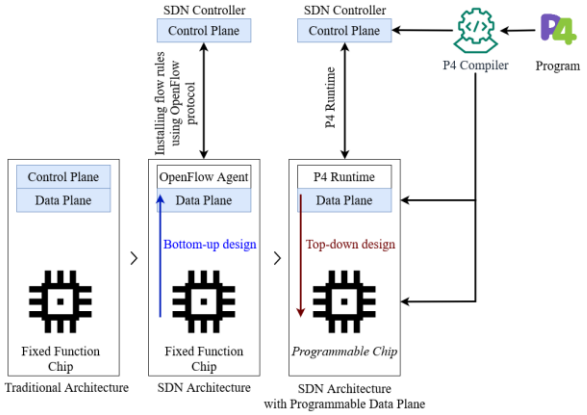

Figure 1. Evolution from the traditional architecture to the mixed SDN model with programmable data planes [3]

II. RELATED WORK

The effects of DDoS assaults and potential security strategies have been the subject of several research papers in recent years. In this part, we address related papers on DDoS defensive techniques in SDN.

One of the solutions for securing the network traffic across multiple environments is presented in [4]. The authors describe the current state of programmable switches and the opportunities and challenges encountered when mitigating DDoS attacks. For this reason, they developed a rapid and cost-efficient protection system that overcomes the major constraints of today's security.

Another defense system for detecting DDoS attacks is proposed in [5].

The Floodlight controller uses the sFlow RT [6] in order to develop the modules needed for attack detection and for blocking applications. The proposed defensive system classifies normal and attack traffic using the Support Vector Machine (SVM) approach. According to the experimental results, it provides a remarkable accuracy of 96.55%.

In [7], the authors propose a DDoS attack detection solution for a vehicular network. For this purpose, the suggested system uses machine learning (ML) and SDN characteristics. It is based on a *Packet_In* based trigger method that reduces the attack detection time. It also incorporates feature extraction utilizing the flow table's entries. A classification model is trained and its performance is evaluated using the retrieved features. According to the data, the suggested solution reduces the false alarm rate and attack detection time.

The impact of spoofed and non-spoofed TCP-SYN flooding attacks on the controller resources in SDN is discussed in detail in [8]. The authors highly suggest an intrusion detection solution based on machine learning. The traffic is classified by using five distinct categorization models that are assessed with various performance metrics. These models are validated using a cross-validation approach which efficiently categorizes the traffic. The experiments demonstrate that all of the categorization models that were taken into consideration performed significantly better.

Study [9] considers the P4 technology in the context of a multi-layer SDN over an optical network. It demonstrates that effective dynamic traffic engineering (TE) actions can be deployed at the data plane level without engaging the SDN controller due to the stateful capacity of P4 nodes. As a second use case, the authors make use of the P4 technology to enforce cyber-security [10], demonstrating the capability to respond to cyberattacks effectively without the need for specialized hardware, such as firewalls. They replaced the SDN controller to manage the traffic and to take decisions on flow entry changes, as is the case with OpenFlow switches. P4 domain specific targets are now used to directly decide particular actions on the switch itself [11]. The P4 implementations have undergone experimental validation on BMv2 P4 switches, demonstrating excellent performance in terms of scalability with the size of the program and in terms of switch latency to carry out operations.

Networks frequently experience harmful attacks like TCP SYN flooding and UDP flooding. The attacks block the network in addition to using up a lot of the target server's resources and network transfer rate. In SDN, several new and innovative defense techniques are put out to protect the network against the attacks mentioned before. Still, the majority of protection mechanisms only work against attacks that use a particular protocol, such TCP or UDP. As a result, the authors of [12] provide a generally applicable defense system in P4-based SDN. The experimental results demonstrate how successfully the suggested protection mechanism may liberate server resources (six times quicker than the associated work after identifying the threat for SYN flooding). In comparison to previous studies, the suggested protection system can minimize the volume of malicious traffic for UDP flooding by around two thirds.

A P4-programmable switch-based SYN flooding DDoS attack detection mechanism is provided in [13]. The detection of the malicious process is moved from a centralized controller to programmable P4 switches, which reduces detection time and distributes workload across the network. The active detection system extends passive classification techniques and seizes SYN flooding DDoS assaults by selective packet dropping. As a result, it is expected to have a more precise detection under overloaded network circumstances compared to the legacy solutions.

In [14], the authors compare Standalone and Correlated DDoS Attacks Detection (DAD) architectures while evaluating ML-assisted DDoS attack detection frameworks for use in SDN environments. Applying the data-plane programmability that the P4 language makes possible, the authors studied how detection latency is decreased when features extraction is carried out at P4 switches level. To achieve this, they evaluated the accuracy and computational efficiency of several ML classifiers and deployed the algorithms in a real-time environment where the P4 switch provided the ML algorithm with various sorts of data, including packet mirroring, header mirroring, and P4-metadata extraction. Using P4 for feature extraction, the results demonstrate that the attack detection can be accomplished with classification accuracy, precision, and a F1-score higher than in 98% of cases. Additionally, in the situation where P4 is employed for features extraction, there is a significant time decrease, down to less than 200 µs.

III. OVERVIEW OF P4 LANGUAGE AND PROGRAMMABLE SWITCHES

As described in [15], the Programming Protocol-Independent Packet Processors (P4) is a declarative programming language used to specify how packets are handled by a network forwarding component, such as NIC, switch, router, or a service-oriented appliance. It is built on an abstract forwarding architecture (Figure 2) that consists of a parser and a group of match + action table elements that are classified into ingress and egress.

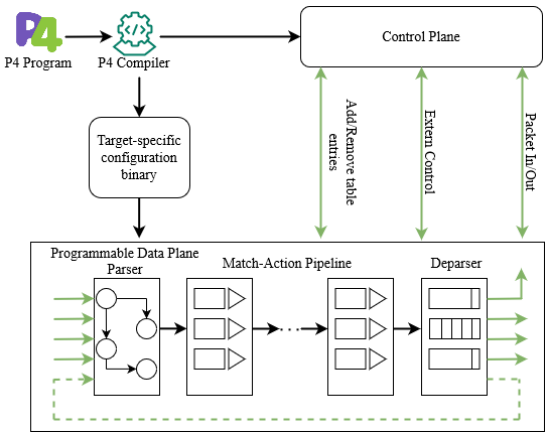

Figure 2. P4 abstract forwarding architecture

A standard P4 program consists of five fundamental components:

- Header: defines the packet header structure
- Parser: the granted header series within packets
- Table: for identifying the appropriate fields and the matching action
- Action: creates a set of instructions
- Flow controller: describes the way tables are organized inside the pipeline and how packets move through it.

Next, P4 applications are compiled into configuration binaries using the compiler. Its task is split into two phases: mapping the target independent intermediate representation (IR) to a particular target and translating P4 programs into IR.

As mentioned in [15], the compiler integrates two fundamental functions: (1) development of runtime mapping metadata to enable the control and data planes to interact using P4 runtime; and (2) creating an executable file for the particular data plane, describing the header format and the appropriate procedure.

P4 programmable nodes, also known as targets, can be acquired using either specialized hardware or CPU-based software (e.g., FPGA, ASIC). Described as a programming paradigm, the P4 architecture offers a functional perspective of the processing based on specific target. Therefore, target producers implement different P4 architectures. For instance, the SimpleSumeSwitch architecture is used by NetFPGA-based devices, the V1Model architecture is used by software switches such as BMv2, whilst the Protocol-Independent Switching Architecture (PISA) architecture is implemented by programmable switches such as Intel Tofino ASIC.

The workflow of a P4 program is divided into several control blocks, so the input packet is processed and then transferred to the following control block in the pipeline. For example, the PISA architecture describes data processing by introducing three main blocks:

- Parser: it describes how to interpret the packet's header. A finite state machine is used to illustrate the steps, and each state is in charge of extracting one header structure.
- Match-Action Pipeline: it has several match-action tables (MAT), each one containing keywords and related action information. These tables compare packets based on various header attributes and specify the actions to take when a match is found.
- Deparser: it arranges the updated headers into packets in a fixed order and transfers them to the next phase.

The final block within Figure 2 is represented by an Application Programming Interface (API) called P4 Runtime [16], and it is used to interconnect the control plane and data plane. It protects the hardware characteristics of the data plane and is unaltered by the capabilities and the protocol that the data plane provides.

According to [17], the primary characteristics of programmable switches are:

- Agility: the capability of designing, testing and embracing new protocols and components in less time.
- Top-down design: In comparison with the traditional bottom-up approach, where all the fixed-function ASICs are located at the bottom, the programmable switches define protocols and features in the ASICs. It is worth noting that the physical layer and certain MAC layer components might not be programmable.
- Visibility: Programmable switches offer a better view of how the network is operating. Without the involvement of the control plane, In-band Network Telemetry (INT) is an example of a framework for collecting and recovering data from the data plane.
- Reduced complexity: Fixed-function are less difficult since they accommodate a wide set of protocols. The processing mechanism, which is hard-coded in silicon, becomes more sophisticated and uses resources as resources as a result of these protocols. Engineers can choose to implement only the necessary protocols using programmable switches.
- Separation: The protocols and unique features can be implemented without the knowledge of the chip manufacturer.
- Improved performance: The performance is enhanced since programmable switches do not degrade the efficiency of the network. Instead, they might operate more effectively than fixed-function switches. For a better understanding,

Table 2 compares Intel Tofino2 programmable switches [18] to fixed-function switches. Based on common features, the P4-based switches present the following benefits: 14% greater performance, 14% lower power, better burst absorption, and real-time visibility.

Technology	Programmable Switches	Fixed-function ASICs
Technology	16nm	16nm
L2/L3 Throughput	6.5 Tb/s	6.5 Tb/s
Availability	Yes	Yes
Max forwarding rate	4.8B packets/sec	4.2B packets/sec
Max 25G/10G Ports	256/258	128/130
Programmability	Yes	No
Typical system power draw	4.2W per port	4.9W per port
Large scale NAT	Yes (100k)	No
Large scale stateful ACL	Yes (100k)	No
Large scale tunnels	Yes (192k)	No
Packet buffer	Unified	Segmented
LAG/ECMP Hash algorithm	Full entropy, programmable	Hash seed, reduced entropy
ECMP	256-way	128-way
Telemetry and analytics	Line-rate per low stats	Sflow (Sampled)

Table 1. Comparison between a P4 programmable switch and a fixed-function switch [18]

IV. IMPLEMENTATION

When dealing with network attacks, the programmable data plane features three main benefits over OpenFlow: visibility per packet, scalability, and high-speed processing capabilities. As shown in Figure 3, traditional security applications and defense mechanisms can be implemented at the hardware level including access control lists (ACLs) and packet filtering techniques, whereas higher-level firewalls and deep packet inspection tactics must be implemented by software [19]. This is also a major reason why traditional switches must transmit sampled packets to the controller. The programmable switch can handle every layer's protocol fields at the hardware level, provide deep packet analysis and protocol field change, and enable customized protocols and complex defensive mechanisms.

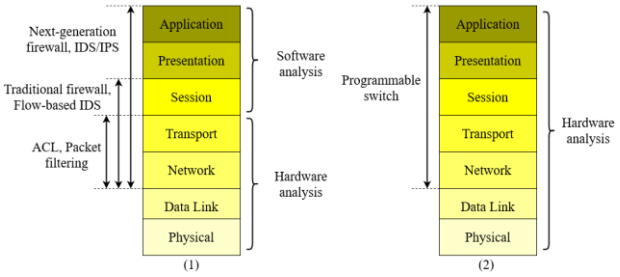

Figure 3. OSI model for security: (1) traditional devices; (2) programmable devices

In order to mitigate the DDoS attack, the SYN scanning approach is proposed as a strategy to determine the status of a communications port without establishing a complete connection. Our approach uses the TCP SYN scan, which is a version of the standard SYN scan. It is a fast and effective scan that is not blocked by firewalls since it never completes the entire TCP connection. As a result, TCP SYN scanning is also known as half-open scanning and can identify open, filtered, or closed port status.

In our work, the attacker used TCP SYN to do the port scan. P4 switch detected it and blocked the packets sent from H1. The detection method proposed and tested herein blocked the connection when the number of SYN segments minus the number of SYN+ACK segments was higher or equal to 3 (see Figure 4).

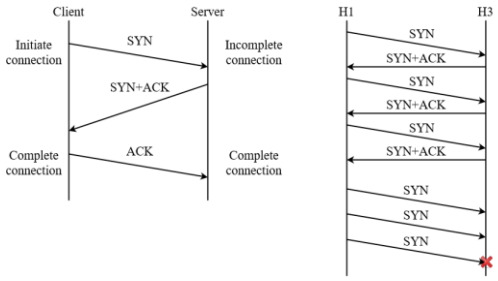

Figure 4. Diagram of TCP SYN flooding detection

For our experiments, we used Mininet as a working environment. The P4 language is the vital component of the P4-based solution, mainly due to the capacity of enabling the specification of packet formats (protocol headers) to be recognized by the P4 switch (BMv2 [20]), as well as of the actions to be taken on arriving packets (forwarding, headers modification, adding protocol header, etc.). The workflow used to program the BMv2 switch is illustrated in Figure 5:

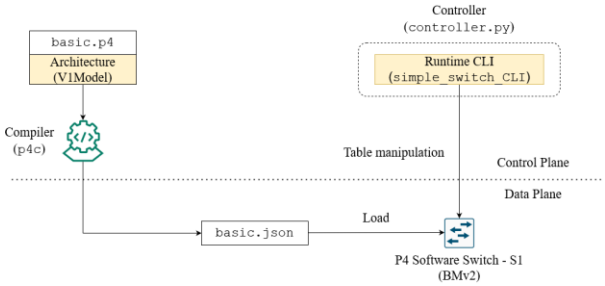

Figure 5. Workflow of BMv2 switch

The P4 program (`basic.p4`) was divided into three parts: protocols definition (data declaration), parser logic (parser and deparser) and a series of control blocks with Match-Action tables. The first section describes the protocol headers that will be recognized by the switch (S1), i.e. the IPv4 header (see Figure 6).

```
header ipv4_t {
    bit<4>    version;
    bit<8>    diffserv; (…)
```

Figure 6. IPv4 header

We only specified the header fields and their length. The headers are used to parse the incoming data and to identify the type of the packet. The Parser Logic is a finite state machine that defines the stages involved in reading and parsing incoming packets. It may be seen as a cyclic network in which each node processes a protocol's header. Additionally, we defined a number of control blocks in the P4 program that contain Match-Action tables. See the routing table in Figure 7.

By reading the destination IPv4 address we checked the match using the Exact comparison technique [21].

```
    table routing_table {
        key = {
            hdr.ipv4.dstAddr: exact;
        }
        actions = {
            forward;
            drop;
        }
        size = 1024;
        default_action = drop();
    } (…)
```

Figure 7. IPv4 routing table

Then, for the packets that fit the rule, we performed either forwarding or dropping. The final step was to define the Deparser, which determined the order of packet headers for outgoing packets, as in Figure 8.

```
control deparser(packet_out packet, in headers
hdr) {
    apply {
        packet.emit(hdr.ethernet);
        packet.emit(hdr.ipv4);
        packet.emit(hdr.tcp);
        packet.emit(hdr.udp);
    }
```

Figure 8. Deparser

At this point, we only deployed the P4 switch's data plane, so we used Python to develop a simple controller (`controller.py`) to listen to the packets being transmitted from the data plane. After the network started, we ran the controller script to populate the forwarding table of S1. The controller was based on the *Thrift API [22]*, and it was used with any P4 switch. The implementation of the *Thrift API* was built on the `SimpleSwitchThriftAPI` (as in Figure 9).

```
if (syn1-syn2>=3) and (TCP in pkt) and
pkt[TCP].flags==2:
  src = pkt.sprintf('{IP:%IP.src%}')
  if src not in blockip:
   self.controllers["s1"].table_add("block_pkt",
"_drop", [str(src))], [])
   blockip.append(src)
```

Figure 9. `SimpleSwitchThriftAPI`

The `syn1` value represented the number of SYN packets sent by the attacker, while the `syn2` was the number of SYN+ACK packets sent by the server. If the condition was true, then the controller applied the `drop packet` rule and sent it to the switch for blocking the incoming packets from H1.

V. EXPERIMENTAL RESULTS

The following section illustrates the capacity of the P4 switch to block the incoming packets sent from the attacker (Figure 10). We started from the implementation described in [23], although their code was designed for investigating Denial-of-Service (DoS) only. Note that our paper is focused on DDoS, and as the work is in progress, herein we present just the preliminary results.

After the P4 program was initialized, the detection mechanism began. As mentioned earlier, this mechanism was implemented with the BMv2 software switch (see Figure 11).

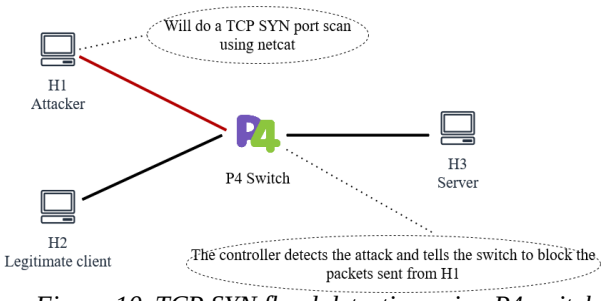

Figure 10. TCP SYN flood detection using P4 switch

```
simple_switch: no process found
Building mininet topology.
p4c --target bmv2 -arch v1model -std p4-16
"basic.p4"
Switch port mapping:
s1: 1:h1  2:h2  3:h3  4:sw-cpu
```

Figure 11. BMv2 switch in Mininet

In order to verify the network connectivity, H1 performed a ping test to H3, as depicted in Figure 12. Because no attack was detected by the controller, the `syn1` and `syn2` values were 0.

```
mininet> h1 ping -c 3 h3
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data.
64 bytes from 10.0.3.1: icmp_seq=1 ttl=63
time=1.92 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=63
time=2.22 ms

--- 10.0.3.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet
loss, time 1009ms
rtt min/avg/max/mdev = 1.921/2.073/2.225/0.152
ms
```

Figure 12. Server ping test

H3 was set up as a server using the command `python -m SimpleHTTPServer`. H1 tried to do a port scan on H3 (Figure 13) using Netcat (`nc`) utility program.

```
mininet> h1 nc -nvz -w 1 h3 80-85
nc: connect to 10.0.3.1 port 80 (tcp) failed:
Connection refused
nc: connect to 10.0.3.1 port 81 (tcp) failed:
Connection refused
nc: connect to 10.0.3.1 port 82 (tcp) failed:
Connection refused
nc: connect to 10.0.3.1 port 83 (tcp) failed:
Connection refused
nc: connect to 10.0.3.1 port 84 (tcp) timed out:
Operation now in progress
nc: connect to 10.0.3.1 port 85 (tcp) timed out:
Operation now in progress
```

Figure 13. H1 port scan on H3

The controller detected the attack, and sent the `drop packet` rule (Figure 14) to the P4 switch to block H1.

```
interface: s1-cpu-eth1
summary: Ether / IP / TCP 10.0.1.1:46416 >
10.0.3.1:82 S
count1[ 10.0.1.1, 10.0.3.1 ]= 3
syn1: 3 syn2: 0 syn3: 0 total: 3
Adding entry to exact match table block_pkt
match key:           EXACT-0a:00:01:01
action:              _drop
```

Figure 14. The controller set the rule "drop packet"

When H1 tried again to ping H3, the switch dropped all the packets since it added the controller rule to block the connection coming from the attacker (Figure 15).

```
mininet> h1 ping -c 3 h3
PING 10.0.3.1 (10.0.3.1) 56(84) bytes of data.

--- 10.0.3.1 ping statistics ---
2 packets transmitted, 0 received, 100% packet
loss, time 1010ms
```

Figure 15. H1 blocked IP address

VI. CONCLUSIONS

In order to minimize the disruptions caused by DDoS assaults, malicious traffic had to be detected and mitigated. In this paper, we presented a TCP SYN flooding attack detection system that was implemented using P4-programmable switches. Unlike the related work, P4 was used to effectively respond to attacks that required stateful behaviors. In this case, the attacker wanted to do a port scan to find the potential open ports on the target host, before launching the attack. The implementation was experimentally evaluated using BMv2 P4 switch. It showed a notable scalability with increasing P4 program size and in terms of switch latency with conducting P4 actions. We proved that using this strategy in the data plane rather than in an SDN centralized controller, we obtained a reliable detection. Multiple vulnerabilities have to be addressed too in a future development. For example, the detection system must support a higher number of attackers and a wider range of DDoS attack types. Because the current architecture was implemented in the Mininet emulator, we plan to speed up the processing power. This could be done by directly compiling the P4 program in a dedicated hardware, hosted either on-site, either in a cloud. We need to carry out security experiments in a more complex network architecture using realistic traffic.

ACKNOWLEDGMENT

This paper was financially supported by the Project "Entrepreneurial competences and excellence research in doctoral and postdoctoral programs - ANTREDOC", project co-funded by the European Social Fund financing agreement no. 56437/24.07.2019.

REFERENCES

[1] "DDoS attack trends for 2022 Q2", Cloudflare 2022, [Online], Available: https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/.
[2] Q. Niyaz et al. "A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN)." EAI Endorsed Trans. Security Safety 4 (2017): e2.
[3] "P4 Language", PLVision, 2021 [Online], Available: https://plvision.eu/expertise/sdn-nfv/p4.
[4] M. Zhang, et al., "Poseidon: Mitigating Volumetric DDoS Attacks with Programmable Switches", Network and Distributed Systems Security (NDSS) Symposium, 23-26 February 2020, San Diego, CA, USA, ISBN 1-891562-61-4, Available: 10.14722/ndss.2020.24007.
[5] J. Liu, Y. Lai, and S. Zhang, "FL-GUARD: A Detection and Defense System for DDoS Attack in SDN". In Proceedings of the 2017 International Conference on Cryptography, Security and Privacy (ICCSP '17). Association for Computing Machinery, New York, NY, USA, 107–111. https://doi.org/10.1145/3058060.3058074
[6] "sFlow-RT", InMon 2022, [Online]. Available: https://sflow-rt.com/.
[7] Y. Yu, L. Guo, Y. Liu, J. Zheng and Y. Zong, "An Efficient SDN-Based DDoS Attack Detection and Rapid Response Platform in Vehicular Networks," in IEEE Access, vol. 6, pp. 44570-44579, 2018, doi: 10.1109/ACCESS.2018.2854567.
[8] R. Swami, M. Dave, V. Ranga, "Detection and Analysis of TCP-SYN DDoS Attack in Software-Defined Networking", Wireless Pers Commun 118, 2295–2317, 2021, Available: https://doi.org/10.1007/s11277-021-08127-6
[9] F. Paolucci, F. Cugini and P. Castoldi, "P4-based Multi-Layer Traffic Engineering Encompassing Cyber Security," 2018 Optical Fiber Communications Conference and Exposition (OFC), 2018, pp. 1-3.
[10] Y. Afek, A. Bremler-Barr and L. Shafir, "Network anti-spoofing with SDN data plane," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057008.
[11] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini and P. Castoldi, "Experimental Demonstration of Segment Routing," in Journal of Lightwave Technology, vol. 34, no. 1, pp. 205-212, 1 Jan.1, 2016, doi: 10.1109/JLT.2015.2473656.
[12] Z. -Y. Shen, M. -W. Su, Y. -Z. Cai and M. -H. Tasi, "Mitigating SYN Flooding and UDP Flooding in P4-based SDN," 2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS), 2021, pp. 374-377, doi: 10.23919/APNOMS52696.2021.9562660.
[13] P. Golchin, L. Anderweit, J. Zobel, R. Kundel, R. Steinmetz, "In-Network SYN Flooding DDoS Attack Detection Utilizing P4 Switches", Accessed: October 03, 2022. [Online]. Available: https://tinyurl.com/y9hm6nf9.
[14] F. Musumeci, "Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Networks", Journal of Network and Systems Management (2022) 30:21, https://doi.org/10.1007/s10922-021-09633-5
[15] Y. Gao, Z. Wang, "A Review of P4 Programmable Data Planes for Network Security", Hindawi Mobile Information Systems Volume 2021, Article ID 1257046, 24 pages, https://doi.org/10.1155/2021/1257046
[16] N. McKeown, T. Sloane, and J. Wanderer, "P4 Runtime-Putting the Control Plane in Charge of the Forwarding Plane", 2017, https://p4.org/api/p4-runtime-putting-the-control-plane-in-charge-of-theforwarding-plane.html.2017.
[17] A. Shapiro, "P4-programming data plane use-cases." in P4 Expert Roundtable Series, [Online]. Available: https://tinyurl.com/y5n4k83h.
[18] P. Kennedy, "Intel Tofino2 Next-Gen Programmable Switch Detailed", ServeTheHome 2022, https://www.servethehome.com/intel-tofino2-next-gen-programmable-switch-detailed/.
[19] E. F. Kfoury, J. Crichigno and E. Bou-Harb, "An Exhaustive Survey on P4 Programmable Data Plane Switches: Taxonomy, Applications, Challenges, and Future Trends," in IEEE Access, vol. 9, pp. 87094-87155, 2021, doi: 10.1109/ACCESS.2021.3086704.
[20] "Behavioral Model (bmv2)", GitHub 2022, [Online]. Available: https://github.com/p4lang/behavioral-model.
[21] L. Wang, "Advanced Computer Networks. Programmable Data Plane", [Online]. Available: https://linwang.info/docs/acn21/lec10-pdp.pdf.
[22] "P4-Utils API reference", Networked Systems Group 2021, [Online], Available: https://nsg-ethz.github.io/p4-utils/p4utils.utils.thrift_API.html.
[23] C.-H. Ke, "Anti TCP SYN Port Scan", Department of Computer Science and Information Engineering, National Quemoy University, Kinmen, Taiwan, 2020 [Online].