Comparative Analysis between the k-means and Fuzzy c-means Algorithms to Detect UDP Flood DDoS Attack on a SDN/NFV Environment

João Ribeiro de Almeida Neto[a], Layse Santos Souza[b] and Admilson de Ribamar Lima Ribeiro[c]
Department of Computing, Federal University of Sergipe (UFS), São Cristóvão, Brazil

Keywords: Attack Detection, DDoS Attack, UDP Flooding, SDN, NFV, Microservices, Machine Learning, Fuzzy c-means, k-means.

Abstract: Distributed Denial of Service (DDoS) attacks are a growing issue for computer networks security and have become a serious network security problem. Environments based on Software Defined Networking (SDN) and Network Function Virtualization (NFV) offers the ability to program a network and allows dynamic creation of flow policies. Allied to that, clustering algorithms can be used to classify and detect DDoS. This paper presents a study and an analysis of two unsupervised machine learning algorithms used to detect DDoS attacks in an SDN/NFV simulated environment. The results obtained by the two algorithms include an accuracy rate of 99% and the k-means algorithm was 33% faster than fuzzy c-means, which demonstrates its effectiveness and scalability.

1 INTRODUCTION

Distributed Denial-of-Service (DDoS) is a significant issue in Network Security for data centers (Patgiri et al., 2018). Those attacks are capable of causing massive disruption in any information communication technology (ICT) infrastructure (Bawany et al., 2017). According to recent analysis, DDoS attacks doubled in frequency in a period of just six months (Newman, 2019).

DDoS is a specialized form of attack that takes place into a widely distributed environment, through a coordinated anonymous identification of the attackers and their infected malware. The first step is to create the bot army, either through malware injection or by infiltration.

Once the army is infiltrated, an anonymous attacker sends commands to a controller, which passes on the information to the army. Such action eventually results in the army sending request packets to the target system, thereby stealing its resources and pulling them away from the active zone.

The features of software based traffic analysis and centralized control over the network of Software Defined Networking (SDN), allows cloud administrators detecting and reacting to the DDoS attacks more effectively (Bhushan and Gupta, 2019).

NFV has been proposed to innovate in service delivery by virtualizing functions previously performed by specific hardware appliances. Virtualized network functions, which compose the service chain, are the basic elements to achieve the complete virtualization of service delivery. Dynamic composition of services make such task even harder than in legacy networks but easier with SDN.

SDN and NFV (Network Function Virtualization) concepts are considered complementary, sharing the goal of accelerating innovation inside the network by allowing programmability, and altogether changing the network operational model through automation and a real shift to software based platforms (Kreutz et al., 2014).

Signature-based matching is a widely used technique that can be used for intrusion detection and has high accuracy (Vigna et al., 2004). However, this technique does not adapt to new scenarios in a satisfactory way, since there is a need for frequent updating of signatures. In addition, developing and maintaining an updated signature database can be costly and impractical. For this reason, the use of anomaly-based technique may be more interesting in dynamic environments. It uses pattern recognition to clas-

[a] https://orcid.org/0000-0002-7090-4756
[b] https://orcid.org/0000-0002-9908-0949
[c] https://orcid.org/0000-0003-2010-6024

Neto, J., Souza, L. and Ribeiro, A.
Comparative Analysis between the k-means and Fuzzy c-means Algorithms to Detect UDP Flood DDoS Attack on a SDN/NFV Environment.
DOI: 10.5220/0010176201050112
In *Proceedings of the 16th International Conference on Web Information Systems and Technologies (WEBIST 2020)*, pages 105-112
ISBN: 978-989-758-478-7

sify data based on statistical information (Yuan et al., 2010). In addition, it is possible to use machine learning to develop applications that detect network anomalies, especially unsupervised machine learning techniques. They can be a viable alternative, since the system itself separates network traffic into different categories without the need for a supervisor (Suresh and Anitha, 2011).

In this context, this paper shows a technique to detect DDoS attacks on a SDN/NFV environment with the implementation of two different unsupervised machine learning algorithms to improve network security. This is a preliminary study and the goal of such an experiment is to effectively detect DDoS attacks and compare the accuracy and efficiency of the described algorithms.

This paper is divided and organized as follows. Section II presents related work. Section III displays the proposed architecture. Section IV describes the experiments. Section V exposes the results. Finally, Section VI brings out conclusions and future work.

2 RELATED WORKS

Duy et al. present a statistical mechanism based on detecting DDoS attacks in the SDN environment using the entropy metric, considering the profile differences of the host function to suspect the sub-attack state and, as well, the time factor information gathering activities (Duy et al., 2018) .

Mutu et al. demonstrate how SDN responsiveness to UDP flood attacks can be measured, by creating switches at the switch level using the percentage of traffic and an amplification rate (Mutu et al., 2015).

Preamthaisong et al. lecture the enhanced DDoS detection using genetic algorithms combined with SDN decision trees. The types of attacks generated were TCP SYN flood, UDP flood, ICMP flood and TCPKill. They also evaluated the performance of the algorithms to analyze network traffic (Preamthaisong et al., 2019).

Nugraha et al. demonstrate a new network based approach to impact analysis for SDNs. To do that, they adopted qualitative and quantitative approaches, took three types of DDoS attacks (ICMP, UDP and TCP Syn Flood) into consideration, calculated the impact of those in the entire network and identified the most impacted component (Nugraha et al., 2019).

Wei et al. exposed an UDP flood DDoS attack, demonstrated the proposal to successfully subvert the attack to a SDN functionality and exhibited a defense mechanism that can effectively reduce the damage caused by UDP flood attack (Wei et al., 2016).

Although Duy et al. paper had great contributions, it did not use SDN to manage network resources observing the depletion of UDP bandwidth. Besides that, NFV was not used to develop network functions separating them in microservices and machine learning to analyze the performance and accuracy of the two algorithms used to detect DDoS attacks (Duy et al., 2018).

Regardless Mutu et al. proposed a responsive mechanism to UDP attacks in SDN using actuating triggers at the switch level, they did not work with machine learning and the NFV environment (Mutu et al., 2015).

While the study proposed by Nugraha et al. focus on analyze the impact of DDos attacks, we use unsupervised machine learning algorithms to detect UDP Flood DDoS attack in a SDN/NFV environment with a microservices architecture. Also, analyze the performance and accuracy of these algorithms (Nugraha et al., 2019).

Preamthaisong et al. and Wei et al. papers had great contributions to detecting DDoS attacks in SDN networks, but none of them worked with the NVF environment or had a microservice architecture (Preamthaisong et al., 2019) (Wei et al., 2016).

Considering those points, this work has a goal to study and analyze the accuracy and efficiency of unsupervised machine learning algorithms to detect DDoS attacks in a SDN network and work with NFV and microservices architecture.

3 PROPOSED ARCHITECTURE

Despite of their widespread adoption, current networks are complex and hard to manage, due to control and data plans being bundled inside networking devices to, vertically integrated, reducing flexibility and hindering innovation (Esch, 2014).

SDN is an emerging networking paradigm that aims to change the limitations of current networks (Esch, 2014). Its architecture is based on decoupling data and a control plan, routing decisions are flow-based, control logic is moved to an external entity and the network is programmable through software applications (eg, Firewall, IDS, Routers) (Kreutz et al., 2014).

SDN architecture can be exploited to improve network security by providing a system for monitoring and analyzing traffic or detecting anomalies. Applications can run on the controller and a new or updated security policy can be propagated across the network in the form of flow rules (Scott-Hayward et al., 2013).

SDN offers a favorable environment for the implementation of network functions virtualization (ETSI, 2017). NFV allows the development of network functions via software and they are executed on general purpose hardware, previously they are implemented on specific hardware. It is possible to develop and test services on the same infrastructure, reducing development costs and time to market availability (Bonfim et al., 2019).

Microservices are small and autonomous services that work together making application development, integration and maintenance much easier (Newman, 2015). Such architecture allows the separation of monolithic systems into smaller components that can be deployed, updated and disposed of independently. These small, independent services, work together and can be integrated with SDN and NFV to break down network functions from underlying hardware functions (Newman, 2015) (Dragoni et al., 2017).

Based on that, we propose a NFV/SDN architecture to allow the deployment of network services on demand, showed in Figure 1. It comprises three layers: Application, Control and Infrastructure. Application Layer consists of network services. Control Layer has a global view of computer and network resources. Infrastructure Layer is composed of various networking equipment which forward network traffic.

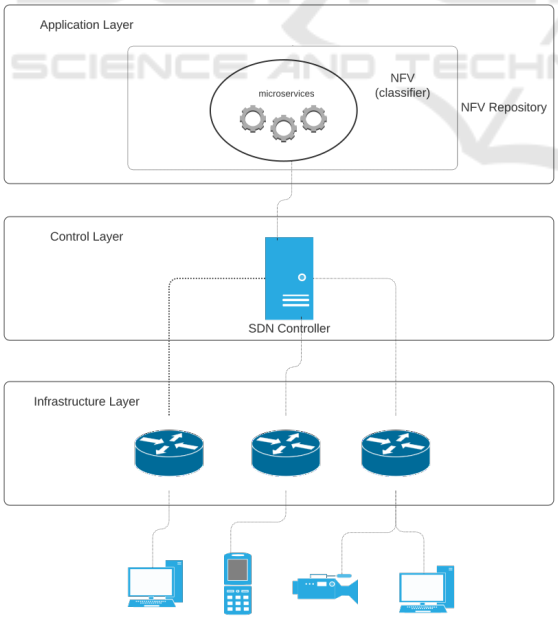

Figure 1: Proposed Architecture.

In general, a DDoS attack aims to prevent legitimate users from accessing a target system or service by overloading the system's resources (Mahjabin et al., 2017). There are two types of DDoS attack detection (L. Dali, 2015):

- Signature-based detection: this approach is generally used based on rules or filters to detect known and stored attacks within the database, otherwise limitation is a new undetected attack;
- Anomaly-based detection: this approach works towards behavior, in other words, to detect a new attack it is necessary to train the typical behavior of the network.

Microservices integration with SDN and NFV attests to an improvement in security, in particular, the detection of anomaly-based DDoS attacks. Anomaly-based detection uses the baseline concept of network behavior and machine learning can be used to predict behavior that leaves that baseline (Sahoo et al., 2018).

Basically, classification systems are either supervised or unsupervised, depending on whether they assign new inputs to one within a finite number of discrete supervised classes or unsupervised categories, respectively (Xu and Wunsch, 2005).

A supervised machine learning algorithm needs a previous classification of data for training the algorithm. The unsupervised approach does not need previous classification and data with similar characteristics are grouped.

In unsupervised classification, called clustering or exploratory data analysis, no labeled data are available. The goal of clustering is to separate a finite unlabeled dataset into a finite and discrete set of "natural," hidden data structures, rather than provide an accurate characterization of unobserved samples generated from the same probability distribution (Xu and Wunsch, 2005).

4 EXPERIMENTS

For the conduction of the proposed experiments, a physical host, running Windows 10 operating system, equipped with an Intel Core i7-4500U processor, 8GB of RAM and a 240GB SSD was used. Virtual machines were created with VirtualBox hypervisor in version 6.0. VM01, a virtual machine running Ubuntu 16.04 operating system, configured with 3GB of RAM, 1 CPU and 20GB of HD was used in all three experiments.

The SDN controller chosen was POX, is an open source controller for developing SDN applications. POX Controller provides an efficient way to implement the OpenFlow protocol which is the communication protocol between the controllers and the switches (Kaur et al., 2014). The OpenFlow protocol

was proposed to standardize the communication between the switches and the software-based controller in an SDN architecture (McKeown et al., 2008). The Flow Table characterizes the flows received and six information in it were chosen as features to elaborated dataset, packet_count, byte_count, duration_sec, duration_nsec, tp_src and tp_dst. The description of each of them can be seen in Table 1.

Table 1: Chosen Features.

Feature	Description
packet_count	Number of packets in a flow
byte_count	Number of bytes in a stream
duration_sec	Time the flow was active in seconds
duration_nsec	Time the flow was active in nanoseconds
tp_src	TCP Source Port
tp_dst	TCP Destination Port

The experiments were simulated on Mininet, a standard network emulation tool for SDN environment. It allows the creation of a virtual hosts network, a simple system with switches, controller and links. We used a topology in Mininet with one switch and 10 hosts and created a synthetic dataset for the experiments. For that, we developed two scripts, one to generate random normal TCP, UDP, and ICMP traffic and another to generate the UDP Flood attack. We used the Scapy to create the scripts, a powerful tool for generating, scanning, sniffing, attacking, and forging packages, allowing programmers to create and send customized packages to meet their requirements. Besides that, we developed a component for the POX Controller that stores the statistics of Flow Table for a current data stream in a .csv file. Then, both scripts were executed separately and the component created two files including normal and attack traffic samples. The samples were marked correctly with the normal and anomalous labels and merged. The dataset contains 8,820 samples, 6,282 of normal traffic, and 2,538 of malicious traffic.

We used Docker Container for experiments. It is a container virtualization platform that facilitates the creation and administration of isolated environments. Docker makes it possible to package an application or even an environment inside a container, thus making it portable for any other host that has Docker installed. It has advantages such as speed, boot time, resources saving, the understanding of how container processes are carried out within the host and the possibility of uploading several containers simultaneously while consuming fewer resources of virtual or physical hardware (Mouat, 2015).

Besides Docker, Containernet (Peuster et al., 2016) was also used. It enables the use of the NFV concept and allows adding and removing containers from an emulated network at runtime, which is not possible in Mininet (Peuster et al., 2016). There is the possibility of on demand installation, without the need to install new equipment, develop and test services on the same infrastructure, cutting down development costs and time to market availability (Bonfim et al., 2019).

We studied and analyzed two machine learning unsupervised clustering algorithms, fuzzy c-means and k-means, working in anomaly-based detection of UDP Flood DDoS attack.

Fuzzy c-means have been used very successfully in many applications, such as pattern recognition (Ming-Chuan and Don-Lin, 2001). It is used to analyze several input data points based on distance, thus, its structure allows data to belong to two or more clusters, which are formed according to the distance between data points and the cluster centers (Velmurugan and Santhanam, 2010).

K-means clustering is the well-known technique of unsupervised learning (Kumari et al., 2016). It is used to classify a given set of data through a certain number of clusters. A cluster refers to a collection of data points aggregated due to certain similarities (Velmurugan and Santhanam, 2010).

Three experiments were carried out: making the model and do tests using test dataset, detection in an SDN environment and detection in an SDN/NFV environment oriented to microservices. In the last two experiments, network monitoring and management was made based on the assumption that it would be compromised, that is, the botnet would have already been formed and hosts would already have been infected.

4.1 Experiment A

The purpose of this experiment is to create a model of both algorithms and test it. To support them, the scikit-learn library was used, a simple and efficient tool for predictive data analysis. K-means algorithm is natively present on it but, in order to use fuzzy c-means, it was necessary to install a pip package providing it (Dias, 2019). The dataset cited in section 4 was divided in two, one half was used for training and the other half for testing. The mass of data used for training had 4,410 samples divided in 3,141 samples with normal traffic and 1,269 samples with malicious traffic. To prove the effectiveness of the algorithms, the other half of the original dataset was used, which was called as test dataset. Data was pre-

processed using the sklearn.preprocessing package, because, in general, machine learning algorithms benefit from data standardization. MinMax scaler was the pre-processing method used in this experiment. It is a method that transforms resources by scaling each one to a specific interval. In addition, it scales and translates each resource individually, so that belongs to the specified interval in the training set, for example, between zero and one.

4.2 Experiment B

In this experiment we only focus on the SDN environment. We created a star topology with one switch and four hosts (h1, h2, h3 and h4) in Mininet. The host h4 was the victim and we used a script to generate random traffic (ICMP, TCP and UDP) and another to generate the UDP flood attack traffic. Besides that, we created a component for the POX Controller for performing flow classification, as illustrated in Figure 2. Through the statistics monitor of the component, the flow features are extracted and then the flow classifier will arrange the flow as normal or anomalous. If the flow is normal, it is installed and will run normally, if not, an anomalous flow alert is triggered.

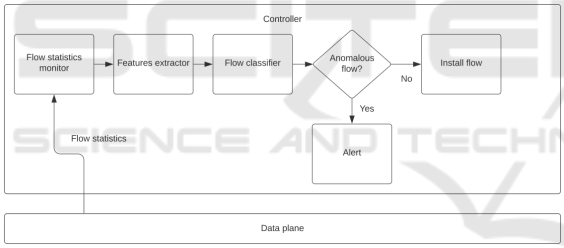

Figure 2: POX Component Proposed.

4.3 Experiment C

In this experiment the SDN/NFV environment was used, implementing microservices. Figure 3 presents a component diagram which is a static view of how the system was implemented and which components are used in proposed architecture to detect UDP flood DDoS attacks. The environment consists of a VM host, a POX Controller that runs directly on the VM host and a containerized environment, where virtual network services will be executed. For that, it was necessary to install the Containernet tool, which includes the Docker Container and Mininet tools. Within the Docker Container, the following virtual network microservices were created: flow classifier, a network host with normal traffic generator and a network host with anomalous traffic generator. For this experiment, we created four network hosts. Host h1, which generated normal network traffic between all hosts. Host h2 that was the victim and received the DDoS attacks and hosts h3 and h4, which generated traffic attack for h2. In addition, there was also a flow classifier.

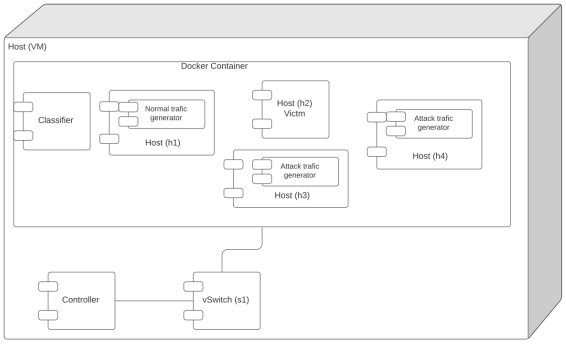

Figure 3: Component Diagram.

5 RESULTS

The metrics used to analyze fuzzy c-means and k-means algorithms in the experiments were: precision, recall, f1-score and support, which are present in the package sklearn.metrics.precision recall fscore_support. Each of the metrics is described in Table 2.

Table 2: Metrics chosen.

Metric	Description
*P*recision	tp / (tp + fp), tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative
*R*ecall	tp / (tp + fn), tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples
*F*1-score	Harmonic average of the precision and recall
*S*upport	Number of occurrences of each class in y_true

In addition, the following metrics were used: accuracy and execution time of the algorithm. Accuracy is a statistical value that determines the algorithm's proximity to its ideal. If its value is 1, it means that the algorithm has no error and classifies data perfectly. Below is the accuracy formula (1). To calcu-

late it, the accuracy_score method, also present in the sklearn.metrics package, was used. To measure execution time, the time module in Python was used.

$$Accuracy = \frac{CorrectClassifications}{TotalSamples} \quad (1)$$

5.1 Experiment A

For Experiment A, pre-processing to the test dataset was performed using the MinMaxScaler scaler. After using the created model, fuzzy c-means and k-means algorithms obtained precision, recall, f1-score and support values, as shown in Table 3.

Table 3: Metrics for fuzzy c-means.

	Precision	Recall	F1-score	Support
0	1.00	1.00	1.00	3141
1	1.00	1.00	1.00	1269
Avg/ Total	1.00	1.00	1.00	4410

As can be seen in Figure 4, both algorithms had a 99.98% accuracy rate, that is, it revealed that both of them have a satisfactory accuracy in detecting such type of attack.

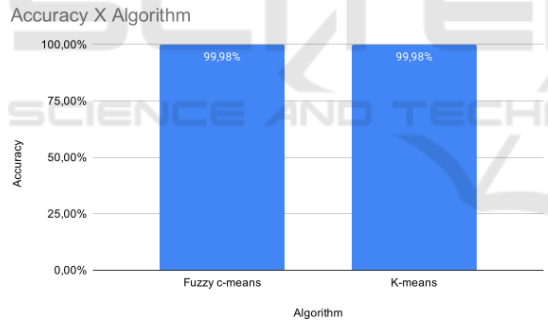

Figure 4: Accuracy in experiment A.

In this scenario, fuzzy c-means algorithm had 0.61 seconds of execution time. K-means algorithm, took 0.44 seconds to be completed, as it can be seen in Figure 5. K-means algorithm proved to be 0.17 seconds, or 27.86% faster than fuzzy c-means.

5.2 Experiment B

All metrics, except execution time, were the same in Experiments A and B. For Experiment B, fuzzy c-means algorithm execution time was 1.25 seconds. As for the k-means algorithm, it was 1.09 seconds, as can be seen in Figure 6. K-means algorithm proved to be 0.16 seconds faster, or 12.8% faster. The execution time is the simples arithmetic mean of the time of

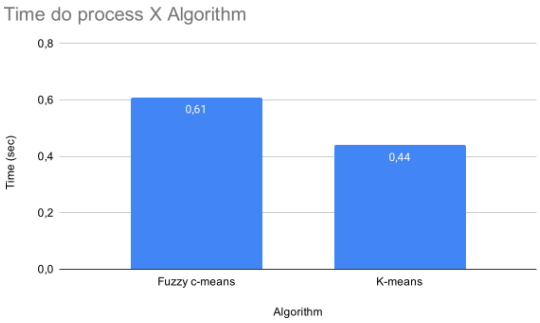

Figure 5: Time to process in experiment A.

1,000 predictions. This scenario might have increased due to the execution of the classification method had competed for processing resources on the machine with the POX Controller process, since the classification took place in the controller.

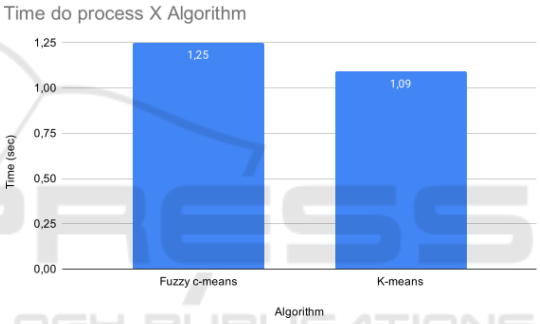

Figure 6: Time to process in experiment B.

5.3 Experiment C

All metrics, except execution time, were the same in Experiments A, B and C. For Exercise C, the execution time for fuzzy c-means algorithm was 0.03 seconds. As for k-means algorithm, it was 0.02 seconds, as can be seen in Figure 7. K-means algorithm turned out to be 0.01 seconds faster, which makes its execution 33.33% faster. The execution time is the simples arithmetic mean of the time of 1,000 predictions. This scenario may have been greatly reduced by the fact that the execution of the classification method was performed in a specific container.

6 CONCLUSION

During the experiment we encountered some challenges and among them we noticed the impossibility of classifying all flows sequentially in the environment of Experiment C, which used the concept of

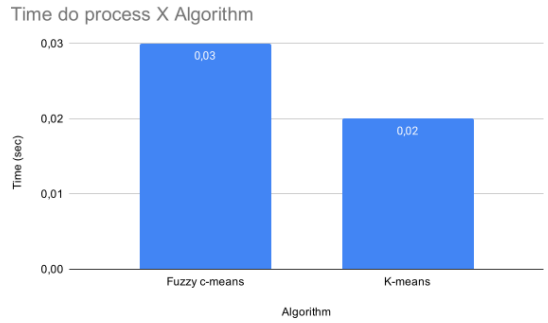

Figure 7: Time to process in experiment C.

NFV and microservices. That was due to the fact that the call to the classifier occurs through POST/HTTP. For that, we used a Python library called requests, to make the request, while on the server side we used FLask, which is a web framework written in Python and based on the WSGI library. When making calls to the classifier endpoint in an uninterrupted sequential manner with each flow that reaches the controller, the API requests collapses and stops working. That is, it is as if the DDoS attack was also affecting requests library. Thus, in order to carry out the experiment, it was necessary to classify only a few samples of flows.

The experiments have proved to be effective in detecting UDP Flood DDoS attacks, using booth fuzzy c-means and k-means algorithms. We obtained satisfactory results for all the metrics studied: precision, recall, f1-score, support, accuracy and execution time.

As future work, we intend to research datasets created from SDN/NFV environments and use them for cross-validation of the proposed approach, since in this work we use synthetic dataset and traffic. Also, we intend to research how to use real traffic in a simulated network or to use a real controlled network for validation and testing. As also, research on different machine learning libraries to compare them. Besides that, use another strategy of extracting statistics to be compared with the one explored in this work, in order to analyze its overall performances in architecture. As a chosen strategy, the POX component web.webcore could be used together with the webservice module from openflow.webservice, which exposes some information between them and the flow statistics. In addition, there is the possibility of using some cache solution that stores the flows that reach the POX Controller, in order to solve the problem found in the requests library. Furthermore, it could be analyzed the advantages and disadvantages of this scenario with caching, since traffic analysis will not take place online.

REFERENCES

Bawany, N. Z., Shamsi, J. A., and Salah, K. (2017). Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42(2):425–441.

Bhushan, K. and Gupta, B. B. (2019). Distributed denial of service (ddos) attack mitigation in software defined network (sdn)-based cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*, 10(5):1985–1997.

Bonfim, M. S., Dias, K. L., and Fernandes, S. F. (2019). Integrated nfv/sdn architectures: A systematic literature review. *ACM Computing Surveys (CSUR)*, 51(6):1–39.

Dias, M. L. D. (2019). fuzzy-c-means: An implementation of fuzzy *c*-means clustering algorithm.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2017). Microservices: yesterday, today, and tomorrow. In *Present and ulterior software engineering*, pages 195–216. Springer.

Duy, P. T., Pham, V.-H., et al. (2018). A role-based statistical mechanism for ddos attack detection in sdn. pages 177–182.

Esch, J. (2014). Prolog to," software-defined networking: a comprehensive survey". *Proceedings of the IEEE*, 103(1):10–13.

ETSI, N. (2017). Network functions virtualisation (nfv); network operator perspectives on nfv priorities for 5g.

Kaur, S., Singh, J., and Ghumman, N. S. (2014). Network programmability using pox controller. In *International Conference on Communication, Computing & Systems (ICCCN'2014)*, pages 134–138.

Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Kumari, R., Singh, M., Jha, R., Singh, N., et al. (2016). Anomaly detection in network traffic using k-mean clustering. In *2016 3rd International Conference on Recent Advances in Information Technology (RAIT)*, pages 387–393. IEEE.

L. Dali, A. Bentajer, E. A. K. A. H. E. E. F. B. A. (2015). A survey of intrusion detection system. In *2nd world symposium on web applications and networking (WSWAN)*, pages 1–6.

Mahjabin, T., Xiao, Y., Sun, G., and Jiang, W. (2017). A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12).

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Ming-Chuan, H. and Don-Lin, Y. (2001). An efficient fuzzy c-means clustering algorithm. In *Proceedings of IEEE International Conference on Data Mining, ICDM-2001*, pages 225–232.

Mouat, A. (2015). *Using Docker: Developing and Deploying Software with Containers*. " O'Reilly Media, Inc.".

Mutu, L., Saleh, R., and Matrawy, A. (2015). Improved sdn responsiveness to udp flood attacks. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 715–716.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.".

Newman, S. (2019). Under the radar: the danger of stealthy ddos attacks. *Network Security*, 2019(2):18–19.

Nugraha, B., Hajizadeh, M., Phan, T. V., and Bauschert, T. (2019). A novel impact analysis approach for sdn-based networks. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 10–18.

Patgiri, R., Nayak, S., and Borgohain, S. K. (2018). Preventing ddos using bloom filter: A survey. *arXiv preprint arXiv:1810.06689*.

Peuster, M., Karl, H., and van Rossem, S. (2016). Medicine: Rapid prototyping of production-ready network services in multi-pop environments. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 148–153.

Preamthaisong, P., Auyporntrakool, A., Aimtongkham, P., Sriwuttisap, T., and So-In, C. (2019). Enhanced ddos detection using hybrid genetic algorithm and decision tree for sdn. In *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 152–157.

Sahoo, K. S., Iqbal, A., Maiti, P., and Sahoo, B. (2018). A machine learning approach for predicting ddos traffic in software defined networks. In *2018 International Conference on Information Technology (ICIT)*, pages 199–203.

Scott-Hayward, S., O'Callaghan, G., and Sezer, S. (2013). Sdn security: A survey. In *2013 IEEE SDN For Future Networks and Services (SDN4FNS)*, pages 1–7. IEEE.

Suresh, M. and Anitha, R. (2011). Evaluating machine learning algorithms for detecting ddos attacks. In *International Conference on Network Security and Applications*, pages 441–452. Springer.

Velmurugan, T. and Santhanam, T. (2010). Performance evaluation of k-means and fuzzy c-means clustering algorithms for statistical distributions of input data points. *European Journal of Scientific Research*, 46(3):320–330.

Vigna, G., Robertson, W., and Balzarotti, D. (2004). Testing network-based intrusion detection signatures using mutant exploits. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 21–30.

Wei, H.-C., Tung, Y.-H., and Yu, C.-M. (2016). Counteracting udp flooding attacks in sdn. In *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pages 367–371.

Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

Yuan, R., Li, Z., Guan, X., and Xu, L. (2010). An svm-based machine learning method for accurate internet traffic classification. *Information Systems Frontiers*, 12(2):149–156.