International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/211627 Volume 9, Issue 1, January – February (2022)

RESEARCH ARTICLE

Improving Performance and Efficiency of Software-Defined Networking by Identifying Malicious Switches through Deep Learning Model

Thangaraj Ethilu
Department of Computer Science and Engineering, Annamalai University, Chidambaram, Tamil Nadu, India
ethilthangaraj@yahoo.co.in

Abirami Sathappan
Department of Computer Science and Engineering, Annamalai University, Chidambaram, Tamil Nadu, India
reachabisv@gmail.com

Paul Rodrigues
Department of Computer Science and Engineering, King Khalid University, Abha, Saudi Arabia
drpaulprof@gmail.com

Received: 05 December 2021 / Revised: 08 January 2022 / Accepted: 13 January 2022 / Published: 28 February 2022

Abstract – In recent times, Software Defined Networking (SDN) has developed widely to provide capable solutions for future internet services. As with the solutions, SDN brings us a hazardous rise in malicious threats. We investigated a sort of Distributed Denial of Services (DDoS) assault known as an internet services attack, which evaluates the influence of both traffic flow and throughput depletions in order to characterize the abnormalities. This sort of attack has a significant impact on the whole SDN. This paper introduces a deep learning method to improve the performance efficiency of the SDN by classifying the network switch into either a trusted switch or a malicious switch device. In this research, an attack detection methodology for Internet services utilizing Software Defined Networking (SDN) is proposed. The SDN controller may evaluate traffic flow, detect anomalies, and restrict both incoming and outgoing traffic as well as source nodes. The SDN considers a Convolutional Neural Network (CNN) based attack detection system that can identify malicious node. Kaggle datasets are used to test and train CNN and the features such as packet duration, packet count, byte count, accuracy for identifying the flow of trusted and malicious switches. According to the results, the CNN-based attack detection system can identify the attack with an accuracy of 89 percent. The comparison evaluation with the already proposed LeNet CNN of the feature classification proves that the flow is the trusted one and with the constant throughput with the help of the deep learning model.

Index Terms – Software Defined Networking (SDN), Kaggle Dataset, Convolutional Neural Networks (CNN), Keras, Internet Service Attack, Malicious Switches, Malicious Node, Distributed Denial of Services.

1. INTRODUCTION

Deep Learning Model focuses on the trusted and the malicious [1-4] or untrusted switch detection for better network performance [5]. This is carried out through the Software Defined Networking (SDN) using deep learning architecture. The basic deep learning model is illustrated in below Figure 1.

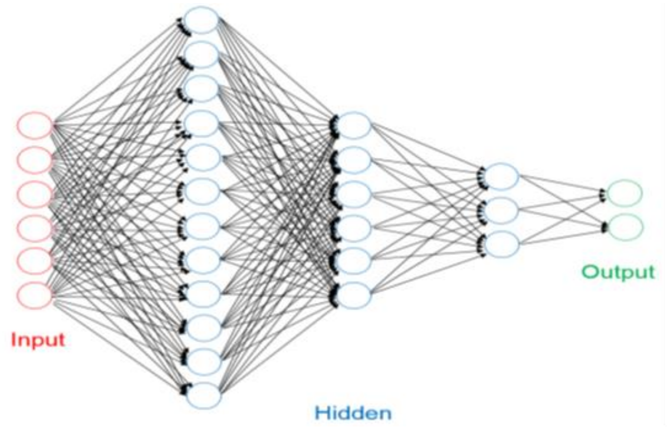


Figure 1 Basic Deep Learning Model

This work is based on two stages training and testing stages. In the training stage, the computations of trusted and malicious network switches relate to the SDN controller for the external feature charts. They are trained by the Modified Convolutional Neural Networks (CNN) with LeNET datasets to produce the trained patterns. In the testing stage, the computations of trusted and malicious network switches relate to the SDN controller for the external feature charts specially used for checking malicious activities. Modified Convolutional Neural Networks (CNN) with LeNET datasets will produce the categorized response as a trusted or untrusted network switch [6]. This Architecture helps in the classification of the unidentified switch to either trusted or malicious switch where external and internal feature charts are required. The internal maps are computed by the architecture itself but external needs to be computed outwardly from the network switch device where the external process is determined. Through the deep learning modified architecture, the untrusted switches are detected for the better performance of the SDN [7]. Here the external feature charts are computed for trained and categorized from the switch device through Convolutional Neural Networks (CNN).

Similarly, our proposed work focuses on the deep learning environment using the Kaggle dataset. The best feature selection for the identification and detection of trusted and untrusted switches. We created a deep learning environment using TensorFlow and Keras environment. Here, CNN will be used for text classification and feature importance. The flow of trusted and untrusted switches and the security of the attacker node will be determined.

1.1. Problem Statement

In our proposed system, larger scalability issues of Software Defined Networking (SDN) due to lower throughput and high latency are addressed. The partition of the control plane and the data plane is a contributor to the above-mentioned issues of SDN architecture especially the scalability of a control plane. This decoupling is carried out by the controller (i.e., pox controller) which helps in the classification of network devices. The process of identifying the trusted or untrusted network by classifying the features of the packet in an efficient manner through constant throughput. A deep learning environment is a process followed for analyzing and implementing the network switch. A throughput comparison study with the existing LeNet architecture helps to determine the trusted flow of network switches efficiently.

1.2. OBJECTIVE

The suggested technique takes into account the following goals:

1. To identify malicious switches using a deep learning model and to increase Software Defined Network performance and efficiency by recognizing malicious and trusted switches through the Kaggle dataset.

2. A novel method of attack detection methodology for Distributed Denial of Services (DDoS) attacks in internet services is created using entropy values and Principal Component Analysis (PCA) based on Convolutional Neural Networks (CNNs).

3. To implement CNN and their characteristics such as packet duration, packet count, byte count, and accuracy for distinguishing the flow of trustworthy and malicious switches are tested and trained.

4. The performance of the proposed SDN system is compared to that of the present LeNeT architecture. The SDN performance may be determined by using constant throughput and comparing it to an existing LeNet architecture.

The Section 2 defines the related literature survey of the existing systems. Section 3 describes about the methods and materials used in the analysis part. The 4th section defines the complete information of the proposed approach and the SDN implementation and 5th Section describes our experimental results. Section 6 will finally conclude the paper with future expansions.

2. LITERATURE SURVEY

Oliveira et al. [1] focus network’s control plane approach for energy efficiency. It is capable of parallel processing which is splitting of multi-tasks of the controller through the multicore processor. Through this division of tasks among the similar cores, the frequency can be lowered can help in lowering the consumption of energy by not keeping the same quality of service level. Here, they used the multicore processor to save the energy consumed without compromising the level of service. They performed many standard measures like latency [8]; throughput and standard energy efficiency metrics for data centers namely Communication Network Energy Efficiency (CNEE) metric [9, 10]. Comparing a single core with this multicore processor gives better efficiency performance. To show the effects of handling the packets on the controller energy consumption SDN model is presented. The motivation on energy savings behindhand the multicore controller is conferred.

Mohsin et al. [2] in their paper the controller pane is determined through the fewer power nodes. Here, the meta-heuristic Software Defined Networking (SDN) approach is evaluated for shortest path selection in Wireless Sensor Networks (WSN). To overcome the problems of exploration and exploitation [11, 12] a new modified algorithm Dolphin Echolocation Algorithm (DEA) is created. This algorithm will help to find the efficient path of the nodes. In this paper, the residual selections of nodes are considered to select the desired shortest path.

Madhukrishna et al. [3] in their paper need of the energy-efficient algorithm is determined. They have focused on active and sleep mode in their consumption model for the large and wide SDN applications. For the real-time application, this energy consumption routing algorithm helps to reduce energy. A framework for energy efficiency for Software Defined Networking (SDN) is introduced in this paper. Efficient Routing Algorithm Selection (ERAS) is the algorithm used to perform energy consumption [13] calculation and link load optimization. This optimization is used by all the neighbor nodes of the controller plane with the help of link state data.

Naeem et al. [14] developed a malware classification system (MICS) that collected hybrid characteristics from malware before classifying it using Support Vector Machine (SVM). Using only samples of 9339 from malware 25 families, they were able to obtain a classification accuracy of 97.4%. Their strategy for small-scale analysis attained an accuracy of 99.6% using just samples of 5116 from malware of ten families. To categorize metamorphic malware J48, Logistic Regression, and Naive Bayes is used to mining common sub-graphs from metamorphic malware operation code graphs. Its goal is to reduce the need for human expertise as well as the associated costs [15]. Andrzej et al. [16] discuss the malicious flow of switches in SDN. Openflow method was carried in this proposed method to detect and identify the untrusted flow of switches. According to this method, packet dropping and packet swapping methods are classified to determine the switch flow. To address the problem of vulnerability and network security new algorithm was created. Two vulnerable behavior were identified to resolve the untrusted flow of switches. Here, the controller will analyze the statistics periodically to help the features of packet count, byte count, and duration of flow be determined. Yan et al. [17] created a deep neural network that learned features from malicious files using CNN and Long-Short Term Memory (LSTM) networks. It significantly lowered the cost of creating artificial features. By expressing malware executable as entropy value streams, Gibert et al. [18] transformed the malware classification issue into a time series classification problem, and then utilized a CNN classifier to discover optimum discriminant sequels of the time classification. This approach had a 98.28 percent accuracy for the Microsoft dataset. However, it will not be effective for malware families with a tiny number of samples.

3. DEEP LEARNING MODEL-ANALYSIS SECTION

In this proposed system, we divide the process into two parts: one is data analysis part and SDN implementation parts. The data analysis part is carried out through a deep learning environment using Tensorflow and Keras environment [19]. The dataset used in this process is the Kaggle dataset. We take Kaggle data for SDN trusted and untrusted flow. The flowchart of the proposed analysis part is shown in Figure 2.

3.1. Deep Learning Environment with Keras

In general, deep learning is a machine language used to detect the features in similes. This deep learning uses Convolutional Neural Networks (CNN) which can work as a brain in a human being. It consists of many layers where each layer can excerpt one or many distinctive features. For machine language, it is very difficult to identify the structure as trusted or untrusted. It is more acute to feature extraction maps to be automated using machine language. To overcome real-world issues, deep learning is used to analyze and identify objects in similes. Here, the process is circulated in a well-timed practice.

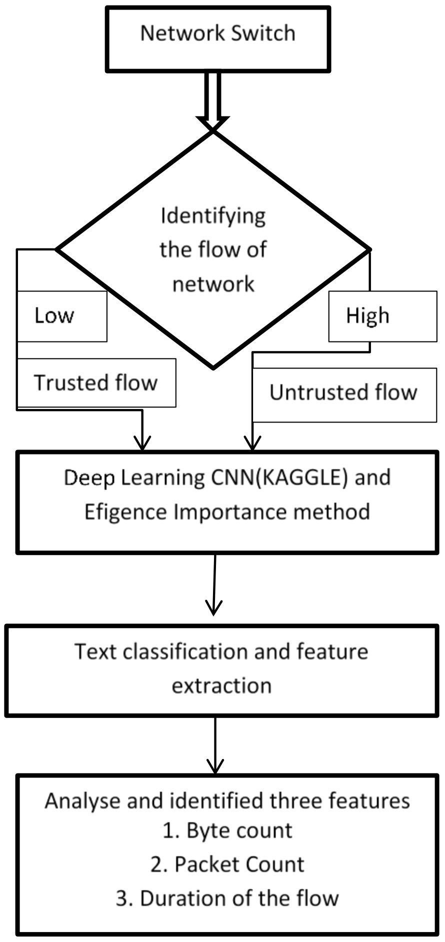

Network Switch
Identifying the flow of network
Low
High
Trusted flow
Untrusted flow
Deep Learning CNN(KAGGLE) and Efigence Importance method
Text classification and feature extraction
Analyse and identified three features
1. Byte count
2. Packet Count
3. Duration of the flow

Figure 2 Flowchart of Analysis Part

Figure 2 clearly describes the flow of trusted or untrusted network switches. After the network switch, nodes and controller are initiated, the Software Defined Networking (SDN) will identify the flow of the network. Through analyzing, three features are identified [20].

If the flow of duration is high, then the network switch will be untrusted, and low means flow will be trusted. Some of the deep learning frameworks include TensorFlow and Keras. Our proposed system is carried out in Keras framework to excerpt exterior feature plots.

3.2. Feature Extraction

The main goal of text classification is to categorize text into applicable classes or forms. This system comprises a feature extraction tool that works out several plain text docs and a classifier through prior data of the labelled data [21]. The classification of features is based on the trusted and untrusted flow of switches. As the analysis part progresses, the packet is taken as input where three features are determined. They are byte count, packet count, and duration of the flow. In this proposed work, we divide the process into two parts data analysis and Software Defined Networking implementation parts. In the analysis part, the deep learning environment is created using TensorFlow and Keras environment. Keras is providing a firm user investigation. Keras is now integrated with Tensorflow Application Program Interface (API). When the flow starts trusted and untrusted flow is identified. If the duration of flow is high means untrusted flow and if the flow is low means trusted flow. The maximum and minimum flow of those features are tabulated in below Table 1.

Feature	Max	Min
Packet Count	350000	4000
Byte Count	147128002	23814
Duration	800	250

Table 1 Min and Max Features

3.2.1. Entropy Values

Entropy is the notch of uncertainty. The level of the disorder can be computed through the entropy of the information. A higher level of entropy denotes complex or high uncertainty and the most disorderly system. Low entropy information provides the possibility to guess fore coming engendered values. Entropy [22] is computed using the below equation (1).

$$(2.25) \times \mathrm{N} \int \quad \min(\mathrm{X}) \times \max(\mathrm{X}) \times \mathrm{P} \times \log 1 \mathrm{pxdx} \qquad (1)$$

Where P(X) is the probability density function (PDF) of the X(n) switch. Entropy is regarding the maximum and minimum amount of data or information converged by a certain bit. The entropy e is known through the below equation (2).

$$(1.14)\mathrm{e} = 1\mathrm{npDilNpDi} \qquad (2)$$

Where D is distribution, N elements and p(i) is the probability of the element. The natural log is defined as log2 in binary, and the equation is stated as follows (3).

$$\mathrm{np}(1.15)\mathrm{e} = 1\mathrm{npp\ Di\ log2pDi} \qquad (3)$$

Where log 2(N) is the maximum value of entropy.

3.2.2. Min-Max Level

The maximum and minimum values of entropy features such as packet count, duration flow, and byte count. The minimum and maximum of all features are computed as shown in below Table 1. This analysis is carried out for the trusted flow [23] of switches.

3.2.3. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a mathematical process that reduces a large number of (potentially) linked variables to a smaller number of unconnected variables known as principal components. The initial principal element accounts for as considerable variation in the data as feasible, with each subsequent component accounting for as much variability as possible. PCA is a technique for emphasizing variance and highlighting significant outlines in a dataset. It's frequently used to make statistical data easier to understand and comprehend. The CNN architecture is used for categorizing trusted and untrusted networks from an unknown network requires both interior and exterior maps. The interior map feature will be produced by itself whereas exterior maps are computed from the switch [4]. The weight of the feature chart is computed through the below equation (4).

$$WFM = \frac{\alpha_i * \beta_i}{w_i} \qquad (4)$$

Whereas α_i is the index factor of the alpha and β_i is the index factor of the beta with the computation weight w_i. The index factor of the alpha (α_i) is calculated using the below equation (5) which denotes the unknown network whose packet flow of controller and SDN controller is obtained.

$$\alpha_i = \frac{P_s * P_r}{N} \qquad (5)$$

Whereas P_s is the total number of successfully transmitted packets from SDN controller(c) to switch(s) over t is time't', P_r the number of packets received the successfully at the controller from switch and finally N is the number of packets transmitted from s to c. The index factor of beta (β_i) is calculated based on below equation (6) which is computed between controller and switch of the dropped packet.

$$\beta_i = \frac{d_{sc}}{N} * E_s \qquad (6)$$

Whereas d_{sc} is the total packets dropouts from c to s and E_s is the switch's energy consumption. The SDN controller weight index (w_i) is calculated using the below equation (7).

$$w_i = \frac{\sum_{i=1}^{N1} P_{ic}}{N1} \qquad (7)$$

Whereas P_{ic} is the received packets of the SDN controller from its switches surrounded and N1 is the total number of adjoining switches around SDN controller. This feature computation is stored as a matrix called Feature Map (FM) and is collaborated with CNN for detecting trusted and untrusted switch flow in Software Defined Networking (SDN).

3.3. Classification Factors

The byte count is computed using a number of bytes. The byte count for trusted (1) and untrusted switch (0) is illustrated in below Figure 3. From this, we set entropy for trusted (1) and untrusted switch (0). Packet count is nothing, but the total number of packets accumulated. Packet count is one method of classifying traffic. Packet count of the trusted (1) and untrusted (0) flow are shown in the below figure 4. The maximum and minimum of the feature is computed in the above Table 1. Packet count is nothing, but the total number of packets accumulated. Packet count is one method of classifying traffic. Packet count of the trusted (1) and untrusted (0) flow are shown in the below Figure 4. The maximum and minimum of the feature is computed in the above Table 1.

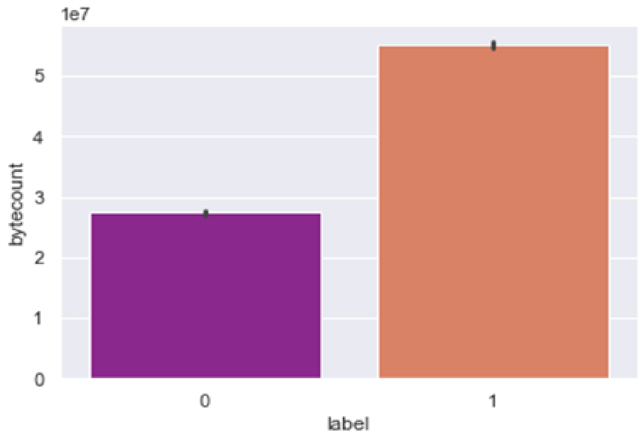


Figure 3 Byte Count

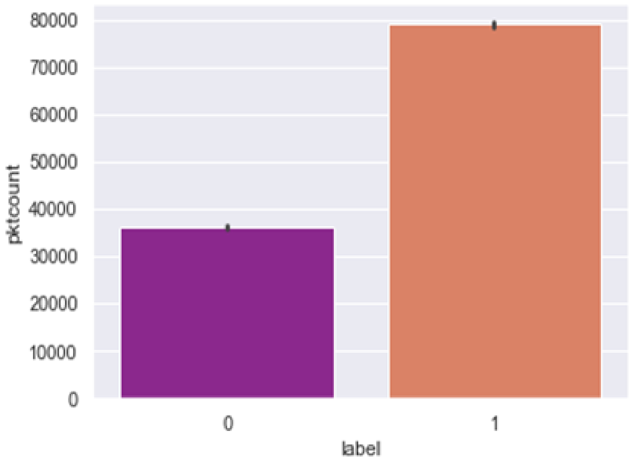


Figure 4 Packet Count

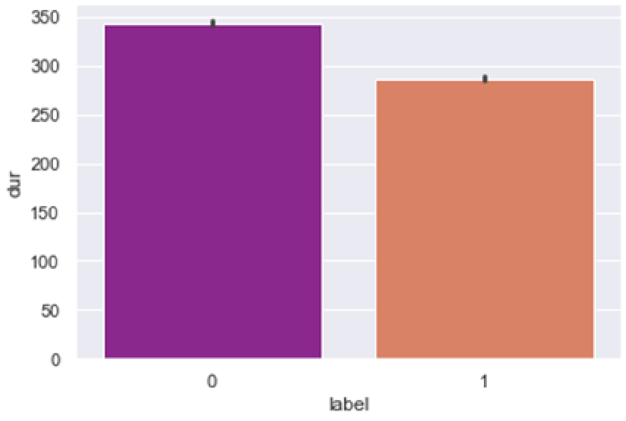


Figure 5 Duration

The duration of the flow is categorized in the above figure. This is used to statistically label stream flow. These data are mean flow daily values measured over a specified equation below. It is computed using definite time interval. The exceedance duration [24] is given as the below equation (8), where P is the probability of the trusted flow, m is the high to low ranking and n denotes mean flow [25] of the network switch.

$$P = 100 * \left(\frac{m}{n+1}\right) \tag{8}$$

4. SDN IMPLEMENTATION

Second part of implementation is carried out in SDN through importing the predefined libraries. TensorFlow implementation is as follows: TensorFlow Implementation is an open source for deploying deep learning models. Keras is used for Convolutional Neural Networks (CNN) environment that run and compile on Central Processing Units (CPU) and Graphics Processing Unit (GPU). This implementation part is classified into data plane, controller plane, and data plane.

4.1. Mininet, OpenFlow, and the Pox Controller

In our proposed system Mininet is used to create an SDN environment. It enables an SDN environment in any Pc or laptop. Software Defined Networking (SDN) provides fast prototyping. Pox controller [26] is the open flow switch that allows SDN exploration. POX controller default program is modified according to our proposed system. Through Pox custom-built controller will be created for other switch operation flow. Pox is already installed on the Mininet virtual machine.

4.2. Flowchart of Proposed Work

The flowchart of SDN implementation is shown in below Figure 6. Here, the SDN switches help in traffic management from source to destination nodes. For every new flow identified in the switch, the SDN controller will send the routing instruction to the switch and the redirection to its respective network takes place. OpenFlow will look after the interaction with the controller with the other interface.

4.2.1. Module 1: Network Creation

Mininet is used to create an SDN environment. At the first controller, switches and nodes are created. We can have any number of switches while running the host. We have taken the example of 64 nodes in that 9 switches, 1 controller then created data plane, controller plane and management plane. The switch upholds certain data correlated to the traffic flow and might make available the particular data to the controller on its demand. Attacks are launched by attacker node by the user which makes the switch to expose directly to the switch.

4.2.2. Module 2: Packet Transmission

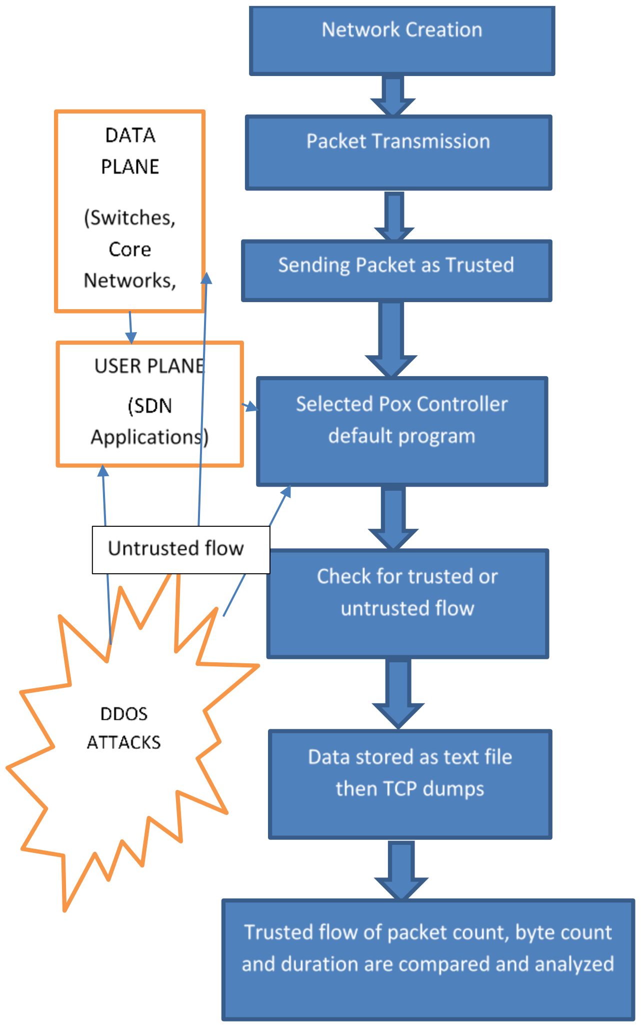

Network Creation
Packet Transmission
Sending Packet as Trusted
Selected Pox Controller default program
Check for trusted or untrusted flow
Data stored as text file then TCP dumps
Trusted flow of packet count, byte count and duration are compared and analyzed
DATA PLANE
(Switches, Core Networks,
USER PLANE
(SDN Applications)
Untrusted flow
DDOS ATTACKS

Figure 6 Flowchart of SDN Implementation

In this module, packet transmission is carried out by sending the packet as the trusted switch. By default, we will send the packet as trusted flow for the features found in the analysis part of section 3. Packet count, byte count and the duration flow are the three features used for the identification of trusted switch. Packet transmission are measured based on the number of packets per second. The trusted and malicious flow are evolved in packet count.

4.2.3. Module 3: Controller

Here, we used POX controller default program. This program is modified according to our proposed implementation. With the features made through the analysis part we have set them alone inbuilt with the controller. Through these specified switches with the specified condition alone the switch will be allowed.

SDN controller communicates with the data plane and user plane for detecting the vulnerabilities in the switches sent from the transmission. SDN controller with the help of pox controller will detect the attacker node and makes the interruption of the switch to travel further.

4.2.4. Module 4: Trusted Flow

By default, the switch will be trusted through the controller default program. For those switches and for that specified conditions alone the flow will be considered as trusted. If the condition is not satisfied, then the flow will be disconnected. Finally, the normal packet will be sent will be reached successfully. The trusted flow of the switch will be detected for further feature classification. If the attacker node is sent, the flow will be interrupted.

4.2.5. Module 5: Untrusted Flow

For the untrusted flow, we have a file attacker. When sending them from the host, the switch will be interlinked. The flow will be disconnected if the flow will be untrusted and it will not evaluate the features found from the analysis part. The malicious flow of the network switch looks for vulnerabilities and security in this module using the SDN controller. When a DDoS attack happens, the network experiences a packet burst. Apart from the number of packets, only DDoS has two patterns: one is a large number of packets from several sources sent to a single target, and the other is a rapid start. Unlike a hot subject, which takes time to split and generally spreads exponentially, a DDoS assault grows like a spark, unless the perpetrator pretends to be someone else. Another difference between a hot subject and DDoS is that a hot topic is generally only seen once, but DDoS requires each bot to visit the target repeatedly to enhance its efficacy and power. According to the above-mentioned matching problem, SDN environments will have two additional side effects.

4.2.5.1. Effect on Switches

Assume that a DDoS attack happens over the whole network, with bots from the attacker's "botnet" separating in each switch. If no corresponding flow table rule exists, the switch's storage capacity for uninstructed packets will be depleted, and the switch will have to delete old or new packets when a new uninformed packet attains. Another issue is that if the controller does not properly manage the flow table, each independent packet with a changed destination and source, has its own flow table, the flow table space would rapidly run out of capacity.

4.2.5.2. Effect on Controller

When a DDoS attack occurs, a large number of uninformed packets are sent through various switches, awaiting the controller's alert. This rapidly exhausts the controller's processing capacity, time out and, causes latency, resulting in packet loss or the controller's complete shutdown, rendering the network inoperable. Higher specification controller and switch devices with more capacity and faster processing time permit the SDN setting to shift more packets, however, this is insufficient to solve the problem. Other methods to exacerbate these negative effects are also available.

- A low volume of traffic. According to [26], regardless of how busy a new stream's traffic is, only the primary packets of the flow are wrapped in packet-in messages and forwarded to the controller. As a result, attackers will favor targeting low-traffic flows in order to have a greater impact on the controller.
- There is a lot of traffic. On the divergent, we may employ high traffic, with each packet containing worthless data, to consume the greatest amount of switch space.

4.2.6. Module 6: Storing the Flow

Both the trusted and untrusted flow of switches will be a text file. Then those files will be saved as Transmission Control Protocol (TCP) dumps. If trusted flow means the packet will reach the switch and the data will be stored as text file. Here, we consider the trusted flow as the efficient network switch where the data are saved as text file.

4.3. Algorithm of the Proposed Work

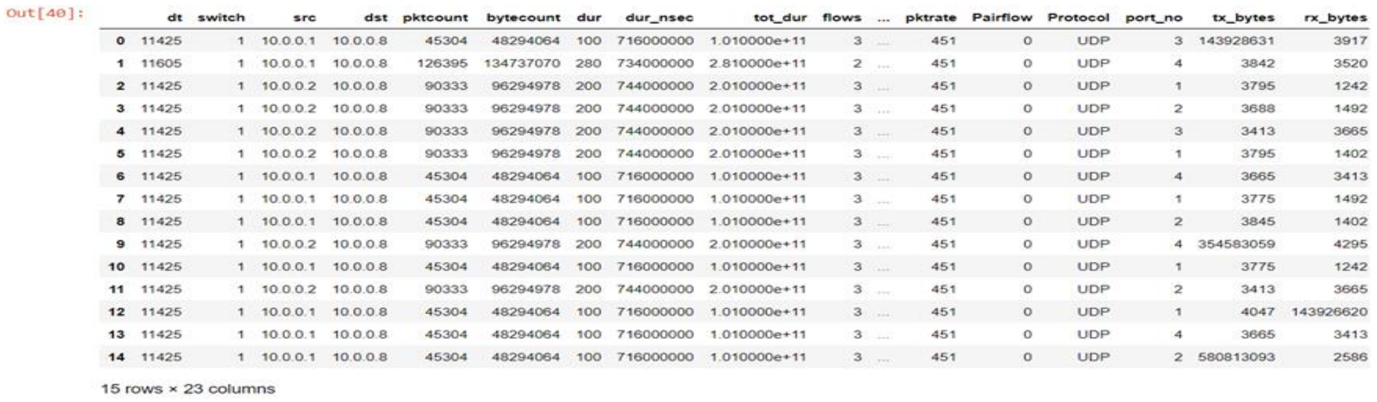
The computation flow of feature classification of trusted flow is listed in the below Algorithm 1.

```
1 **Start** by network creation
2 **Initialize n** number of nodes in which some are switches and some are controllers.
3 **Next** create data plane, controller plane and management plane.
4 Packet are analyze and **compute** three features Byte count, Packet count and Duration for feature classification.
5 **Then** flow of duration starts, **if** the duration is high then it is untrusted flow (0) **else** it is trusted flow (1).
6 The untrusted flow due to DDoS attacks on switches and controllers may interpret the flow of switches
7 The exceedance duration is given as $\boldsymbol{P = 100} * (\frac{\boldsymbol{m}}{\boldsymbol{n+1}})$ ,Where P is the probability of the trusted flow, m is the high to low ranking and n denotes mean flow of the network switch.
8 Firstly, sending packets as trusted using default Pox controller, if trusted the flow continues and stores as TCP dumps and text file else the flow gets interrupted.
9 **End**
```

Algorithm 1 Computing Feature Classification for Trusted Flow

5. PERFORMANCE EVALUATION

The performance of trusted flow and untrusted flow with the evaluation of three features are experimentally proved in the below figures. Figure 7 shows the Kaggle dataset for the SDN trusted and untrusted switches. In our work, we address the malicious switches and the trusted flow of switches from the network node are classified using the pox controller in Software Defined Networking (SDN).



```
Out[40]:
       dt  switch       src       dst  pktcount  bytecount  dur   dur_nsec       tot_dur  flows  ...  pktrate  Pairflow  Protocol  port_no    tx_bytes   rx_bytes
 0  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        3  143928631       3917
 1  11605       1  10.0.0.1  10.0.0.8    126395  134737070  280  734000000  2.810000e+11      2  ...      451         0       UDP        4       3842       3520
 2  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        1       3795       1242
 3  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        2       3688       1492
 4  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        3       3413       3665
 5  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        1       3795       1402
 6  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        4       3665       3413
 7  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        1       3775       1492
 8  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        2       3845       1402
 9  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        4  354583059       4295
10  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        1       3775       1242
11  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  2.010000e+11      3  ...      451         0       UDP        2       3413       3665
12  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        1       4047  143926620
13  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        4       3665       3413
14  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  1.010000e+11      3  ...      451         0       UDP        2  580813093       2586

15 rows × 23 columns
```

Figure 7 Dataset Description

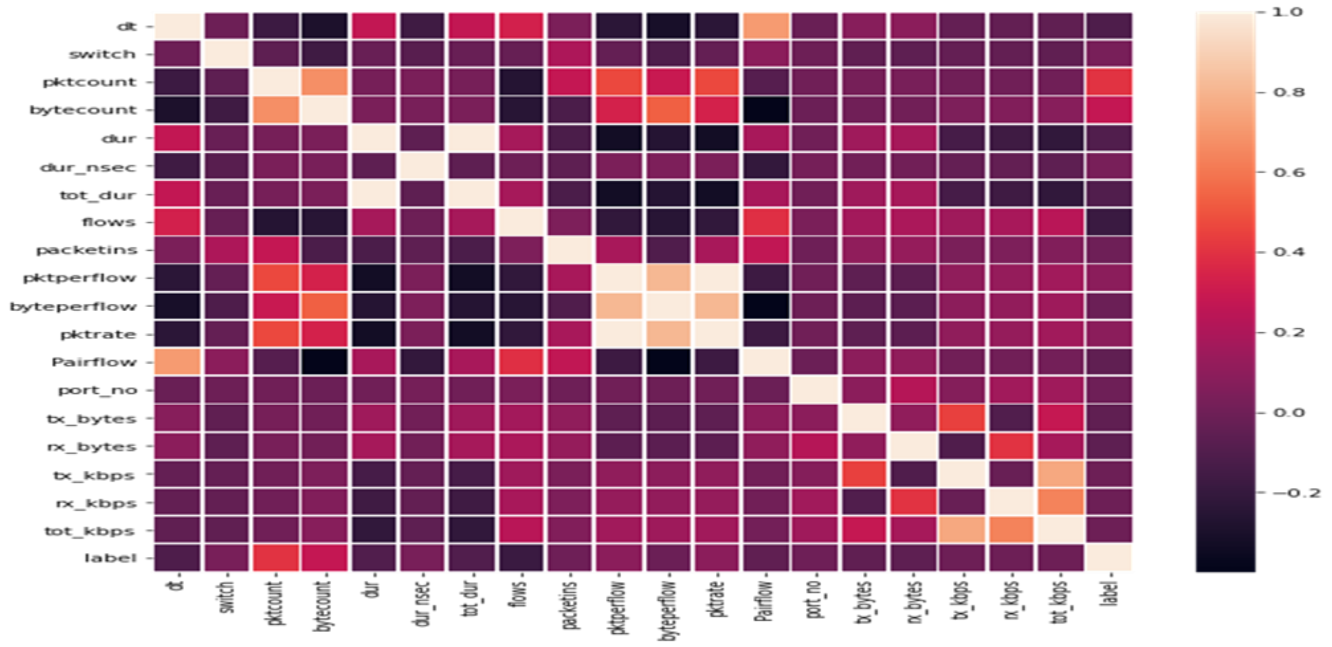
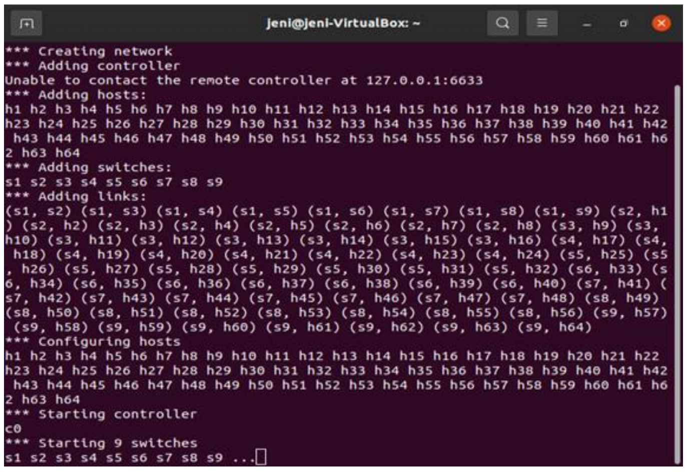


Figure 8 Heat Map

Figure 8 describes the heat map which is the packet visualization method where the features are mentioned in different colours. To give detailed pictorial of the flow of trusted and untrusted switch this heat map is used. Heat map provides the labels mentioned for vertical and horizontal axis. Figure 9 shows the creation of networks with the needed host, controller, links and switches.

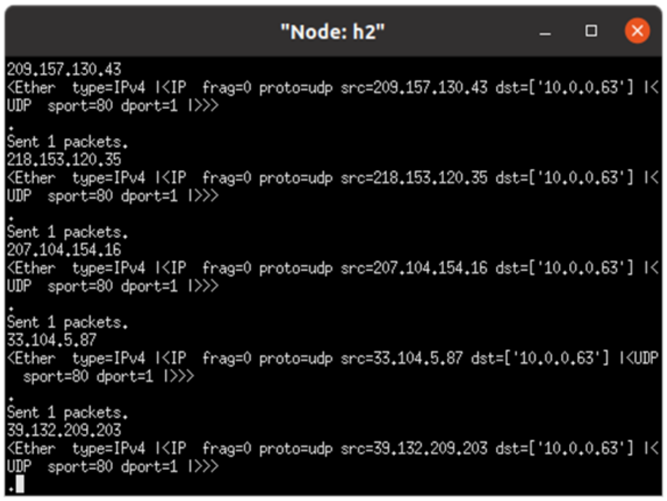
Figure 10 shows the attacker node of the switch flow. Here shows the Flow control of S2 since h2 is connected to s2.



```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h6
2 h63 h64
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8 s9
*** Adding links:
(s1, s2) (s1, s3) (s1, s4) (s1, s5) (s1, s6) (s1, s7) (s1, s8) (s1, s9) (s2, h1
) (s2, h2) (s2, h3) (s2, h4) (s2, h5) (s2, h6) (s2, h7) (s2, h8) (s3, h9) (s3,
h10) (s3, h11) (s3, h12) (s3, h13) (s3, h14) (s3, h15) (s3, h16) (s4, h17) (s4,
 h18) (s4, h19) (s4, h20) (s4, h21) (s4, h22) (s4, h23) (s4, h24) (s5, h25) (s5
, h26) (s5, h27) (s5, h28) (s5, h29) (s5, h30) (s5, h31) (s5, h32) (s6, h33) (s
6, h34) (s6, h35) (s6, h36) (s6, h37) (s6, h38) (s6, h39) (s6, h40) (s7, h41) (
s7, h42) (s7, h43) (s7, h44) (s7, h45) (s7, h46) (s7, h47) (s7, h48) (s8, h49)
(s8, h50) (s8, h51) (s8, h52) (s8, h53) (s8, h54) (s8, h55) (s8, h56) (s9, h57)
 (s9, h58) (s9, h59) (s9, h60) (s9, h61) (s9, h62) (s9, h63) (s9, h64)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h6
2 h63 h64
*** Starting controller
c0
*** Starting 9 switches
s1 s2 s3 s4 s5 s6 s7 s8 s9 ...
```

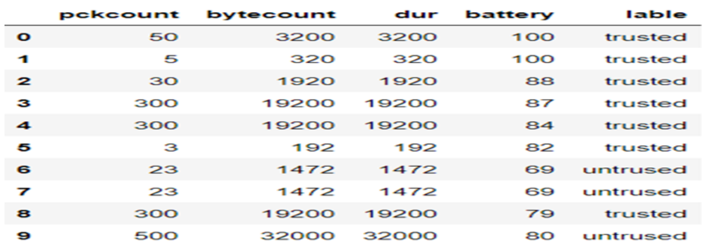
Figure 9 Creation of Network

The total count of trusted and untrusted flow is mentioned in Figure 11. The total number of trusted flows is 7 and untrusted flow is 3.



```
209.157.130.43
<Ether  type=IPv4 |<IP  frag=0 proto=udp src=209.157.130.43 dst=['10.0.0.63'] |<
UDP  sport=80 dport=1 |>>>
.
Sent 1 packets.
218.153.120.35
<Ether  type=IPv4 |<IP  frag=0 proto=udp src=218.153.120.35 dst=['10.0.0.63'] |<
UDP  sport=80 dport=1 |>>>
.
Sent 1 packets.
207.104.154.16
<Ether  type=IPv4 |<IP  frag=0 proto=udp src=207.104.154.16 dst=['10.0.0.63'] |<
UDP  sport=80 dport=1 |>>>
.
Sent 1 packets.
33.104.5.87
<Ether  type=IPv4 |<IP  frag=0 proto=udp src=33.104.5.87 dst=['10.0.0.63'] |<UDP
  sport=80 dport=1 |>>>
.
Sent 1 packets.
39.132.209.203
<Ether  type=IPv4 |<IP  frag=0 proto=udp src=39.132.209.203 dst=['10.0.0.63'] |<
UDP  sport=80 dport=1 |>>>
.
```

Figure 10 Attacker Node



	pckcount	bytecount	dur	battery	lable
0	50	3200	3200	100	trusted
1	5	320	320	100	trusted
2	30	1920	1920	88	trusted
3	300	19200	19200	87	trusted
4	300	19200	19200	84	trusted
5	3	192	192	82	trusted
6	23	1472	1472	69	untrused
7	23	1472	1472	69	untrused
8	300	19200	19200	79	trusted
9	500	32000	32000	80	untrused

Figure 11 Result of Trusted and Untrusted Flow

If the flow is trusted, duration of the flow denoted the time taken by the packet to reach the destination. The comparison of duration of flow between trusted (1) and untrusted (0) is shown in above Figure 12. Duration of flow is measured in terms of seconds in the y-axis. Figure 13 describes the comparison of packet count of both trusted and untrusted label. The packet count is measured for smallest sizes Before dropping date packet of data per second are measured comparing the label of trusted (1) and untrusted (0) from 0 to 500.

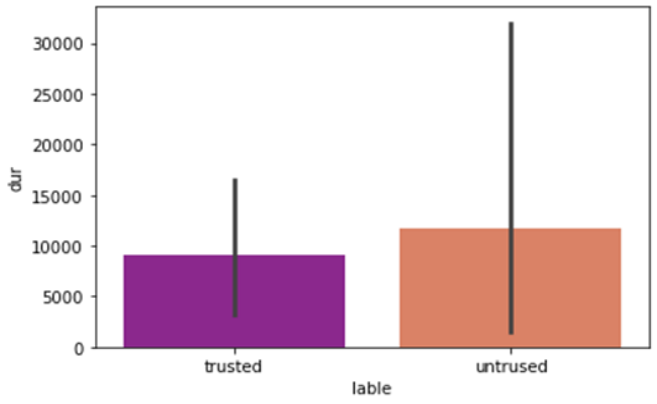


Figure 12 Duration Vs Trust

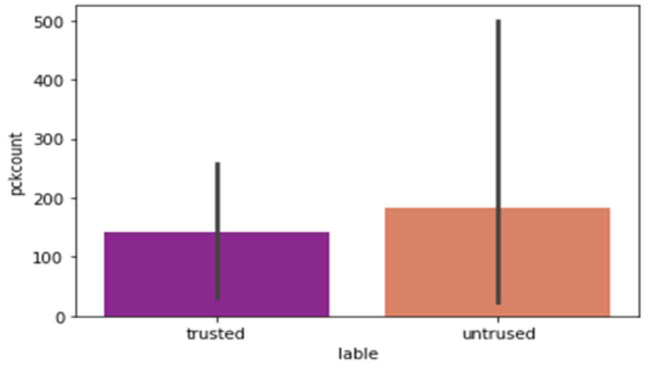


Figure 13 Packet Count vs Trust

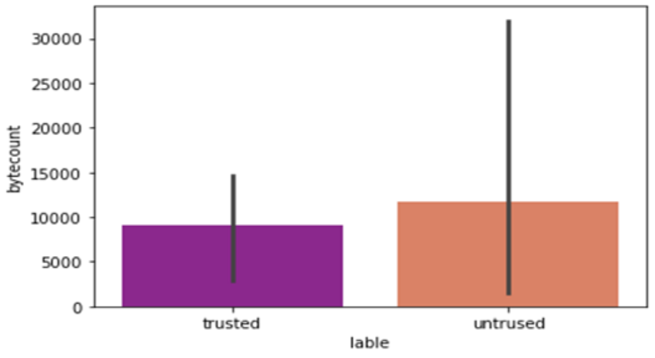


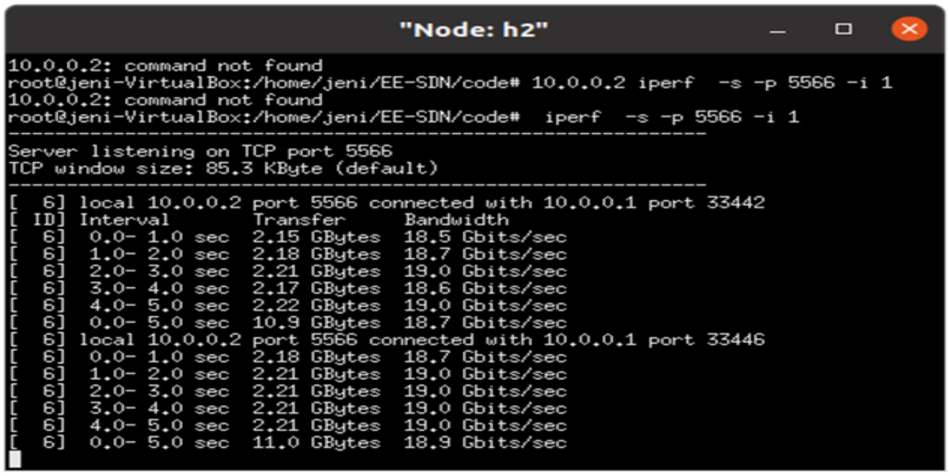
Figure 14 Byte count vs trust

The comparison of byte count for trusted and untrusted flow is mentioned in the below Figure 14. If the packet is sent normally, trusted packet reads from the S2. If we give file attacker to S2, then S2 will be considered as untrusted switch label. A byte is order of 8 bits and the label is classified as trusted (1) and untrusted (0). The below comparison denotes the byte count from 0 to 30000 with the trusted or trusted flow. Byte count is measured in terms of total number of bits per seconds that are determined without dropping data.

Throughput is the rate or rate production deals with the data sent through network to process the way of nodes travelling. Computation of throughput shown in Table 2 are carried out from host to host, host to switch and host to switch are mentioned in the Figure 15. Bandwidth of host measured in terms of kb/s. Throughput slot is measured in terms of time interval.

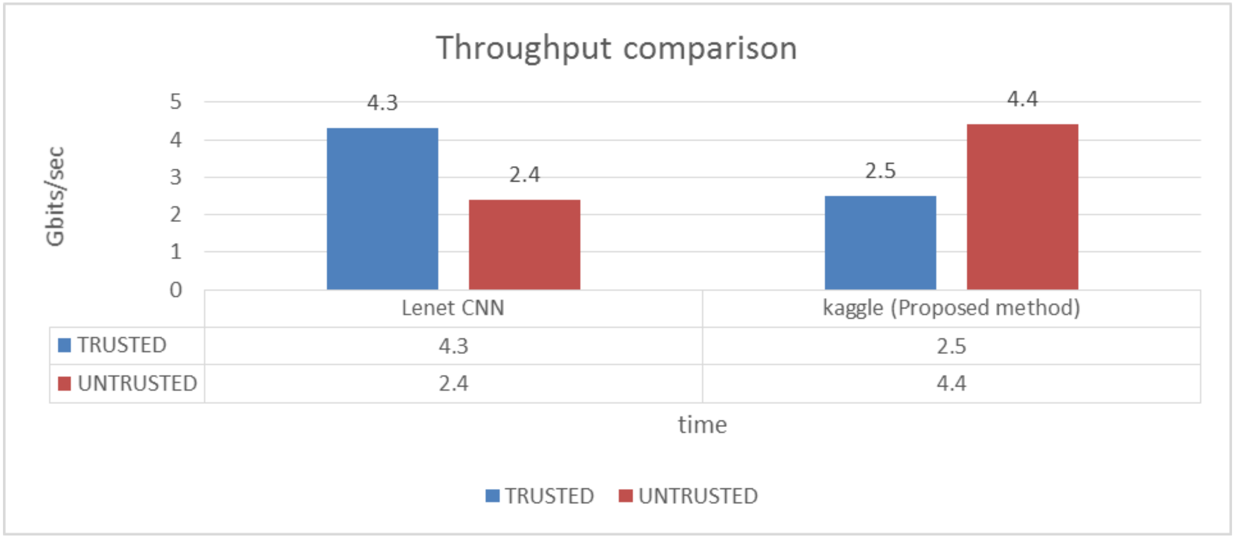
Host to Host	Host to Switch	Switch to Controller	No of Host	No of Switches
18.8	27.6	27.8	64	10

Table 2 Computing Throughput



```
10.0.0.2: command not found
root@jeni-VirtualBox:/home/jeni/EE-SDN/code# 10.0.0.2 iperf  -s -p 5566 -i 1
10.0.0.2: command not found
root@jeni-VirtualBox:/home/jeni/EE-SDN/code#  iperf  -s -p 5566 -i 1
------------------------------------------------------------
Server listening on TCP port 5566
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.2 port 5566 connected with 10.0.0.1 port 33442
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0- 1.0 sec  2.15 GBytes  18.5 Gbits/sec
[  6]  1.0- 2.0 sec  2.18 GBytes  18.7 Gbits/sec
[  6]  2.0- 3.0 sec  2.21 GBytes  19.0 Gbits/sec
[  6]  3.0- 4.0 sec  2.17 GBytes  18.6 Gbits/sec
[  6]  4.0- 5.0 sec  2.22 GBytes  19.0 Gbits/sec
[  6]  0.0- 5.0 sec  10.9 GBytes  18.7 Gbits/sec
[  6] local 10.0.0.2 port 5566 connected with 10.0.0.1 port 33446
[  6]  0.0- 1.0 sec  2.18 GBytes  18.7 Gbits/sec
[  6]  1.0- 2.0 sec  2.21 GBytes  19.0 Gbits/sec
[  6]  2.0- 3.0 sec  2.21 GBytes  19.0 Gbits/sec
[  6]  3.0- 4.0 sec  2.21 GBytes  19.0 Gbits/sec
[  6]  4.0- 5.0 sec  2.21 GBytes  19.0 Gbits/sec
[  6]  0.0- 5.0 sec  11.0 GBytes  18.9 Gbits/sec
```

Figure 15 Throughput in a Host



	Lenet CNN	kaggle (Proposed method)
🟦 TRUSTED	4.3	2.5
🟥 UNTRUSTED	2.4	4.4

Figure 16 Throughput Comparison

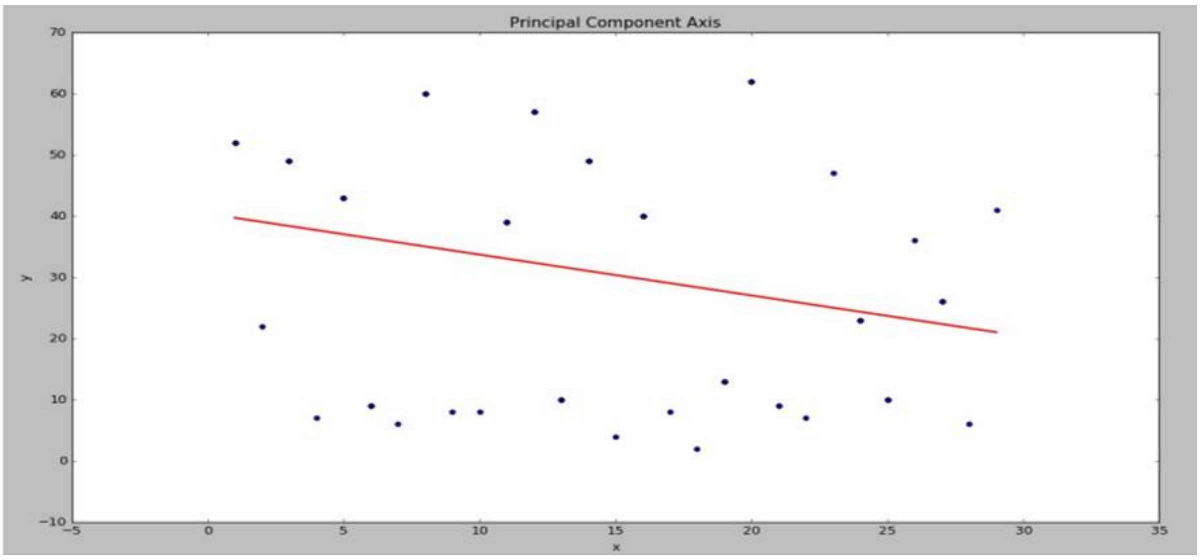


Figure 17 Principal Component Analysis with Normal Flow

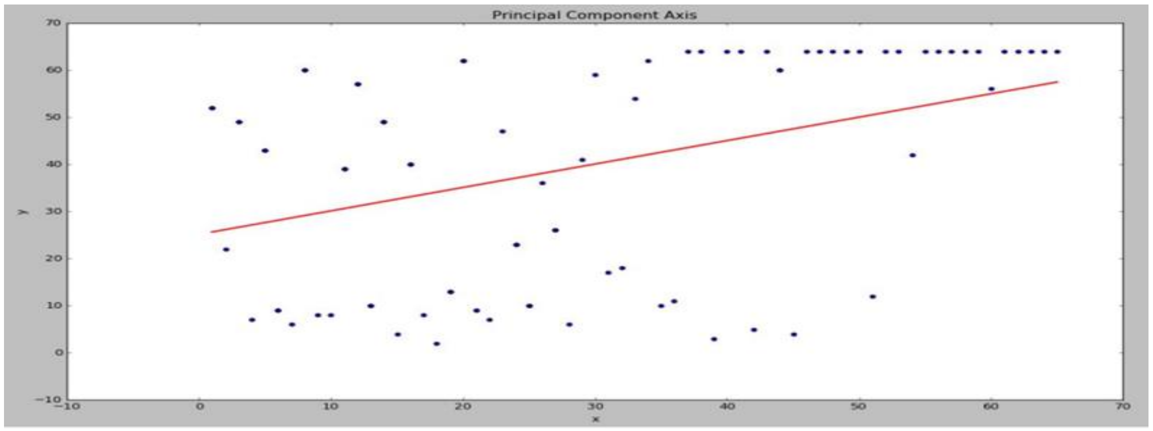


Figure 18 Principal Component Analysis During Attack

No of packets sent	Received Packets	No of packet dropped	Loss (%)	No of switches	Accuracy	Time(ms)
200	180	20	0	2	86%	1022541
600	587	13	2.6	3	87%	1307431
1200	904	296	29.6	5	82%	100600
1600	1400	200	19.886	7	87%	1505892
2100	2066	34	17	9	85%	1670831

Table 3: Results from the Implementation on Different no of Packets

Throughput comparison is computed for already proposed LeNet[4] architecture with the proposed kaggle dataset in terms of time are shown in the below Figure 16.

PCA is a technique for highlighting variance and highlighting significant outlines in a dataset. It's frequently used to make data exploration and visualization simple. Consider a dataset with only two dimensions, such as (x is packets sent, y is time taken). This dataset may be shown as a series of arguments in a plane. If we wish to isolate variation, PCA creates a new coordinates system with new (x', y') values for each point. The axes are "primary components," which are groupings of x and y that are chosen to provide one axis a lot of variances. Figure 17 depicts the PCA normal flow without DDos attacks.

Figure 18 illustrates the PCA with DDos attacks. The scatter plots denote the traffic in attacked flow of switches. X axis is packets sent and y axis is time interval.

Table 3 lists the total number of packets received, number of packets losses, percentage of loss, number of switches, accuracy percentage and the time interval in milliseconds.

6. CONCLUSION AND FUTURE ENHANCEMENTS

In this proposed system, we have experimentally proved that the flow of switch will be trusted using the features of packet count, byte count duration, and accuracy of 89 %. Packet transmission towards the switch flow is illustrated and attack detection for DDos is determined using SDN. The comparative study of the feature classification and constant throughput proves that the flow is the trusted one through deep learning model. Based on these features trusted and untrusted switch is found. The future enhancements can be focused on other features of dataset of SDN which can transmit the network switch for the energy efficiency and badge power.

REFERENCES

[1] Oliveira, T.F.; Xavier-de-Souza, S.; Silveira, L.F. Improving Energy Efficiency on SDN Control-Plane Using Multi-Core Controllers. Energies, 14, 3161, 2021.
[2] Mohsin Masood, Mohamed Mostafa Fouad, Saleh Seyedzadeh and Ivan Glesk, "Energy Efficient Software Defined Networking Algorithm for Wireless Sensor Networks", 13th International Scientific Conference on Sustainable, Modern and Safe Transport, 2019.
[3] Madhukrishna Priyadarsini, Padmalochan Bera, and Mohammad Ashiqur Rahman, "A New Approach for Energy Efficiency in Software Defined Network", Fifth International Conference on Software Defined Systems (SDS), 2018.
[4] Thangaraj Ethilu, Abirami Sathappan, Paul Rodrigues, "Modified Deep Learning Methodology Based Malicious Intrusion Detection System in Software Defined Networking", International Journal of Computer Networks and Applications (IJCNA), 8(4), PP: 381-389, 2021, DOI: 10.22247/ijcna/2021/209704.
[5] Danda B. Rawat and Swetha R. Reddy, Software Defined Networking Architecture, Security and Energy Efficiency: A Survey,IEEE communication surveys and tutorials,2019.
[6] W. Meng, W. Li, Y. Xiang and K.-K.R. Choo. A Bayesian Inference-based Detection Mechanism to Defend Medical Smartphone Networks Against Insider Attacks. Journal of Network and Computer Applications, vol. 78, pp. 162-169, Elsevier, 2017.
[7] Rinki Gupta, Sreeraman Rajan, "Comparative Analysis of Convolution Neural Network Models for Continuous Indian Sign Language Classification", Procedia Computer Science 171 (2020) 1542–1550.
[8] P. -W. Chi, M. -H. Wang and Y. Zheng, "SandboxNet: An Online Malicious SDN Application Detection Framework for SDN Networking," 2020 International Computer Symposium (ICS), 2020, pp. 397-402.
[9] Sebbar, A., ZKIK, K., Baddi, Y. MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context. J Ambient Intell Human Comput 11, 5875–5894 (2020).
[10] Nife, F.N., Kotulski, Z. Application-Aware Firewall Mechanism for Software Defined Networks. J Netw Syst Manage 28, 605–626 (2020).
[11] Neu C. V., Tatsch C. G., Lunardi R. C., Michelin R. A., Orozco A. M. S.,and Zorzo A. F.: Lightweight IPS for port scan in OpenFlow SDN networks. In NOMS 2018 IEEE/IFIP Network Operations and Manag. Symposium, Taipei, Taiwan, pp. 1–6, (2018).
[12] H. Naeem, B. Guo, and M. R. Naeem, ''A light-weight malware static visual analysis for IoT infrastructure,'' in Proc. Int. Conf. Artif. Intell. Big Data (ICAIBD), May 2018, pp. 240–244.

RESEARCH ARTICLE

[13] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, ''Classification of ransomware families with machine learning based on N-Gram of opcodes,'' Future Gener. Comput. Syst., vol. 90, pp. 211–221, Jan. 2019.
[14] A. Khalilian, A. Nourazar, M. Vahidi-Asl, and H. Haghighi, ''G3MD: Mining frequent opcode sub-graphs for metamorphic malware detection of existing families,'' Expert Syst. Appl., vol. 112, pp. 15–33, Dec. 2018.
[15] Y.-S. Liu, Y.-K. Lai, Z.-H. Wang, and H.-B. Yan, ''A new learning approach to malware classification using discriminative feature extraction,'' IEEE Access, vol. 7, pp. 13015–13023, 2019.
[16] Chang Y., and Lin T.: Cloud-clustered firewall with distributed SDN devices. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, pp. 1–5. (2018).
[17] J. Yan, Y. Qi, and Q. Rao, ''Detecting malware with an ensemble method based on deep neural network,'' Secur. Commun. Netw., vol. 2018, pp. 1–16, Mar. 2018.
[18] D. Gibert, C. Mateu, J. Planes, and R. Vicens, ''Classification of malware by using structural entropy on convolutional neural networks,'' in Proc. 32nd AAAI Conf. Artif. Intell., (AAAI), 30th Innov. Appl. Artif. Intell. (IAAI), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI), New Orleans, LA, USA, 2018, pp. 7759–7764.
[19] Z. Ma, L. Liu, W. Meng. Towards Multiple-Mix-Attack Detection via Consensus-based Trust Management in IoT Networks. Computers & Security, In press (2020).
[20] Y. Meng. The practice on using machine learning for network anomaly intrusion detection. The 2011 International Conference on Machine Learning and Cybernetics (ICMLC 2011), IEEE, pp. 576-581, 2011.
[21] Andrzej Kamisiński, Carol Fung," FlowMon: Detecting Malicious Switches in Software-Defined Networks", ACM CCS workshop on Automated Decision Making for Active Cyber Defense ,2015.
[22] Lis, A.; Sudolska, A.; Pietryka, I.; Kozakiewicz, A. Cloud Computing and Energy Efficiency: Mapping the Thematic Structure of Research. Energies 2020, 13, 4117.
[23] Aujla, G.S.; Kumar, N.; Zomaya, A.Y.; Ranjan, R. Optimal Decision Making for Big Data Processing at Edge-Cloud Environment: An SDN Perspective. IEEE Trans. Ind. Inform. 2018, 14, 778–789.
[24] Xu, G.; Dai, B.; Huang, B.; Yang, J.; Wen, S. Bandwidth-aware energy efficient flow scheduling with SDN in data center networks. Future Gener. Comput. Syst. 2017, 68, 163–174.
[25] Fernández-Fernández, A.; Cervelló-Pastor, C.; Ochoa-Aday, L. Energy Efficiency and Network Performance: A Reality Check in SDN-Based 5G Systems. Energies 2017.
[26] Son, J.; Dastjerdi, A.V.; Calheiros, R.N.; Buyya, R. SLA-Aware and Energy-Efficient Dynamic Overbooking in SDN-Based Cloud Data Centers. IEEE Trans. Sustain. Comput. 2017, 2, 76–89.

Authors

Mr. Thangaraj Ethilu, received the M. Tech in Computer Science and Engineering at Dr. M.G.R Educational and Research Institute University in Chennai and B. E degree in Computer Science and Engineering at Madurai Kamaraj University in Madurai. Presently he is pursuing the Ph.D. in Department of Computer Science and Engineering, Annamalai University Chidambaram, Tamil Nadu India. His area of interest is networking and cloud computing.

Dr.S. Abirami Sathappan, received the M. E in Computer Science and Engineering at Annamalai University in Chidambaram and Ph.D. in Computer Science and Engineering at Annamalai University in Chidambaram. Presently she is working as an Assistant Professor in Computer Science and Engineering, Annamalai University Chidambaram, Tamil Nadu India. Her area of interest is Image Processing.

Dr. Paul Rodrigues received the M. E in Computer Science and Engineering at Mothilal Nehru Regional Engineering College in Allahabad and Ph.D. in Computer Science and Engineering at Pondicherry University in Pondicherry. Presently he is working as a Professor in Computer Science and Engineering, King Khalid University, Abha, Saudi Arabia. His area of interest is networking.

How to cite this article:

Thangaraj Ethilu, Abirami Sathappan, Paul Rodrigues, "Improving Performance and Efficiency of Software-Defined Networking by Identifying Malicious Switches through Deep Learning Model", International Journal of Computer Networks and Applications (IJCNA), 9(1), PP: 72-83, 2022, DOI: 10.22247/ijcna/2022/211627.