uc3m | Universidad Carlos III de Madrid

Master Degree in Cybersecurity
2021-2022

Master Thesis

Multidimensional DoS Attacks Detection in SDN Using Sflow with an Entropy Approach

Iván Gallardo Romero

Tutor: Pedro Revidiego Vasallo

Madrid, September 2022

Multidimensional DoS Attacks Detection in SDN Using Sflow with an Entropy Approach

Iván Gallardo Romero

***Abstract*—Presently, as the volume and complexity of data on the Internet increase exponentially, Software Defined Networks (SDN) are becoming increasingly common. SDN is an agile and flexible way to adapt to growing network requirements and enable automation and agility. Nevertheless, software-defined networks have more security challenges than traditional networks, Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are among the main challenges. In order to detect these types of attacks, it is advantageous to use sFlow. sFlow, short for "sampled flow", is a standard for packet export that enables visibility of all the infrastructure. Therefore, it allows for the detection of multi-vector and multi-layer attacks. After flow acquisition, an entropy-based algorithm has the flexibility to be used to classify malicious traffic and regular traffic. This approach, for traffic classification, has other several advantages. For example, it does not consume high resources and does not make the network more complex, among others. This research is about implementing a multidimensional denial of service (DoS) detector using Sflow and an entropy approach for detecting different DoS attacks that have different characteristics, improving the traditional entropy approach into a more effective and accurate detection. The results are 100% detection tested with the most accurate dataset publicly available as a method for detecting DoS and DDoS attacks. Furthermore, by easily adding one line in the configuration file, it is possible to detect new attacks. Moreover, the false-negative ratio is zero because with Sflow it is possible to analyze more attack-dependent variables.**

***Index Terms*—Denial of service (DoS), Distributed Denial of service (DDoS), Sflow, Entropy, Mininet, SDN, Sflow-RT.**

I. Introduction

Nowadays, as networks are becoming more complex, the new data centers and cloud services require more scalability and flexibility.
On the one hand, traditional networks are mainly based on hardware, which is expensive, hard to manage and slow for changes.
On the other hand, software-defined networks (SDN) are a promising alternative in which the hardware it is not linked to the software. This let control all the network in a coherently and globally way using the software. The advantages over the other networks are several:

- It allows to change the network remotely: Currently, with the exponential growth of telework [1] the industry demands a remote way to apply changes.
- It allows for the execution of new settings and automated solutions.
- It allows for quickly scalable and flexible infrastructures reducing the volume of work of administrators and technical staff.
- It reduces the network complexity making the network less error-prone.

These networks suffer several new attacks [2] than traditional networks and the solutions are harder because of the characteristics of these software defined networks. The main vulnerabilities are in the following fields:

1) Denial of service attacks: Linked to the specific features that characterize SDN networks, these types of attacks can easily and cheaply be launched and the impact is critical.
2) Incomplete encryption: The low-level cryptography algorithms and the default unencrypted communications in SDN could represent a security weakness.
3) Information disclosure: Some applications don't have the proper security measures which may result in an attacker taking control of the network traffic information supplied.

This research focuses on Dos and DDoS attacks which are one of the most relevant security challenges in SDNs. Recent studies reveal that traditional detection approaches are no longer effective in SDNs. It is needed to find the best method in order to stop these attacks. This method should not make the network more complex and slower. Moreover, it should be effective in detecting the attacks. DoS and DDoS attacks are the result of multiple factors and could affect different resources. For example, some attacks could overflow the switch table, others could overload the memory, and others could overwhelm the available processor. Furthermore, some attacks are launched in the application layer (layer 7), others in the transport layer in different protocols (TCP, UDP), etc. As a result of this, the method should have mechanisms in order to detect and stop the attacks in a multidimensional way.
Finally, it is needed to develop an application in order to test the selected method. This is challenging because of the

multiple factors that need to be controlled in order to detect all the DoS and DDoS related elements. This application probably will have to use multiple processes, and shared memory, among others.
The remaining of this document is organized as follows:

- Section II. Preliminaries: An overview of the problem of denial of service attacks in SDN and the current solutions.
- Section III. Selection of the detection algorithm: The discussion about choosing the best detection algorithm solution.
- Section IV. Selection of the monitoring protocol: The discussion about selecting the best detection monitoring protocol.
- Section V. Entropy: The explanation of the concept and formulas of the selected detection algorithms, entropy.
- Section VI. SFlow. The description of the monitoring protocol Sflow.
- Section VII. Proposed solution: The discussion about the architecture and functionalities of the developed application using the selected algorithm, Entropy, over the chosen monitoring protocol, SFlow.
- Section VIII. Experimental results: The results of the proposed method using the developed application.
- Section IX. Conclusions and future work: The summary of the developed application and the investigated method with ideas about future work.

II. Preliminaries

A. *Software defined networking*

In order to understand the problems in SDNs networks, we have to understand how they work. OpenFlow [5] is the most widely used implementation architecture of SDN. Openflow enables the export of flow-based network information of all the elements of the network such as routers, and switches that can be effectively applied to network monitoring [8]. This research is focused on OpenFlow implementation. The architecture of SDN is not unique and has progressed with time, but all the models have the following common points:

- Control Plane: It operates network-wide policies and traffic flows. It consists of an SDN controller that manages and connects the application layer to the infrastructure layer.
- Infrastructure Plane: Contains the physical switches and routers. It is the hardware layer.
- Application and Data Plane: Includes a suite of applications and network features.

SDN has two APIs in order to communicate the different planes. Firstly, the Northbound API communicates the control layer and application layer. Secondly, the Southbound API for the communication of the control and infrastructure layers. It is possible to use the Northbound API to mitigate DoS and DDoS attacks, but it is not the best approach. The SDN layers are illustrated in Figure 1.

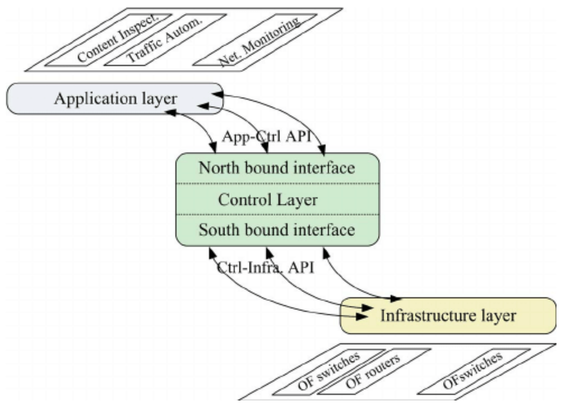

Fig. 1: The three planes/layers in the SDN architecture [12].

B. *Related solutions*

The following publicly available solutions have been highlighted in order to select the best method and improve it. To begin with, redundancy in the centralized network it is not the solution. For example, the use of multiple controllers is not the answer to DDoS attacks as it can lead to cascading failures of multiple controllers as demonstrated in [13]. The reason is that the load on a controller bearing the load of a failed controller can exceed its capacity, which can lead to worse situations such as cascading controller failures.

Secondly, we can split the solutions that analyze, classify and mitigate the attack traffic in SDN into three groups covered in the next subsections.

C. *Detection based on policies*

The detection based on policies is when certain network properties are analysed. For example, we can mark malicious traffic depending on the trust of the origin, malformed packet flags, time, etc. Several solutions have been published following this detection methodology, the main are:
R. Kandoi and M. Antikainen in [21], have made the first approach. They have estimated the impact of denial of service attacks using network parameters such as bandwidth, flow tables, latency, etc. These authors have investigated without proposing a solution.
The authors in [14] have proposed FlowRanger, a solution that prioritizes the traffic depending on the origin following three steps. Firstly, it calculates the confidence level for every packet, depending on the origins of the packet. Secondly, queues the packet with the corresponding priority. Finally, it processes the packet following a Round Robin

approach.
In [15], the authors propose a solution in which the traffic is categorized depending on the threat probability calculated by an intrusion detection system (IDS). Then it redirects the malicious traffic, and it is put on the queue.
Sflow-RT has official applications [16] in order to stop denial of service attacks. The first one is a fairly light anti-DoS that stops the flood attacks by the traffic rate and blocks the traffic using a BGP black hole. The second one is similar, but the traffic is blocked using BGP FlowSpec. The solution uses standard features of modern routing hardware to easily scale to large, high-traffic networks.

D. *Machine learning*

Machine learning has developed rapidly in recent years, but the research flow is going in the Deep learning way. Anyway, we can highlight the authors of [6] that use the Support Vector Machine (SVMs) applied to the DoS attack detection. Firstly, they extract the network features and then the SVM algorithm classifies the flow. They use the KDD99 dataset in order to make experiments. Finally, if the flow is considered as attack traffic a rule is sent to the controller in order to block the traffic. The conclusions are that it is important to optimize and enhance the detection method because it is limited to the studied dataset and environment and with a different environment it could change.

E. *Parallel Online Deep Learning (PODL)*

The traditional Deep learning (DL) algorithms are discarded due to the training phase not being possible because the system cannot be interrupted using DL, but online deep learning algorithms (ODL) can be used because they learn how to optimize the model over a stream of data instances in a sequential manner while the system is running. The problem is that ODL lets train the network with only one parameter, and the DoS attacks depend on different constraints. In [18], they use Parallel Online Deep Learning (PODL) in order to consider more constraints. Firstly, the available queue capacity of the controller. Secondly, the space available in the flow table of each switch. To sum up, PODL is a new promising alternative but it has been not tested in big networks, the use of PODL can consume excessive resources and nowadays is currently under study because it is not tested on big networks.

F. *Statistic analysis*

The statistical analysis of certain properties of network traffic is effective and optimum. In [19], they introduce the entropy with a detection rate of 96% for the threshold and a very low rate of attack traffic with early detection. They implement the system in the controller, and they said that it is very likely to implement a system to detect attacks in hosts, routers and all the network. In the research, they said that it is possible to use entropy to detect attacks in multiple environments.

G. *Monitoring algorithms*

It is needed to research methods that allow getting flow samples in SDN networks. OpenFlow is the used implementation, as explained before in Section II-A. It allows exporting the flow information of hosts, routers and other elements in the network. In order to improve the detection rate, it is needed to have the most complete view of the network. The following solutions are available:

- Sflow: It is the standard for packet sampling in order to monitor the network. Sflow is characterized by the low use of resources. The company InMon, currently, has several open-source applications focused on network traffic monitoring.
- Netflow: It is the Cisco proposed protocol for network monitoring. It is proprietary and not available in all the network elements. Furthermore, if some changes are applied to the network, the adaption of the Netflow is very expensive. Moreover, it consumes high resources. Nevertheless, Netflow provides complete visibility of the network and provides a lot of information.
- SNMP: It is the Simple Network Management Protocol. It provides the best solution for traffic monitoring in medium and low-traffic networks. The industry has not widely adopted this protocol due to several security problems [20] and the lack of support for adding new configurations because it is very simple.

III. Selection of the detection algorithm

Firstly, the redundancy method is excluded because it is not effective in stopping DDoS attacks. Furthermore, the detection based on policies has several problems:

- It is not flexible and adaptive: The traffic is analyzed based on the fixed parameters of the packets, and it is not possible to use it as a generic way to detect different attacks.
- It makes the network more complex: It is needed to add several new functions to the network, making it more complex.
- It increases the number of packets: In some cases, the scheme introduces more communications over the network, for example exchanging messages between the controller, the IDE and the main detection application.
- It does not block malicious traffic: The rate limit policies make the attack harder but still possible.

Secondly, it is possible to highlight the authors of [17] which analyze the different available proposed machine

learning approaches in order to get the limitations and enhance them. In this research, it is possible to find several conclusions:
On the one hand, the classic ML techniques in order to detect sophisticated attacks on big network environments have some drawbacks due to irrelevant features and the lack of labelled training data. These problems lead to difficulties to detect malicious traffic.
On the other hand, deep learning techniques are one of the most relevant solutions nowadays. DL techniques enhance ML techniques, and it is used by several companies, such as Google, Meta, and Microsoft, among others. The most relevant part is that DL extract automatically the relevant parameters from unlabeled data records, solving the lack of training in labelled data. Finally, SDN networks are very flexible and particularly susceptible to significant changes. Therefore, in the DL approach, it is not possible to train the model while the system is running. For that reason, this approach is not the most suitable for this goal.

Parallel Online Deep Learning (PODL) is the new promising alternative, but it has been not tested in big networks, the use of PODL can consume excessive resources and nowadays is currently under study. Table 1 compares the different solutions.

Problem	Redundancy	Policies	ML	DL	Entropy
Complexity	Low	Low	High	High	Low
Traffic	Low	High	Medium	Medium	Medium
Flexibility	Low	Low	Low	Medium	High
Detection	-	Very low	Medium	High	High

Table 1. Comparison of the different solutions for detection

To sum up, the detection based on policies, traditional machine learning and deep learning has several constraints as explained before. Furthermore, the online deep learning algorithms have an important disadvantage; the analysis can only depend on one constraint and the denial of service attacks usually depend on multiple factors. Parallel online deep learning is a solution to this issue, but it is currently under research, it was not tested on large networks and its performance is under debate. In consequence, the analysis method to focus on is the entropy approach. Using another method different from the analysis proposed and described before, it is possible to improve it in order to enable the analysis of other elements in the network, using a flow sample protocol, for example.

IV. Selection of the monitoring protocol

There are several packet flow sampling solutions. Table 2 resumes the comparison between the most used ones.

Feature	SNMP	NetFlow	Sflow
Packet capture	Yes	No	Yes, configurable
Protocol support	UDP	Layer 2, IP	Layer-independent
Configurable fields	Yes	Yes	Yes
Hardware acceleration	No	Yes	No
Byte count	Yes	Yes	Yes

Table 2. Comparison of the different monitoring protocol solutions

Sflow is layer independent and it is better in multi-protocol networks than Netflow. Furthermore, Netflow is proprietary and developed by Cisco, whereas Sflow is open source and available in OpenFlow, the most common Sflow standard. Moreover, Sflow provides a broader view of every flow because it is possible to configure the packet capture in order to get the relevant information. Finally, the authors of Sflow in [7], recommend the use of Sflow against another method in order to reduce the price in future network upgrades due to the majority of vendors supporting Sflow. In Netflow, adding monitoring flows later is very expensive. For the reasons explained before and the comparison of others available in Table 2, it is recommended to use Sflow. SNMP is highly used as a network monitoring protocol but due to its performance, it is recommended to use Sflow in a multiprotocol and multilayer environment.

V. Entropy

Entropy is a concept coming from thermodynamics and adopted by information technology. It is described as an uncertainty measure. For example, when a neighbour told us that the streets are wet, this information is not relevant because it is very common. Therefore, when the same neighbour told us that the streets are wet, but we know that it is not raining, he was giving us several relevant information.

A. The Shannon entropy

The information entropy was introduced by Shannon in 1948 [9]. The main objective of Shannon was to measure the uncertainty of a source of information. Given a set of data X and one finite symbols set $x_0 ... x_n$ which its respective probabilities $p_0 ... p_n$, it is possible to express the entropy of information as H(X):

$$H(X) = \sum_i p(x_i) \cdot \log_2 p(x_i) \qquad (1)$$

The base 2 logarithm is applied under the assumption that the information to be processed is represented by binary code. By changing the system encoding, the base of the logarithm must match that of the new representation.

When the value of entropy is 0 is associated with even distribution and when the entropy gets the value of

$\log_2 p(x_i)$ that is the maximum, the distribution is very random.

The information entropy, in order to detect DDoS attacks, allows to measure the randomness of packets reaching the network.

B. Normalized entropy

The problem with the Shannon Entropy is that for same-shaped distributions, the entropy is different for different sample sizes. Because it algorithm depends on the sample size. When we analyze a number of packets, every packet could have the same features such as the same IP source, for example, and the size of the sample could be different. In order to measure the entropy regardless of the sample size, a normalized entropy version of the Shannon Entropy should be used.

$$H(X) = -\frac{\sum_i p(x_i) \cdot \log_2 p(x_i)}{\log_2 n} \tag{2}$$

Where $n > 1$ is the number of elements. Note that now $H(x) \epsilon [0, 1]$ and if the data is small, the difference depends on the Riemann sum approximating the integrals. This is important because it must be considered in the selection of the packet size.

C. Weighted entropy

In order to enhance the entropy approach, it is possible to use the Weighted arithmetic mean of the entropy to reduce the noise and false positives using the historic values. The weighted average applied to the entropy does not require many resources and is easier to implement than other methods [10]. The weighted mean of a non-empty tuple of entropies $x_0 ... x_n$ with corresponding non-negative weights $w_0 ... w_n$ is:

$$W = \frac{\sum_i w_i \cdot x_i}{\sum_i w_i} \tag{3}$$

It is recommended to use the following weights in order to consider the previous entropy values: [0.05, 0.10, 0.20, 0.25, 0.4].

D. Standard deviation

Finally, the standard deviation is used to quantify the dispersion of the weighted entropy values. It is calculated using the following formula, where are $x_0 ... x_n$ the entropy values, N the number of elements, and W the mean value of these entropies:

$$SN = \sqrt{\frac{\sum_i (x_i - W)^2}{N}} \tag{4}$$

The standard deviation is used in order to measure and see the changes in entropy. If the deviation values of the entropy are similar the traffic is normal, but if the deviation is very different, such as, has high values or low values, the traffic is not usual.

E. Selection of the window

Shunsuke Oshima, Toshinori Sueyoshi and Takuo Nakashima have proposed in [11] a short-term statistics detection method based on entropy calculation. “Short term” refers to calculating entropy in small window sizes. A size of 50 packets was the lowest size that effectively detected attacks. The false-negative was only 5%, proving the effectiveness of the method.

VI. Sflow

Sflow is the new standard for network monitoring, developed by InMon, and it is supported in most network devices. For example, OpenFlow enables Sflow to export flow-based network information of all the elements of the network such as routers, and switches that can be effectively applied to network motorization.

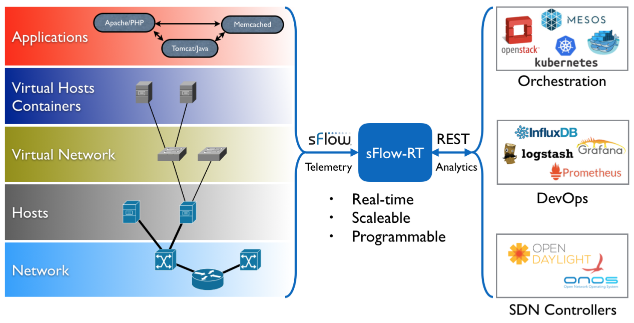

Fig. 2: Sflow-RT ecosystem [22].

Figure 2, shows the Sflow applications ecosystem. Firstly, it uses the Sflow telemetry to get all the network data information. This information must be defined before starting to send the data. This flow data, used for monitoring, can be collected using Sflow Agents.

A. Sflow Agents

The Sflow Agents can be integrated into the network equipment for more than 40 vendors.

- Network devices such as hubs, switches, routers, bridges, gateways, modems, repeaters.
- Hosts such as Windows devices, Linux, macOS, among others.
- Virtual hosts and containers such as Docker, Kubernetes, Hyper-V, KVM/libvirt, and Xen hypervisors.
- The web servers are NGINX, Apache, HAProxy, Tomcat, Memcached servers, and java virtual machines (JVM).

Normally, the Sflow protocol is already available in the network devices as a standard protocol, but it is not possible to monitor the whole network if we only have the network devices' telemetry. In the case of hosts, applications, virtual hosts and other devices another solution is required.

B. Host Sflow

Host Sflow is an open-source solution that has a minimal impact on the systems. It is possible to program and configure Host Sflow in order to get monitoring data from the devices to the monitoring tool. For example, on Windows, it is possible to measure performance such as free memory, the CPU utilization, among others. Another example is in the web server Apache, it is possible to use this agent in order to get all the number of petitions to a 404 page, the load of the web server, etc. All these settings are implemented by default in the Sflow Host application that it is compatible with the most operating systems and applications. However, if an application or operating system is not compatible, it is possible to easily add it as an open source code.

C. Sflow Collector

In this paper, as explained before, Sflow-RT is used as a collector. sFlow-RT is an open source software, provided by the company that developed Sflow, InMon. Using Sflow, this software provides real-time asynchronous network monitoring. In most cases, such as, in Sflow-RT the collector has two functions. The first one is to configure in all the agents the metrics and thresholds that they send. The second one is to get and parse all the monitored data. Sflow-RT collector is also equipped with an API REST to send the telemetry data to different monitoring applications such as InfluxDB, Grafana, Prometheus, and OpenDaylight, among others. One of the most important things is to configure the sampling rate depending on the link speed of each agent. The default configuration of Sflow-RT guarantees the flows that consume 10% of the bandwidth, defined as large flows, are noticed in less than one second.

D. Structure of a typical sFlow-RT application

The applications can be internal, using the JavaScript API, or external using the API REST. Normally, the applications have to implement the following steps:

1) Define flows: In order to define flows, it is needed to use keys. Keys are particular attributes that the agents have. For example, it is possible to match all the packets that have an IP source. Moreover, it is possible to use multiple keys to define one flow. As an advanced user, it is possible to define key functions. Key functions let filter, group, concatenation, and form conditions among others. For example, the following formula selects the IP destination and IP source based on the condition of the UDP packets received:

if:udpdirection:received:ipdestination:ipsource

The advantage is that if we filter the metrics using key functions, the flows will be filtered in the hosts and could carry out a relevant traffic reduction. Once the keys or functions keys are defined, it is needed to define the value of these keys that it is measured by the Sflow agent. Multiple values are admitted: frames, the number of petitions, requests, duration, etc.

2) Define thresholds: It is possible to apply the thresholds to a key, key functions and defined flows. Thresholds trigger depending on different values, e.g., frames, bytes, duration and requests, among others. There are two options in order to trigger the threshold. In the first one, the trigger is the largest flow that matches the metric. In the second one, each flow triggers the event.

3) Control events: On the one hand, it is possible to handle the flow events. Sflow-RT let's get the monitoring information of each flow. On the other hand, it is possible to handle the threshold events. These events are triggered in each agent, so it uses minimal traffic. The API REST call is asynchronous.

4) Control periodic tasks: It is possible to execute tasks that are programmed one-time or a periodic time. For example, it is possible to receive one flow one time or every 10 seconds.

VII. Proposed solution

The proposed solution is a Sflow-RT application written in Python that uses entropy as a method of classifying malicious and non-malicious traffic and the Sflow telemetry used for traffic monitoring protocol. Sflow enables the detection of multiple factors attacks.

A. Architecture Overview

Firstly, the external architecture with Sflow-RT is used as the analytics engine. Sflow-RT provides an open-source solution in order to allow the different network components to send the Sflow telemetry data. These components could be Apache, Nginx (Web servers), Kubernetes, Docker, Windows, Linux, Routers and a wide variety of elements. Sflow lets to configure the flows that should be sent in the network element and allows you to configure thresholds in each network component, this is easily configured using Sflow-RT. Once the Sflow metrics and thresholds are configured, the Sflow is collected and parsed by Sflow-RT and the proposed application receives the parsed flow. Finally, an API rest in the proposed application shows the detected traffic. This traffic could be blocked using methods such as BGP FlowSpec or by the controller. The highlighted method, BGP FlowSpec, is described in the future work section (IX). Figure 3 describes the external architecture.

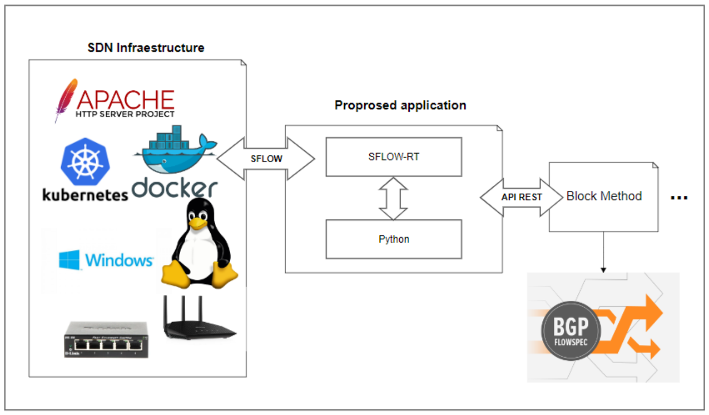

Fig. 3: The external architecture of the proposed application.

Secondly, Figure 4 illustrates the internal architecture, a detailed explanation of each function is shown below.

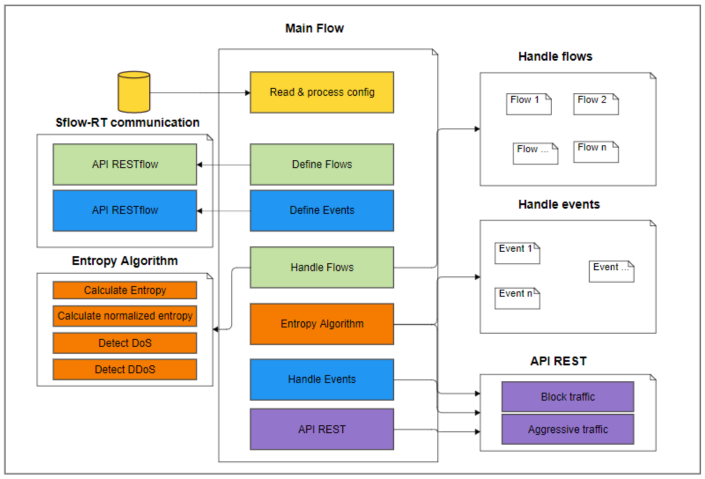

Fig. 4: The internal architecture of the proposed application.

Regarding the internal architecture, firstly, the application reads the configuration file that has the definition of the flow metrics to be sent by the network elements. Moreover, it has the definition of the thresholds (Events) that are triggered in each component. Then, the application contacts SFlow-RT using the API RESTflow in order to define the Flows and Events.
When a flow is triggered by one of the network components, the SFlow with the defined relevant information is sent to Sflow-RT and is received by the application. In the application, there is a thread per flow definition waiting for the data. Then, the entropy algorithm and the DoS and DDoS detection algorithm will be calculated and estimated per flow when the defined packet window is completed (in this case 50). The algorithm marks the traffic as aggressive if an attack is in process.
The events follow the same process as the flows, but are only sent when triggered. For example, when the memory of the host is full. When the traffic is marked as aggressive and the event is triggered, the traffic will be marked as attacker traffic to be blocked.
Finally, an API REST in the application present the attack and aggressive traffic.

B. Sflow telemetry

In the first place, the application set the flows and the thresholds defined on the configuration file. The flows are the network attributes defined by the keys, as explained in the Sflow section (VI), in which the entropy will be analyzed. The thresholds are events that trigger depending on a metric, as explained in the previous Section VII. Thresholds are used for detecting an overload or unusual attributes in hosts, applications, etc.

C. Entropy algorithm

In the second place, when a fixed defined window of 50 packets is reached, the entropy algorithm performs the following functions:

1) Calculate the entropy of each flow: The Shannon entropy value of every flow is calculated depending on their defined attributes, for example, the IP source address or the HTTP requests path and their defined values, for example, the number or packets.
2) Calculate the total entropy of the window: It is the sum of the entropies, and it is the entropy for the windows.
3) Calculate the normalized entropy: This is needed as we have a non-fixed window packet sample size, as explained in the entropy section (V). For example, if we set the window to 50, we can receive 10 packets with the same attributes (for example IP source) or fragmented packets that are correlated and the packets sample size will differ.
4) Calculate the weighted mean: The weighted mean of the window entropy is calculated as explained in Section V. By default, it is configured to use 5 historic samples and has the following weighs: 0.05, 0.10, 0.20, 0.25, 0.4.
5) Calculate the standard deviation: The standard deviation is calculated using the 5 historic samples and the currently weighted mean of the current window. It is used for the experimental graph to measure the analysis data as explained in Section V.

Finally, the entropy weighted mean of each flow passes through the Dos and DDoS detection algorithm.

D. DoS detection algorithm

As explained before, entropy has a minimum at zero and a maximum at one, so the weighted arithmetic means is in this range. When a previously configured threshold is reached, the following things could happen:
The first one is when the entropy trends to high values, for example between 0,6 and 1, the algorithm counts one for a DDoS attack. This is because if the entropy is high, the traffic is very dispersed, for example, the source IP, so the attack is distributed.
The second one is when the entropy trends to low values, for example between 0 and 0,5. The source traffic, for example using the IP address, is very uniform, and we are in a denial of service attack (DoS).

The proposed solution for the DoS attack is to block the incoming traffic from the source attribute such as IP, MAC, Path, port, etc. The proposed solution for the DDoS is to block the destination attribute, such as destination IP, Path, port, etc. This can block the traffic for legitimate users but, normally, the source of the distributed attack is from a compromised host that is spoofing network attributes it is not possible to block the source traffic, so in order to reduce the impact on the network the recommended action is to block the traffic in the destination port, IP and host, among others. As explained before the SDN networks have several security weaknesses, so it is common to have attacks on hosts, but if we want to keep the service running, or we are sure that the attack is coming from outside it is possible to configure the application to block only the source of the attack.

When the traffic is considered as DoS or DDoS, one is added to the pertinent counter. When a configurable threshold is reached, for example, 5 flows marked in 10 flow samples, the application will consider the flow traffic as aggressive traffic.

E. Events telemetry

The application sets the event metrics configured in the settings file. For example, it is possible to configure the application to trigger when the server load is high, the switch table is full, there are several 404 petitions, etc. This is very useful because the DoS and DDoS attacks have multiple indicators that can be used in order to reduce the false-positives. When an event metric is triggered, for example, when the server load is high, the aggressive traffic is marked as attack traffic.

F. Configuration and API REST

In the configuration file, it is possible to set the following things shown in Figure 5.

```
[THRESHOLDS]
ifinoctets = 80%

[FLOW_METRICS]
ipsource = 0

[GENERAL]
port = 8888
ip = 0.0.0.0
```

Fig. 5: An example of the configuration file.

To begin with, it is possible to define the thresholds and the flows. In the case of the flows, the definition is using the flow keys that are supported by sFlow-RT. However, depending on the network some keys won't be available. For example, if your network does not have a web server or HTTP configured, the requests or http_url keys won't be available. Furthermore, it is possible to use functions in order to use more keys, filter inside keys or select flows. The advantage of using functions is that every function is applied by the network element, so it could be reflected in a reduction of the Sflow packets transmitted. It is possible to mark the following functions illustrated in Table 3:

Function	Example
group	group:ipsource:ipdestination
if	if:tcpdirection:received:ipsource:ipdestination
concat	concat:_:ipsource:tcpsourceport
hash	hash:ipsource:ipdestination

Table 3. Key functions.

The defined flows can have the following values in the configuration file:

- 0: If it is used for defining one flow.
- 1: If it is used to define a flow and a threshold at the same time.
- 2:If it is used only to define a threshold.

When a threshold is defined it is possible to use the keys, key functions and defined flows. In the last case, it is possible to define a flow with the values 1 or 2 to be used by a threshold. For example, it is possible to create a flow with the 2 value, so only be used by thresholds. This flow will be *ipdestination, udpsourceport*, so it will match the flows by the IP destination and UDP source port. If we define a threshold using this definition we can put that when this flow (ipdestination, udpsourceport) reaches the 100 packets/s an event have to be triggered. This functionality, with the keys and key functions as explained before, let us detect attacks in all the traffic in multiple ways. Another example is when we define a threshold ifinutilization = 80%, an event will be triggered to mark the aggressive traffic to attack traffic when the switch ports with an ingress / egress utilization exceeding 80%.

Finally, it is possible to find the API REST listening configuration. The API REST has only one GET path that can be used in order to get the attack and block traffic.

G. *GitHub repository*

A publicly GitHub repository has been released with the source code and the instructions to execute and make the experimental environment. The URL is the following: https://github.com/igallar98/DDOS-SDN-Sflow-Entropy

VIII. Experimental Results

A. *Datasets*

Presently, it is very difficult to find an open Dataset publicity available. Most of the available datasets, such as KDD'99 dataset, are outdated and deprecated. Fortunately, InSDN [23] includes tested attack traffic making use of eight popular machine-learning-based techniques for IDSs. Figure 6 illustrates the thread classification accuracy of InSDN.

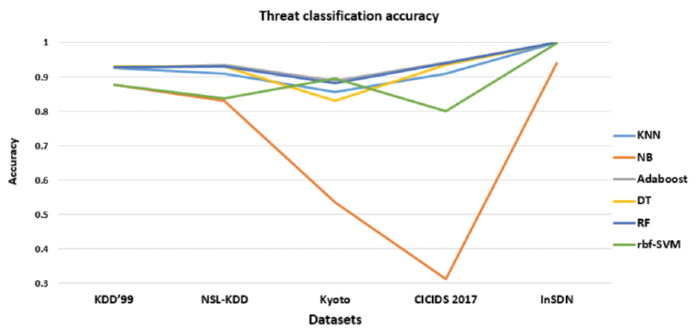

Fig. 6: InSDN dataset accuracy compared with other publicly available datasets [23].

B. *Experiments*

On the one hand, in order to test our entropy algorithm, it has been created a module that reads the dataset and passes the traffic through the entropy DoS detection algorithm and outputs the standard deviation and normalized entropy graphs. Dataset results are shown below.
On the other hand, once tested the algorithms using the datasets, a testing environment is created to test the application in a simulated environment. The environment is explained below.
In both tests has been selected a 50 packet window, as researched in the Entropy section (V) that is the recommendable minimum window size should be 50 in order to detect properly the attacks.

C. *Experimental settings and environment*

As explained in Section VI, OpenFlow is used as the SDN implementation. Through Mininet [24], an open-source SDN network emulator, it is possible to create a virtual SDN network that makes use of OpenFlow. The experimental Mininet topology is a single switch with 4 nodes connected. It is possible to perform attacks between one host to another host. For example, in order to perform an HTTP attack, it was configured a web server in one host and an attack tool has been used in the other host. The following open-source applications were used in order to perform the attacks:

- Hping3: It is a Linux terminal application to make and analyze packets. With Hping3 is very fast to perform flood attacks, such as SYN flood attacks. It is possible to emulate a distributed denial of service by using the option random source that spoofs the source IP addresses.
- Simple HTTP Flood: An open-source application written in python to manipulate and send HTTP requests, such as HTTP POST or GET, in order to perform the denial of service attack.
- slowhttptest: SlowHTTPTest is an easily configurable tool that replicates the Sflowloris application layer denial of service attacks.

All these applications are available in the Kali Linux distribution.

D. *Dataset results*

In order to detect malicious traffic, the selected parameter is the IP source in the case of layer 4 attacks and the attack path in the case of web application attacks. All of the available attacks have been tested, and the results are summarized in Table 4:

Attack	Type	Detected
TCP-ACK flood	DoS	Yes
TCP-SYN flood	DoS	Yes
UDP flood	DoS	Yes
HTTP flood	DoS	Yes
HTTP slow-rate	DoS	Yes
HTTP POST flood	DoS	Yes
HTTP Slowloris	DoS	Yes
TCP SYN flood	DDoS	Yes
UDP flood	DDoS	Yes
ICMP flood	DDoS	Yes

Table 4. Results of the tested attacks.

In order to compare the entropy of the normal traffic and the attack traffic, we get two graphs. The first one is the normalized entropy on the X-axis and the number of flow samples on the Y-axis. The second one is the standard deviation on the X-axis and the number of flow samples on the Y-axis.

In the first place, it is remarkable the InSDN normal traffic dataset 5 with the results shown in Figure 7.

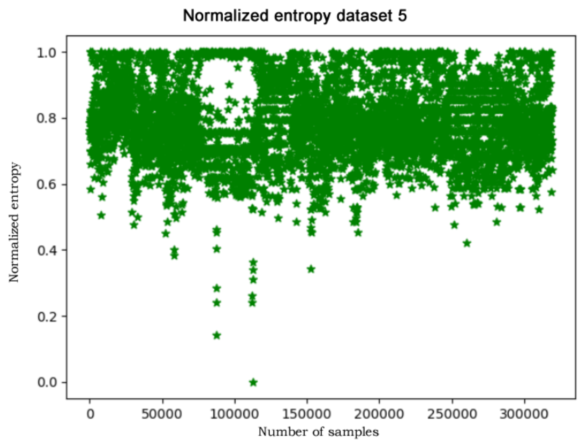

Fig. 7: Normalized entropy graph of the normal traffic dataset 5.

As we can see, the normalized entropy, in normal conditions, is very dispersed. It is possible to find some values that have abnormal entropy values that differ. These values can be reduced using the weighted arithmetic mean. In this case, the values are not significant, and the traffic is not marked, but if a non-attack dataset has several not normal entropy values, the traffic is marked as aggressive in order to check the entropy counters. Then, when the defined thresholds for the specified attack are triggered, the traffic is marked as an attack.
In the standard deviation it is possible to see that the values are dispersed, and the variation is not very high in a normal traffic situation. Figure 8 shows the standard deviation.

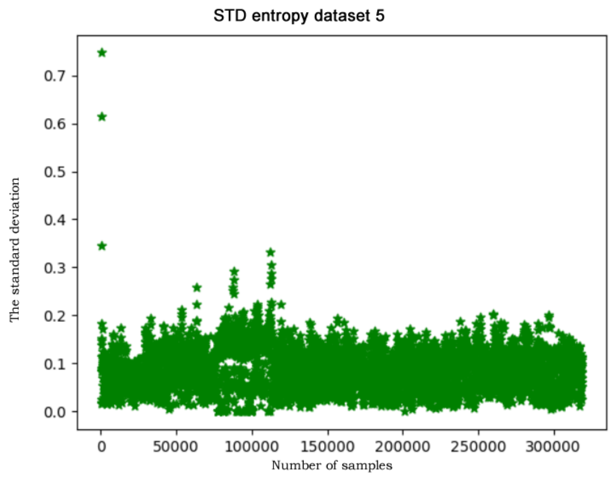

Fig. 8: The standard deviation of the entropy values graph of the normal traffic dataset 5.

When the attack dataset is tested, in the case of an SYN Flood, the entropy tends to zero because the traffic is not dispersed. In the case of a DDoS SYN Flood, the entropy trends to one because the source traffic is very dispersed, as it is shown in Figure 9.

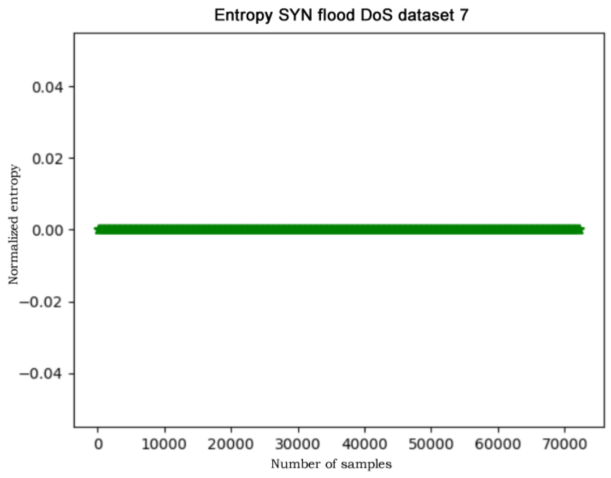

Fig. 9: Normalized entropy graph of the SYN DoS traffic dataset 7.

As we can see in Figure 10, the standard deviation of a DoS attack is stable, such as in the case of a DDoS attack because the entropy is stable.

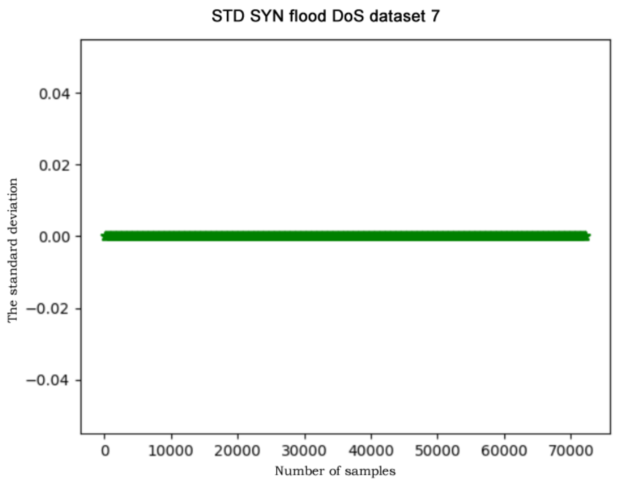

Fig. 10: The standard deviation of the entropy values graph of the SYN DoS traffic dataset 7.

Finally, Figure 11 displays the entropy values when the traffic is normal and an attack is triggered. In this case, the attack is a DoS SYN Flood attack and the entropy tends to zero because of the less dispersion of the traffic during a DoS attack.

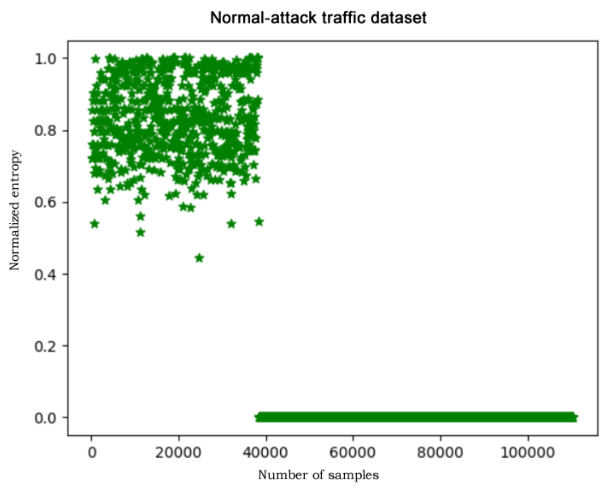

Fig. 11: The standard deviation of the entropy values graph of the SYN DoS traffic dataset 8.

E. Experimental results

In order to reduce the false positives, the application takes advantage of the Sflow protocol for triggering metric defined events. This is explained in Section VII. The following attacks have been tested using the test environment:

- Flood layer 4 attacks include TCP SYN flood, UDP flood, and ICMP flood.
- Flood layer 7 attacks in the application layer, such as HTTP flood.
- HTTP slowloris attack. The slowloris attack opens connections in the web server in order to maintain several open connections until the server gets down. This attack, normally, is very difficult to be detected because of the low traffic that it uses.

The detection rate for these attacks was 100% and there were not false-negatives due to the metric events module used. Moreover, in the dataset experiments the 100% of available attacks were detected with no false positives.

IX. Conclusions and Future Work

This research has provided a detailed study of one of the most important security weaknesses in software-defined networks, the denial of service attacks. Firstly, the current packet sampling and monitoring protocols for SDN solutions have been researched, concluding with Sflow and Entropy respectively. Secondly, it has been developed an application using these solutions. It has been performed benchmarking experiments on the InSDN dataset, which is the most accurate dataset available for SDN, and explained the advantages of using SFlow with an entropy approach. Results show that this method has a good detection rate, 100% in the case of the available attacks in the used dataset, with no false positives. Therefore, by adding only a couple of lines in the configuration file, it is possible to detect new DoS attacks. In the future, we plan to implement a mitigation module based on BGP FlowSpec. BGP Flowspec [25] allows sending of a signal in order to block the DoS traffic in different ways, redirecting the traffic to a particular point to analyze it or policy it at a defined rate or block the flow specification, among others. Moreover, we will propose to study the DoS attack tuple space explosion (TSE). The performance of the switch can be reduced up to 12% of its total capacity with a very low traffic rate during a TSE attack. This attack, in general, does not generate any specific attack traffic pattern, but rather some attack packets with randomly chosen IP headers and arbitrary message contents. This implies that the entropy of the network increases, and theoretically it is possible to detect it using our proposed method.

References

[1] H. Koyama, Y. Nakagawa, S. Tanimoto, T. Endo, T. Hatashima and A. Kanai, ”Risk Assessment of Telework for the New Normal Era,” 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE), 2021, pp. 496-497, doi: 10.1109/GCCE53005.2021.9621909.

[2] Pradhan, Aayush & Mathew, Rejo. (2020). Solutions to Vulnerabilities and Threats in Software Defined Networking (SDN). Procedia Computer Science. 171. 2581-2589. 10.1016/j.procs.2020.04.280.

[3] I. Ahmad, S. Namal, M. Ylianttila and A. Gurtov, ”Security in Software Defined Networks: A Survey,” in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2317-2346, Fourthquarter 2015, doi: 10.1109/COMST.2015.2474118.

[4] W. Sun, Y. Li and S. Guan, ”An Improved Method of DDoS Attack Detection for Controller of SDN,” 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET), 2019, pp. 249-253, doi: 10.1109/CCET48361.2019.8989356.

[5] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. 2008. OpenFlow: enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38, 2 (April 2008), 69–74. https://doi.org/10.1145/1355734.1355746

[6] L. Yang and H. Zhao, ”DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method,” 2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN), 2018, pp. 174-178, doi: 10.1109/I-SPAN.2018.00036.

[7] sFlow and Netflow. (n.d.). Retrieved September 10, 2022, from https://blog.sflow.com/2009/05/sflow-and-netflow.html

[8] Giotis, Kostas & Argyropoulos, Christos & Androulidakis, Georgios & Kalogeras, Dimitris & Maglaris, Vasilis. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Computer Networks. 62. 122–136. 10.1016/j.bjp.2013.10.014.

[9] Y. J. Cheng et al., ”Shannon entropy: A specular echo-insensitive imaging metric showing myocardial anisotropy,” 2009 IEEE International Ultrasonics Symposium, 2009, pp. 373-376, doi: 10.1109/ULTSYM.2009.5441837.

[10] Kumar, Amit & Peeta, Srinivas. (2015). Entropy Weighted Average Method for the Determination of a Single Representative Path Flow Solution for the Static User Equilibrium Traffic Assignment Problem. Transportation Research Part B Methodological. 71. 213-229. 10.1016/j.trb.2014.11.002.

[11] S. Oshima, T. Nakashima and T. Sueyoshi, ”Early DoS/DDoS Detection Method using Short-term Statistics,” 2010 International Conference on Complex, Intelligent and Software Intensive Systems, 2010, pp. 168-173, doi: 10.1109/CISIS.2010.53.

[12] Security in Software Defined Networks: A Survey - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-three-planes-layers-in-the-SDN-architecture_fig1_281578191 [accessed 01 Aug, 2022]

[13] Yao, G., Bi, J., & Guo, L. (2013). On the cascading failures of multi-controllers in Software Defined Networks. 2013 21st IEEE International Conference on Network Protocols (ICNP), 1-2.

[14] L. Wei and C. Fung, ”FlowRanger: A request prioritizing algorithm for controller DoS attacks in Software Defined Networks,” 2015 IEEE International Conference on Communications (ICC), 2015, pp. 5254-5259, doi: 10.1109/ICC.2015.7249158.

[15] L. Dridi and M. F. Zhani, ”SDN-Guard: DoS Attacks Mitigation in SDN Networks,” 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), 2016, pp. 212-217, doi: 10.1109/Cloud-Net.2016.9.

[16] Github Iván Gallardo (2022). DDOS-SDN-Sflow-Entropy - The Proposed application. Retrieved September 17, 2022, from https://github.com/sflow-rt/ddos-protect

[17] M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, ”Machine-Learning Techniques for Detecting Attacks in SDN,” 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), 2019, pp. 277-281, doi: 10.1109/ICC-SNT47585.2019.8962519.

[18] El Kamel, Ali & Eltaief, Hamdi & Youssef, Habib. (2021). On-the-fly (D)DOS attack mitigation in SDN using Deep Neural Network-based rate limiting. Computer Communications. 182. 10.1016/j.comcom.2021.11.003.

[19] S. M. Mousavi and M. St-Hilaire, ”Early detection of DDoS attacks against SDN controllers,” 2015 International Conference on Computing, Networking and Communications (ICNC), 2015, pp. 77-81, doi: 10.1109/ICCNC.2015.7069319.

[20] Haiyan Liu and Yifei Huo, ”Analysis on the security of MIB objects defined in network device,” 2016 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 1067-1070, doi: 10.1109/CompComm.2016.7924868.

[21] R. Kandoi and M. Antikainen, ”Denial-of-service attacks in OpenFlow SDN networks,” 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1322-1326, doi: 10.1109/INM.2015.7140489.

[22] InMon. (2013). Sflow-RT ecosystem. Sflow-RT Ecosystem. Retrieved September 4, 2022, from https://sflow-rt.com/img/rt-ecosystem.png

[23] M. S. Elsayed, N. -A. Le-Khac and A. D. Jurcut, ”InSDN: A Novel SDN Intrusion Dataset,” in IEEE Access, vol. 8, pp. 165263-165284, 2020, doi: 10.1109/ACCESS.2020.3022633.

[24] R. Barrett, A. Facey, W. Nxumalo, J. Rogers, P. Vatcher and M. St-Hilaire, ”Dynamic Traffic Diversion in SDN: testbed vs Mininet,” 2017 International Conference on Computing, Networking and Communications (ICNC), 2017, pp. 167-171, doi: 10.1109/IC-CNC.2017.7876121.

[25] F. Paolucci, A. Giorgetti, F. Cugini and P. Castoldi, ”Service chaining in multi-layer networks using segment routing and extended BGP FlowSpec,” 2017 Optical Fiber Communications Conference and Exhibition (OFC), 2017, pp. 1-3.

[26] Csikor, Levente & Divakaran, Dinil Mon & Kang, Min & Korosi, Attila & Sonkoly, Balázs & Haja, David & Pezaros, Dimitrios & Schmid, Stefan & Rétvári, Gábor. (2019). Tuple Space Explosion: A Denial-of-Service Attack Against a Software Packet Classifier. 10.1145/3359989.3365431.