Attack Detection and Prediction Using Machine Learning

Vaishali Shirsath [1], *Jainil Shah* [2]
Ajay Shah [3], *and Devansh Shah* [4]

Abstract

Data plane and control plane are divided by Software Defined Networking (SDN). A centralized controller oversees and manages the entire network. With SDN, the network may be programmed and flow regulations can be created dynamically. Numerous benefits including adaptability, programmability, and centralized management are offered by this decoupling. However, SDN also creates new vulnerabilities as a result of desired data plane and control plane connectivity. Attacks on switch buffer overflows and control plane saturation are two examples of threats that exploit such flaws. The controller is vulnerable to Distributed Denial of Service (DDoS) attacks, which induce resource exhaustion and impair the controller's capacity to provide services. By flooding the control plane with TCP SYN packets from the data plane (i.e., switches), several attacks can be launched. SVM is the most popular and often used classifier, both for classification and regression, thanks to its high accuracy and low false positive rate. For DDoS detection, the SVM classifier is examined and contrasted with other classifiers. In order to identify anomalies, such as malicious traffic, and report them, Snort, an intrusion detection system, examines the traffic and packets. The entropy approach is used to assess the flow data's randomness. An IP address for the intended recipient and a few TCP flag attributes make up the entropy information. We implement it as an additional module in the Floodlight Controller and assess its viability and efficacy. We thoroughly evaluate how we have implemented things via Mininet, substantial emulation.

Keywords : **Entropy Method, Distributed Denial of Services (DDoS), Machine Learning, Mininet, Software Defined Networks (SDN), Snort, Support Vector Machine (SVM)**

I. INTRODUCTION

The TCP SYN flooding assault is one of the most efficient methods for control plane and target server saturation in software-defined networking (SDN). As there are no forwarding rules in place, the data plane switches are forced to transfer many malicious SYN requests that the attacker generates to the controller. This excessive forwarding strains the computing capacity of both the data plane and the control plane and clogs the communication channel between them.

II. MOTIVATION

The essential advantage of the cloud is that it flexibly scales to fulfill variable needs and it gives the environment which scales up and downsizes quickly, so it needs incredible security from DDoS attacks. These attacks have become more sophisticated and are still evolving quickly. So, identifying and countering these assaults has become a difficult undertaking.

Paper Submission Date : January 27, 2023 ; Paper sent back for Revision : February 17, 2023 ; Paper Acceptance Date : February 23, 2023 ; Paper Published Online : April 5, 2023

[1] V. Shirsath, *Assistant Professor* ; Email : vaishali.shirsath@vcet.edu.in ; ORCID iD : https://orcid.org/0000-0003-2892-526X
[2] J. Shah (*Corresponding Author*), *Student* ; Email : jainilshah.242001@gmail.com ;
ORCID iD : https://orcid.org/0000-0002-0373-2894
[3] A. Shah, *Student* ; Email : shahajay2712@gmail.com ; ORCID iD : https://orcid.org/0009-0003-0070-0306
[4] D. Shah, *Student* ; Email : devanshshah3102001@gmail.com ; ORCID iD: https://doi.org/0009-0006-8196-7775

[1,2,3,4] Department of Information Technology, Vidyavardhini's College of Engineering and Technology, 9RMH+GFX, K.T. Marg, Vartak College Campus Vasai Road, Vasai-Virar, Maharashtra - 401 202.

DOI : https://doi.org/10.17010/ijcs/2023/v8/i2/172775

III. PROBLEM STATEMENT

To create an algorithm that combines different machine learning approaches in order to train a model that is more accurate than each individual machine learning technique used to create the hybrid model in detecting and classifying the type of DDoS attack. A fresh approach for the early detection of TCP SYN flooding is to use the entropy method. A set of guidelines that SNORT should follow when determining whether an incoming packet is malicious can be implemented by the Snort administrator.

IV. LITERATURE REVIEW

A. DDoSs and TCP-SYN Flooding

By depleting resources on a network or host, denial of service (DoS) attacks seeks to disable or degrade network services. Bandwidth on a network, packet-forwarding on a router, server memory, server processing power, and operating system data structures can be the resources that run out. By establishing a TCP connection to the web server and flooding it with pointless TCP packets, this attack can be used to bring down a website. DoS attackers have been known to take down websites in order to offer protection later for a fee. Due to the attacker's constrained computational and network resources, it is not always viable to overwhelm the target's resources with just one computer. However, an attacker could overwhelm the victim by launching a Distributed Denial of Service (DDoS) assault, which uses multiple computers to obtain access to additional resources. One type of DoS attack is TCP SYN flood. A three-way handshake is required for both the client and the server to establish a TCP connection. The client initiates the three-way handshake by sending a SYN message to the server, which responds with a SYN/ACK message to confirm receipt of the SYN message. The client sends the server an ACK to complete the connection formation. Data particular to each service can be exchanged once the connection has been made. In order to create half-open TCP connections, the TCP SYN flood attack spoofs source IPs in SYN messages or ignores SYN/ACKs, so the server never receives the final ACK message, leaving the connection open in part. In order to store half-open connections while awaiting the final ACK, the server must allocate RAM.

B. SNORT, SVM

Snort is an open source NIDS-configurable network intrusion prevention system. The Snort administrator can put in place a set of criteria for what Snort should check for when deciding whether the incoming packets are malicious. Snort uses misuse detection. Users have the option to obtain rules established by the Snort community as well as write their own rules. A packet decoder is used to initially decode packets that enter the computer's network interface controller. The packet decoder examines the packet to determine the protocol being used and whether any deviations from the protocol's rules have occurred. Given that the Snort user has implemented them, the packets are routed to one or more preprocessors after being decoded. Preprocessors are plug-ins that Snort uses to interpret packets that come from the packet decoder in a variety of ways.

C. Mininet

The Mininet has the ability to design SDN components, alter them, share them with other networks, and interact with them. Links, Hosts, Switches, and Controllers are some of these components. On Mininet, a host is a straightforward operating system process that runs its own network environment. Each one makes a virtual network interface, ports, addresses, and routing tables available to processes with exclusive ownership (such as ARP and IP). The packet delivery semantic offered by Mininet's OpenFlow switches is identical to that offered by a hardware switch. Switches can be used in kernel and user space. As long as the system on which the switches are executing has connectivity to the controller, the controllers in a Mininet simulation can run on either a real or simulated network. In the local simulation environment, the Mininet can generate a standard controller at the user's request. Virtual connections can also be established between the elements via their virtual interfaces. To assess the randomness of the flow data, use the entropy approach. The destination IP address and a few TCP flag attributes are included in the entropy data. We include SAFETY as an extension module into Floodlight Controller and test it out in various conditions to demonstrate its viability and efficacy.

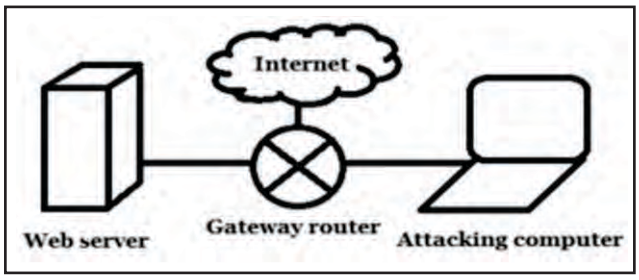

Fig. 1. Experimental Setup

V. EXPERIMENTAL SETUP

On a machine running Ubuntu, a virtual machine was set up to serve as the target of all network assaults carried out for this investigation. This server is referred to as the web server throughout this essay. Your first and last names could be entered through the straightforward user interface. You were unable to enter your name if the server refused service. This was done to see if the web server was still operational during an attack. Through a gateway router, the web server was linked to the Internet. Another computer was used to play the role of the attacker on the same network, which was also linked to the internet through the same gateway router. All attacks were executed from this computer. Figure 1 depicts the surroundings of the experiment.

A. Create a New Virtual Machine

To build a fresh virtual computer, click New. Fill up the necessary information:
Name: The Type and Version will automatically update if your name contains the term Ubuntu.
Computer Folder: Your virtual machines will be kept here so that you can pick up working on them whenever you want.
Type: Linux

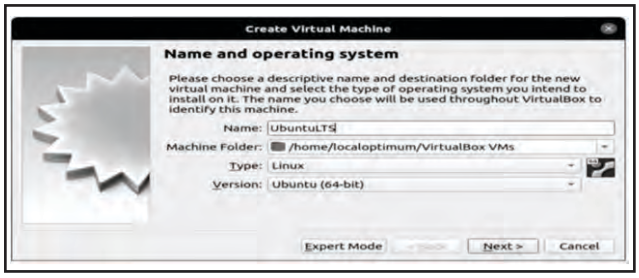

Fig. 2. Create a new Virtual Machine

Release: Ubuntu (64-bit)

B. Implementation of Snort

1) Download and Extract Snort:

Visit the snort website to access the most recent version for free. Extract the snort source.

2) Install Snort

3) Verify the Snort Installation

4) Installation of Hping for DDOS attack generation

Fig. 3.Creating a Mininet Topology

C. Creating a Mininet Topology With One Controller, One Switch, Two Hosts

The minimum topology, which is the default topology, consists of the OpenFlow reference controller, one OpenFlow kernel switch coupled to two hosts. You may also provide this topology on the command line with —topo=minimal. For further details on the other topologies that are pre-installed, refer to the —topo section in the mn -h output.

All four entities are currently hosted by the VM (1 basic controller, 2 host processes, and 1 switch process). For installing the controller outside of the VM, follow the instructions at the bottom. In the absence of a defined test input, the Mininet CLI starts.

You ought to be able to see the kernel switch connected to the reference controller in the Wireshark window.

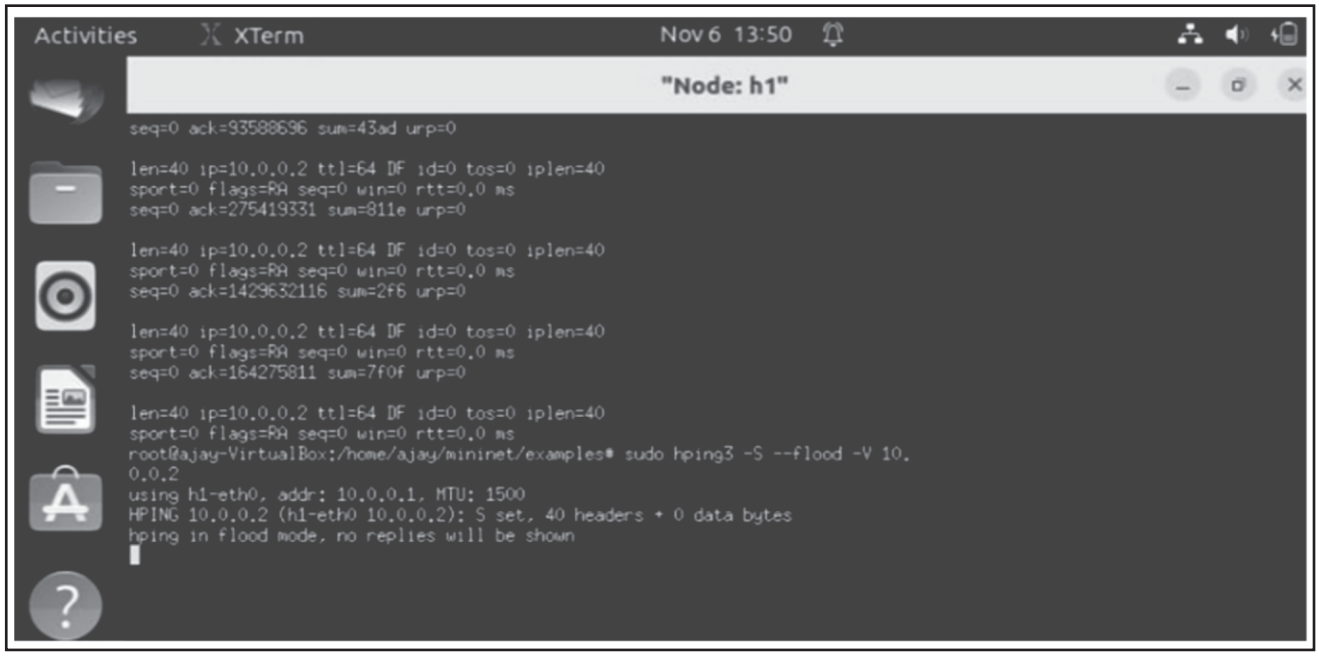

Fig. 4. Attack initiated

1) Attack Initiated : On the attacking system, hping3, an application to carry out DoS attacks was deployed. Hping3 was used to perform three distinct TCP SYN flood attacks against the web server. The first attack was the simplest; it attacked the web server with as many TCP SYN packets as quickly as it could on port 0, which is Hping3's default port for sending traffic. The source IP address of the second attack's TCP SYN packets was changed, but it used the same frequency and port as the first attack. Except for sending packets through port 80, the third attack was identical to the second attempt.

2) Running Snort for Attack Detection: The command-line options used in this command are:

-d: Filters out the application layer packets

Fig. 5. Running Snort for attack detection

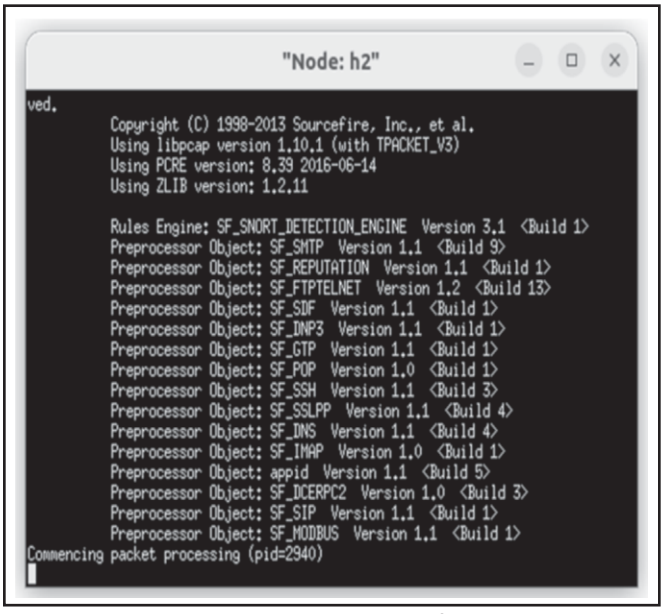

Fig 6. Snort started

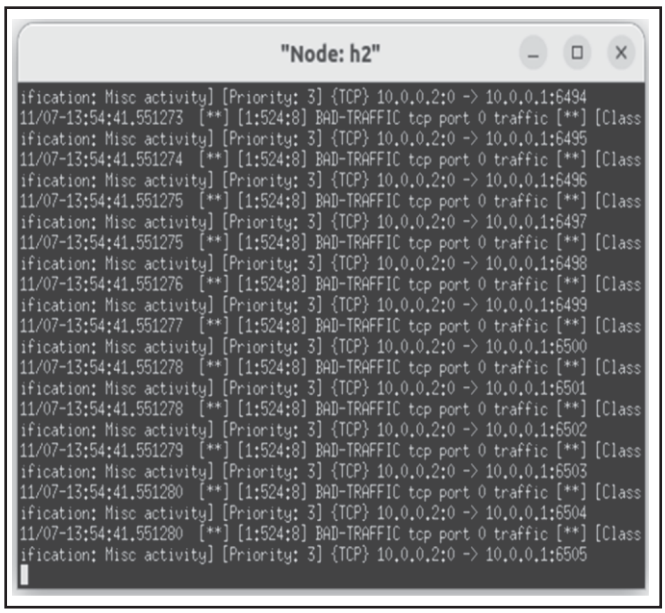

Fig. 7. Attack detected

-l /var/log/snort/: Sets the logging directory

-A console: Sends alerts to the console window

-c/etc/snort/snort.conf: Indicates which Snort configuration file to use

3) Snort Started / Attack Detected: Network traffic that Snort recognizes as potentially malicious is recorded in the logs and alarms are sent to the console window. An effort has been made to scan a machine for data that could be useful to an attacker when an attack is designated as *Information Leaks assault.* This implies that someone is most likely spying on your system. *Denial of Service* attacks aim to saturate your computer with unreliable network traffic. The attack seeks to overwhelm your computer to the point where it is unable to perform as intended. To make sure promiscuous mode is operating properly and that the entire network address range is being protected, we will send some malicious data to a different machines and see if Snort identifies it.

We were surprised to see that SNORT used the registered rule set to identify the TCP SYN flood attacks. Given how frequently the TCP SYN flood assault occurs, we anticipated that it would be recognized. However, all three TCP SYN flood assaults were found using the custom rule 4.1 as we had anticipated. We set the number of packets to an unusually high figure for our server in such a short time interval because this rule considers how many packets are sent in each time interval. To accommodate the traffic on your network, you can adjust the number of packets and the time frame. However, in order to utilize this rule, it is crucial to understand how much traffic your network typically receives. This way, it won't alert you when an usual quantity of SYN packets are received. To explore if SNORT might be avoided, this attack was tweaked to target several ports, however, it had no effect.

VI. USING SVM TO DETECT DDoS ATTACK IN SDN NETWORK

A. System Overview

The intrusion detection system needs to be fed with traffic data in order to detect the DDoS attack. The widely used SVM classifier is employed by the system to identify the assault. SVM can identify a pattern from a small number of training samples and by minimizing false positive data, it creates an accurate classification. This is accomplished by using generalization capability, which is the trained machine's capacity to categorize unfamiliar samples using the model discovered from training datasets. SVM never settles for the local optimum and instead always finds the global optimum solution. By identifying the best hyperplane (biggest margin) that divides two classes, SVM conducts linear separation. Support vectors are the training samples that are nearer to the hyperplane.

SVM is a linear classifier that supports nonlinear classification using kernel functions. The linear, polynomial, radial basis function, and sigmoidal kernel functions are frequently employed.

SVM is a widely used classifier that has a low rate of false positives and high accuracy. We evaluate the SVM classifier for DDoS detection and contrast it with other classifiers. According to the experiments, SVM produces more accurate categorization than the competition.

B. DDoS Detection Based on Support Vector Machine (SVM)

The Openflow switch in the SDN architecture quickly forwards the main network data. The collecting of switch traffic data, administration of the forwarding decision, and forwarding are all tasks that fall under the purview of the SDN controller. The flow table serves as the fundamental data structure for the forwarding policy

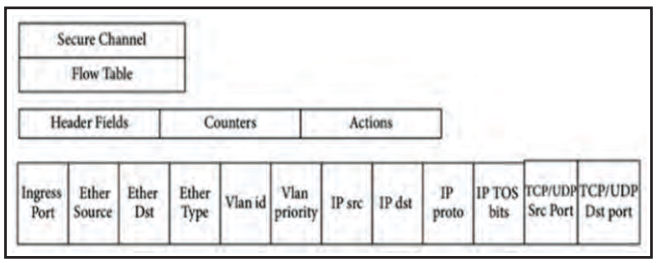

Fig. 8. Flow table entry structure diagram

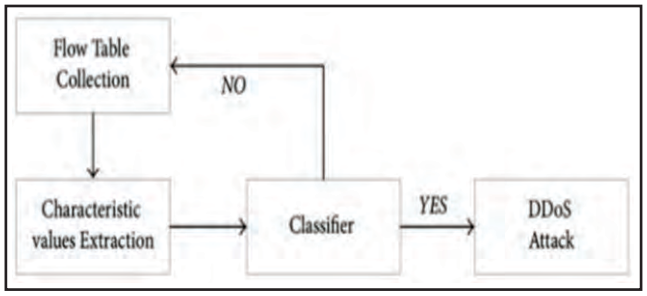

Fig. 9. Flow diagram

management control in the SDN switch. The flow table entries which can forward the packet to one or more interfaces are searched by the SDN to handle the pertinent network traffic. Each entry has a header field as well as counters and actions. The flow table serves as the foundation for the switch's packet routing. Multiple flow entries make up each flow table. The rules for data forwarding are formed by the flow table entries.

C. Flow Diagram and Implementation

The majority of the attack detection flow diagram is made up of the extraction characteristic values, classifier judgment, and flow state gathering. The OpenFlow switch responds with the information from the flow table after the flow state collection periodically asks a flow table from it. The six-tuple characteristic values are produced by the characteristic values extraction, which is primarily responsible for obtaining the characteristic values related to the DDoS attack from the switch flow table matrix. In order to distinguish between normal traffic and attacking abnormal traffic, information regarding six-tuple characteristic values is classified using an SVM-based method.

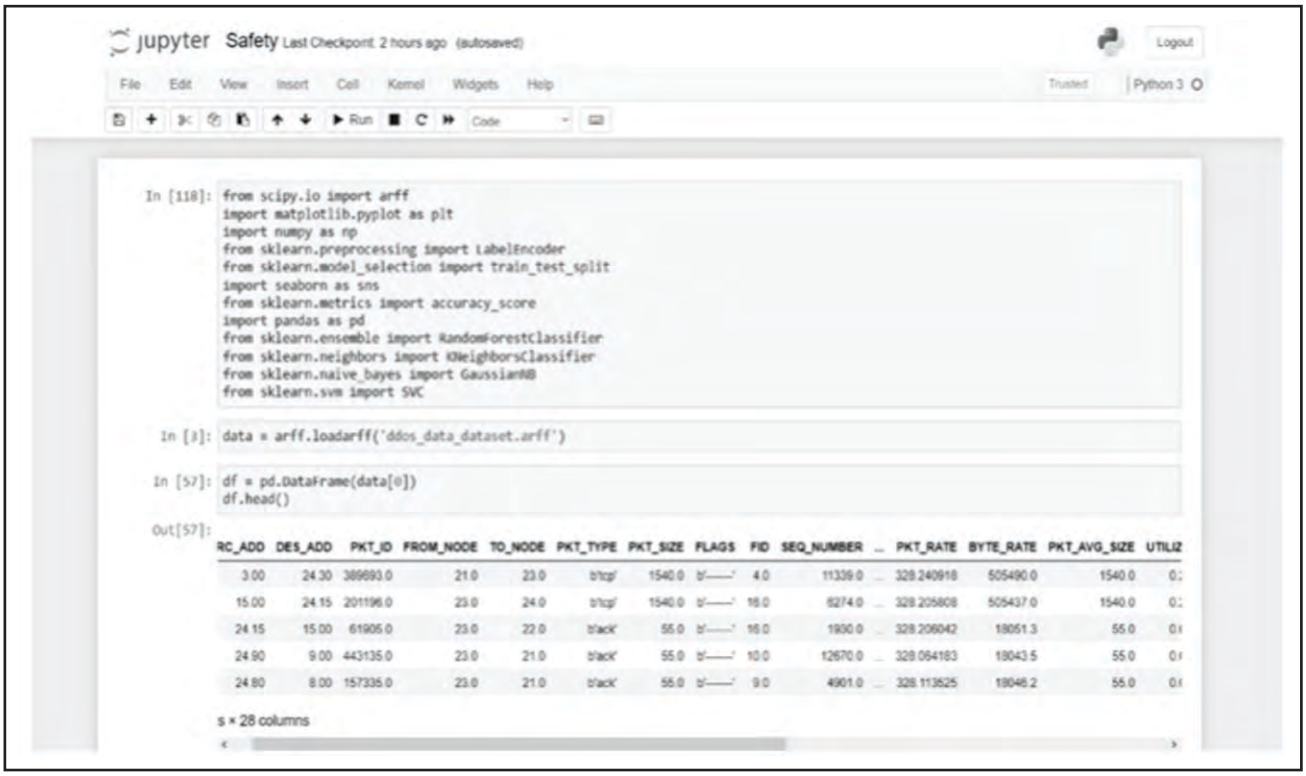

Fig. 10. Importing Libraries and Dataset

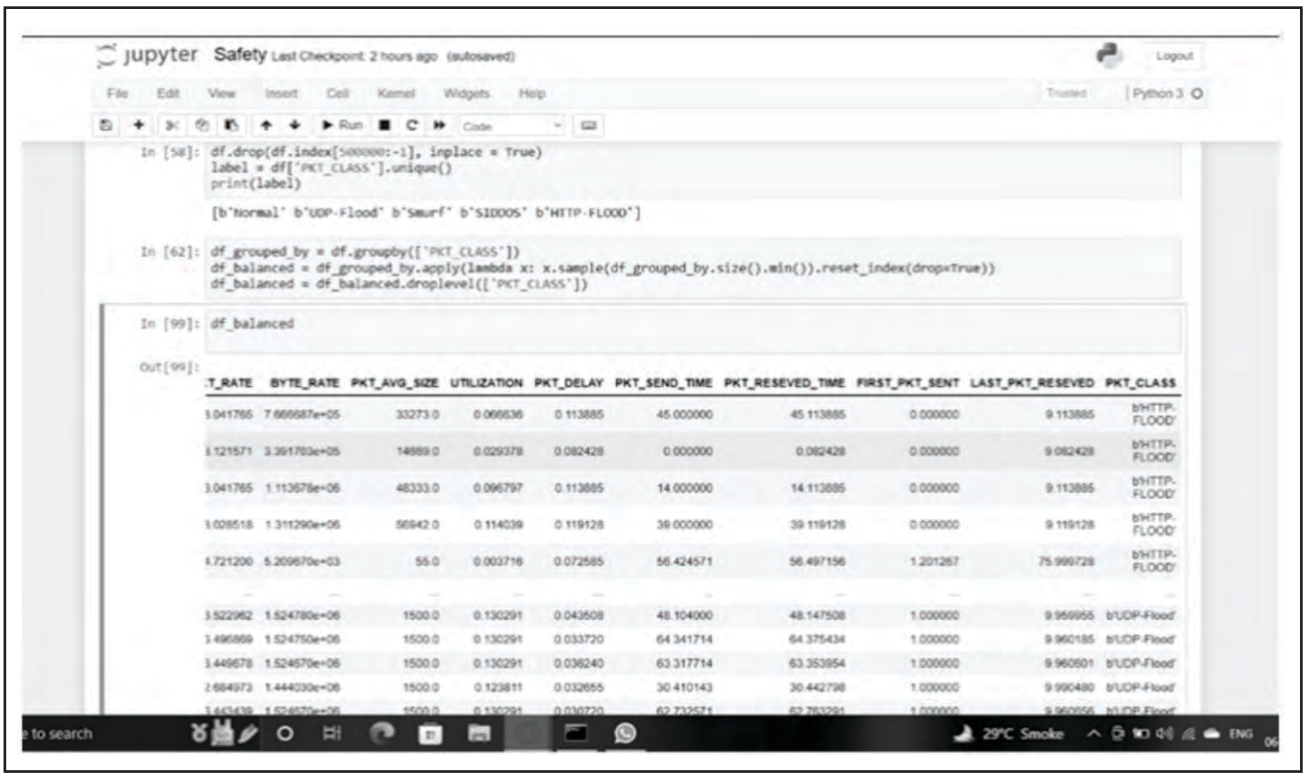

Fig 11. Exploring Dataset

```
In [9]: X_train, X_test, Y_train, Y_test = train_test_split(x,y,test_size = 0.2, random_state = 42)

In [30]: model = SVC(gamma = "scale")
         model.fit(X_train,Y_train)

Out[30]: SVC()
         In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
         On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [31]: y_pred = model.predict(X_test)

In [32]: print(accuracy_score(y_pred,Y_test))

         0.6014056224899599

In [35]: clf = RandomForestClassifier(n_estimators = 100)
         clf.fit(X_train, Y_train)
         y_pred = clf.predict(X_test)
         from sklearn import metrics
         print()
         print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(Y_test, y_pred))

         ACCURACY OF THE MODEL:  0.8042168674698795
```

Fig 12. Model evaluation and prediction

VII. ENTROPY

A. Measure of Random Variable

A random variable for its randomness or uncertainty is measured using the information theory concept of Shannon's entropy. A measure of the information content linked to a random variable is another method to describe it. Higher entropy results from more random variables, and vice versa. formally defined. Equation (1) depicts a window where the random variable, x_i, and its frequency, y_i are both given. Eq. (2) is used to calculate the entropy, where $P(x_i)$ stands for the probability that each random variable in the set will occur.

$$W = \{(x_1, y_1),(x_2, y_2),(x_3, y_3),...,(x_n, y_n)\} \quad (1)$$

$$P(x_i) = y_i / N \quad (2)$$

$$N = y_1 + y_2 + y_3 + \cdots + y_n \quad (3)$$

where, N is the total number of occurrences of all the outcomes. A discrete random variable (X) in a system is said to have an entropy of:

$$E(X) = \sum_{i=1}^{n} -P(x_i) \log_2 P(x_i) \quad (4)$$

where, n is the number of outcomes that are unique (i.e., unique destination IP). Since entropy falls in the range of $[0, \log_2 n]$, we adjust the entropy to ensure that it remains into the same range of values, i.e., [0, 1], irrespective of the sample size of the window {eq. (5)}.

$$E^N(X) = \frac{E(X)}{\log_2 n} \quad (5)$$

B. Key Observations

For a typical traffic scenario, flow entropy will typically be higher due to the uniform distribution of SYN packets for a destination IP, but as soon as SYN flooding is active, attack flows will predominate, and throughout the duration of the attack period, a continual significant fall in entropy will be seen. This is due to the fact that a particular destination IP's (victim node's) packet concentration will be significantly higher than that of the other destination IPs. Therefore, when entropy suddenly drops, it creates a state of concern.

VIII. CONCLUSION

In order to curb the attacks, different techniques and models are needed to be used. In comparison to other methods, the SVM classifier has a lower false positive rate and a higher classification accuracy. However, SVM takes more time to train and generate the detection model, which is used to predict the traffic characteristics. Snort presented few weak spots during the attacks in this study. Since Snort depends so much on the user, we believe that the biggest weakness could be the user's stupidity. Snort's rules determine exactly which incursions it finds. This highlights another weakness of Snort that it is based on misuse detection and has problems dealing with novel attacks. Before Snort can identify an attack, it needs to be aware of its structure. We demonstrate that based on traffic characteristics such as clubbing of destination IP addresses along with TCP flags and following a series of time-based windowing of packets, one may efficiently calculate the entropy to evaluate the degree of randomness of the packets that are received at the SDN controller.

IX. FUTURE WORK

Further efforts will be required to reduce the number of messages exchanged and the controller's saturation from these connection requests as every type of TCP connection is directed at the controller (for creating centralized statistics). Additionally, since real traffic is more random than synthetic traffic, we intend to use it for security experiments. The performance of the SVM classifier can be improved when combining AVL tree and SVM. An AVL tree structure will be used to organize the multiple binary SVM. The reduced testing period is due in part to the AVL tree's capacity to balance height.

AUTHORS' CONTRIBUTION

Prof. Vaishali Shirsath guided and helped throughout the research. Jainil Shah prepared the paper, worked on literature review and worked on the Machine Learning part in the project. Ajay Shah worked on the collection of industry information, and worked on the project in attack detection. Devansh Shah prepared the paper and reviewed it. All the authors collectively finalized the paper.

CONFLICT OF INTEREST

The authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in the manuscript.

FUNDING ACKNOWLEDGEMENT

The authors have not received any financial support for the research, authorship, and/or for the publication of the paper.

REFERENCES

[1] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," in *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1545–1559, Dec. 2018, doi: 10.1109/TNSM.2018.2861741.

[2] Kokila RT, S. Thamarai Selvi, and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," *in 2014 6th Int. Conf. Adv. Comput.*, Chennai, India, 2014, pp. 205–210, doi: 10.1109/ICoAC.2014.7229711.

[3] A. Garg and P. Maheshwari, "Performance analysis of Snort-based Intrusion Detection System," in *2016 3rd Int. Conf. Adv. Comput. Commun. Syst.*, Coimbatore, India, 2016, pp. 1–5, doi: 10.1109/ICACCS.2016.7586351.

[4] N. Ravi, S. M. Shalinie, C. Lal, and M. Conti, "AEGIS: Detection and Mitigation of TCP SYN Flood on SDN Controller," in *IEEE Trans. Netw. Service Manage.* vol. 18, no. 1, pp. 745–759, Mar. 2021, doi: 10.1109/TNSM.2020.3037124.

[5] B. M. Khammas, S. Hasan, R. A. Ahmed, J. S. Bassi, and I. Ismail, "Accuracy Improved Malware Detection Method using Snort Sub-signatures and Machine Learning Techniques," *in 2018 10th Comput. Sci. Electron. Eng.*, Colchester, UK, Sep. 2018, pp. 107–112, doi: 10.1109/CEEC.2018.8674233.

[6] B. Nagpal, P. Sharma, N. Chauhan, and A. Panesar, "DDoS tools: Classification, analysis and comparison," in *2015 2nd Int. Conf. Comput. Sustain. Global Develop.,* New Delhi, India, 2015, pp. 342–346.

[7] S. Woo, S. Lee, J. Kim, and S. Shin, "RE-CHECKER: Towards Secure RESTful Service in Software-Defined Networking," in *2018 IEEE Conf. Netw. Function Virtualization Softw. Defined Netw.*, Verona, Italy, 2018, pp. 1–5, doi: 10.1109/NFV-SDN.2018.8725649.

[8] J. Ali, B. -h. Roh, B. Lee, J. Oh, and M. Adil, "A Machine Learning Framework for Prevention of Software-Defined Networking controller from DDoS Attacks and dimensionality reduction of big data," in *2020 Int. Conf. Inf. Commun. Technol. Convergence*, Jeju, Korea (South), 2020, pp. 515–519, doi: 10.1109/ICTC49870.2020.9289504.

[9] H. Meigen and C. Yunqiang, "A DDoS attack detection method based on time series and random forest in SDN," in *2021 Int. Conf. Intell. Comput., Automat. Syst.*, Chongqing, China, 2021, pp. 323–327, doi: 10.1109/ICICAS53977.2021.00073.

[10] T. S. Chu, W. Si, S. Simoff, and Q. V. Nguyen, "A Machine Learning Classification Model using Random Forest for detecting DDoS attacks," in *2022 Int. Symp. Netw, Comput. Commun.*, Shenzhen, China, 2022, pp. 1–7, doi: 10.1109/ISNCC55209.2022.9851797.

About the Authors

Vaishali Shirsath is Assistant Professor with Vidyvardhini's College of Engineering and Technology in the Department of Information Technology. She has a total 18 years of experience in teaching.

Jainil Shah is a final year student of the Information Technology field. He is eager to develop software that will positively impact the world.

Ajay Shah is a final year student of the Information Technology department who enjoys learning new things.

Devansh Shah is a final-year student of University of Mumbai (Information Technology department) who is keen about new technologies.

INDIAN JOURNAL OF COMPUTER SCIENCE

Statement about ownership and other particulars about the newspaper "Indian Journal of Computer Science" to be published in the 1st issue every year after the last day of February.

FORM 1V
(See Rule 18)

1.	Place of Publication	:	NEW DELHI
2.	Periodicity of Publication	:	BI-MONTHLY
3.	4,5 Printer, Publisher and Editor's Name	:	S. GILANI
4.	Nationality	:	INDIAN
5.	Address	:	Y-21,HAUZ KHAS, NEW DELHI-16
6.	Newspaper and Address of individual	:	ASSOCIATED MANAGEMENT
	who owns the newspaper and partner	:	CONSULTANTS PRIVATE LIMITED
	of shareholder holding more than one percent.	:	Y-21, HAUZ KHAS, NEW DELHI-16

I, S. Gilani, hereby declare that the particulars given above are true to the best of my knowledge and belief.

Dated : March 1, 2023

Sd/-
S. Gilani
Signature of Publisher