IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 April 2022, accepted 12 June 2022, date of publication 21 June 2022, date of current version 27 June 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3184722*

# Overhead Reduction Technique for Software-Defined Network Based Intrusion Detection Systems

**AHMED H. JANABI[1,2], (Member, IEEE), TRIANTAFYLLOS KANAKIS[1], (Member, IEEE), AND MARK JOHNSON[1]**

[1]Department of Computing, University of Northampton, Northampton NN1 5PH, U.K.
[2]Department of Air Conditioning and Refrigeration, Al-Mustaqbal University College, Babylon 51001, Iraq

Corresponding author: Ahmed H. Janabi (ahmed.janabi@northampton.ac.uk)

**ABSTRACT** In Software-Defined Networks, the Intrusion Detection System is receiving growing attention, due to the expansion of the internet and cloud storage. This system is vital for institutions that use cloud services and have many users. Although the Intrusion Detection System offers several security features, its performance is lagging behind in large enterprise networks. Existing approaches are based on centralised processing and use many features to implement a protection system. Therefore, system overload and poor performance occur on the controller and OpenFlow switches. As a result, the current solutions create issues that must be considered, especially when they are implemented on large networks. Furthermore, enhancements in security applications improve the reliability of networks. Following a literature review of the existing Intrusion Detection Systems, this paper presents a new model that offers decentralised processing and exchanges data over an independent channel, in order to solve issues relating to system overload and poor performance. Our model utilises an appropriate feature selection method to reduce the number of extracted features and minimise the data transmitted over the channels. Additionally, the Naive Bayes algorithm has been employed for flow classification purposes, since it is a fast classifier. We successfully implemented our framework, using the Mininet emulator, which provides a suitable networking environment. Evaluations indicate that our proposed system can detect various attacks with an accuracy of 98.46% and nominal decreasing rates of 1.5% in throughput and 0.7% in latency analyses, when the model is implemented in wide range networks.

**INDEX TERMS** Naive Bayes—protection system based distribution process (NB-PSDP), Naive Bayes (NB), software-defined network (SDN), intrusion detection system (IDS), machine learning (ML), distribution process, CSE-CIC-IDS2018 dataset.

## I. INTRODUCTION

The world is moving toward virtualising data by utilising cloud computing facilities. Therefore, it is necessary to protect sensitive data from intruders in these platforms. Existing Intrusion Detection Systems (IDS) need to be flexible to cope with big data and the rapid development of networks. Technological expansion and fast data transmission have led to a growing number of perilous attacks on institutions. Therefore, it is essential to develop efficient IDSs in line with the proliferation of new technology. An IDS plays a critical role in network security as it enables the observation of the traffic flow across the network and predicts abnormal activity by investigating new incoming flows/packets individually, or within fixed time windows.

Conventional network architectures have failed to meet the market requirements for rapid technological growth. The Software-Defined Network (SDN) is a new network architecture that can deal with most market necessities, due to it is flexibility [1]. The central concept of SDN is based on the detachment of the control and data planes. In other words, the data plane simply forwards incoming data to the destination, while the control plane manages the network. Therefore, the data plane does not involve networking configurations, but consists of forwarding devices, such as hubs, switches, and routers, managed by a controller in the control plane. These

The associate editor coordinating the review of this manuscript and approving it for publication was Tiago Cruz.

planes exchange data by using the OpenFlow protocol. The control plane is connected to the data plane via a south-bound interface, using the OpenFlow channel [1], as shown in Fig.1. The control plane has absolute authority to monitor and configure the network over a centralised authentication process.

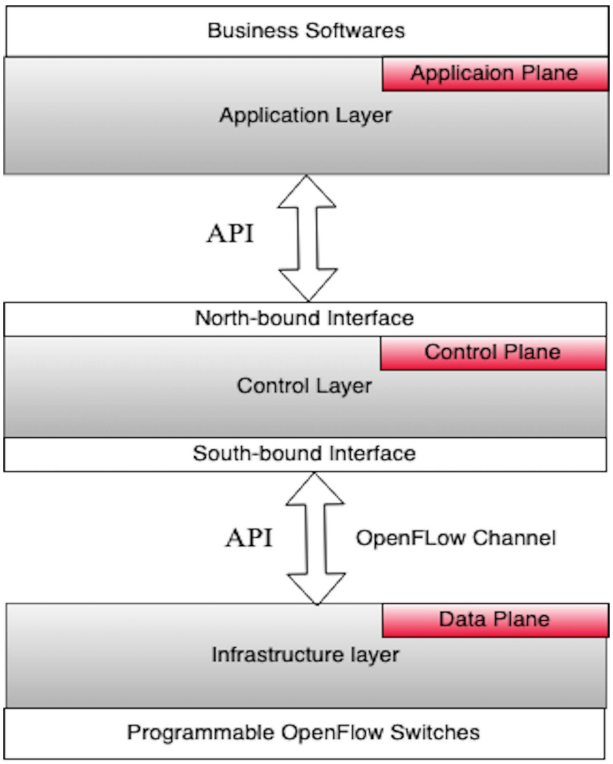


**FIGURE 1.** Standard SDN Architecture.

SDN is widely considered as one of the next-generation network architectures, and it has serious security challenges that enable attackers to target various vulnerable points. Traditional IDSs are inappropriate for the SDN context [2]. Furthermore, state-of-art IDSs do not satisfy the businesses requirements. Most of the solutions were developed for small enterprise networks, and some researchers have been breaking the SDN concept. Therefore, it is essential to develop a modern IDS that handles security issues in an SDN framework and is suitable for large-scale networks. IDS is categorised into two types: signature-based and anomaly-based [3]. Signature-based IDSs identifies as misuse detection or knowledge-based detection. It detects attacks by utilising specific patterns and compares the observed behaviour with those stored as a unique signature in a database. Such IDSs have high speed and low scalability [1]. On the other hand, anomaly-based IDS monitors network traffic activity by employing statistical analysis methods. This method collects specific data of the regular user's behaviour over time. It is utilising the Machine Learning algorithms for classification purposes. At the same time, it is characterised by low speed and high scalability.

The principal contributions of this article are as follows:

- It presents a concise literature review of IDSs in SDN.
- Develops a novel IDS called Naive Bayes – Protection System, based Distribution Process (NB-PSDP). The proposed model distributes the processing steps across the OpenFlow switch and SDN controller. Our system is the first security model that spreads the processing steps between the SDN planes.
- It utilises a new independent channel, called the IDS channel, for signalling purposes in order to avoid the overload in the OpenFlow channel.
- The proposed approach reduces the signalling overhead by minimising the transmission of messages between the IDS and OpenFlow switches.
- The model is the first IDS-based SDN that employs a univariate feature selection method on the CSE-CIC-IDS2018 dataset to reduce the number of features extracted.
- It enhances the protection of SDN networks from several types of attack, including document infiltration, SSH-Bruteforce, Brute Force-XSS, and DoS attacks-Slowloris.
- The experiments indicate that NB-PSDP demonstrates high throughput at a low delay rate and less CPU usage compared with state-of-the-art IDSs.
- The NB-PSDP achieves high prediction accuracy, while implementing a minimum number of features.

The rest of this article is organised as follows: Section two introduces a background on IDS. Section three defines the proposed system and its benefits. Section four discusses the evaluation, experimental setup, and results. Lastly, Section five combines the conclusion of the paper and potential future works.

## II. LITERATURE REVIEW

### A. WHAT IS THE INTRUSION DETECTION SYSTEM (IDS)?

The IDS defences a network by controlling and monitoring its traffic. The IDS is essential, particularly when a network enterprise has security concerns or sensitive data [4]. The IDS releases an alert to the administrator, when malicious traffic has been identified, and redirects or filters this traffic, according to its requirements and policies [5]. In a network-based IDS, the typical components are the IDS management, IDS collector, and IDS classifier. The IDS examines every packet that arrives through the switches using a south-bound API. This process is performed by using a standard switch managed by the controller, so the controller can see and analyse the network traffic moving over the switches [6].

### B. TYPES OF IDS

Based on its targets, IDS can be categorised as follows:

1) **Host-based IDS:** this installs on devices and examines all the activity that occurs on a computer [7].
2) **Network-based IDS:** this system observes the traffic to detect malicious activity by checking the traffic

behaviour at various stages and raises an alarm when it identifies an anomaly [7].

### C. POTENTIAL ATTACKS IN THE SDN

In SDNs, the most concerning attacks are Distributed Denial of Service attacks (DDoS), Denial of Service attacks (DoS), Root to Local (R2L), User to Root (U2R), and Prob attacks. However, there are other threats, and their possible effects on SDNs are discussed below:

1) **Saturation of the controller resource:** the controller is considered the core of the SDN network. Therefore, if the controller crashes, it can affect the network performance. Controller can be exhausted by processing a high volume of flooded requests by an attacker. During controller overloading, it is impossible to handle all incoming flows, which causes a high volume of regular traffic delays, or lapses in the essential processing [8].
2) **Switch overloading:** DoS and DDoS attacks are the main threats to SDN architecture. Such attacks can generate many malignant packets, in order to flood switches. When the switch cannot find an entry that matches a malicious packet in the flow table, it forwards it to the controller to apply specific rules. However, because the switch is limited by the TCAM, not all incoming packets can be processed. Therefore, the incoming flows and requests exceed the flow table's memory. Consequently, regular traffic will not be subjected to this necessary process [7].
3) **Congestion bandwidth between switch and controller:** missing events occur because of two actions during the new incoming packets. The first occurs when incoming packets are stored in the buffer of a flow table. The second occurs when an OpenFlow request has been created, containing an ID with information stored in the packet header. Thus, packets can collide with another bound interface, causing the blocking of services [9].

### D. VULNERABLE POINTS IN SDN

The SDN consists of three planes: data, control, and application, as shown in Fig.1. This nature of SDN makes the network vulnerable to several attacks, such as DDoS and DoS. The points targeted by these attacks are described below:

1) **SDN switches:** the main function of the OpenFlow switch is forwarding the newly received traffic to the controller to take proper action. However, the switch has a flow table with limited memory. Therefore, there is a major security issue if the attacker forwarded a massive number of malignant packets to flood the switch [10].
2) **SDN switch links:** the packets received may transfer between the switches, before reaching the controller. Packets are not encrypted when transmitted over links. Therefore, an attacker can easily catch these packets, particularly in wireless environments [10]. This type of attack is known as man-in-middle attack. In addition, DoS and DDoS attacks can be implemented at this point, so the attacker can forward a massive amount of malignant traffic to stop the transmission of regular traffic, thereby blocking network services.
3) **SDN controller:** as it represents the core of the network, the controller takes critical actions for the environment. Any anomaly can stop network activity. The controller influences the functionality of the network, turning it into an engaging activity for attackers. However, if a single controller is used, the network will face a single point of failure. Therefore, this security concern must be addressed [11].
4) **Controller and switch communication:** when the incoming packet cannot match the flow table records, it will be redirected to the controller for in-depth processing. Consequently, the controller will insert new rules into the flow table entries of the specific switch. At this point, an attacker can intercept a packet to inject malignant code or modify the current rules. Hence, most packets take the wrong direction [7].
5) **Applications:** these applications include traffic monitoring and traffic classification. Attackers target such applications to obtain sensitive information, or to add malignant rules to the controller. An unauthenticated user can perform this process, while communicating with the API. Therefore, SDN applications are considered direct target points to block controller services [12].

### E. RELATED WORKS AND CRITICAL REVIEW

In this subsection, a critical review of existing research is presented, as is a summary in Table 1.

Jin *et al.* [13] constructed a simulation platform, based on Mininet in SDNs, to identify DDoS attacks. A SVM classifier was used to classify new incoming packets. Flow table collection was performed through characteristic value extraction to feed the classifier during the attack detection process. However, this was not performed through a feature selection process. Therefore, some features can affect the classifier's performance and accuracy, which presents a challenge related to the proper selection method. According to the results, the best accuracy rate was obtained by testing with a dataset of 600 TCP packets, achieving 96.83% accuracy. Compared to UDP and ICMP, the best results were obtained for the TCP protocol because the TCP protocol has more network fields for classification. However, UDP has more representativeness, based on the application analysed. Finally, ICMP has representative payload characteristics that allow individual behaviour during attacks. The proposed system achieved a 95.24% detection accuracy rate and 1.26% false alarm rate.

Song *et al.* [14] created an IDS framework called Eunoia. Their proposed model is an IDS ML for SDN networks. Eunoia targets malicious traffic in SDNs. Their solution consists of three sub-processes: data pre-processing, data modelling, along with decision-making and response. However, this method faces many challenges, such as having sufficient

**TABLE 1. IDSs based machine learning.**

| Ref | Approach | F No. | Attacks | Controller | Dataset | Accuracy |
|---|---|---|---|---|---|---|
| [13] | SVM | 6 | DDoS | SDN | Simulation | DR of 95.24% & FA of 1.26% |
| [14] | SVM, NB, KNN, ASVM, etc. | 5 | DDoS | POX | NSL-KDD | 81.5% AC for J48 |
| [15] | J48 | 41 | DoS, Probe, U2R,& R2L | SDN | NSL-KDD | DR of 90.9% |
| [16] | C4.5 | 52 | DoS & Prob | SDN | 1999 Darpa | P of 0.989 & R of 0.964 |
| [17] | LSTM, GRU & SVM | 89 | DDoS | SDN | Generated dataset | AC of 96.67% |
| [19] | J48-tree | 41 | Unspecified | NetFPGA10G | KDD'99 | DR of 91.81% & FA of 0.55% |
| [20] | NB, KNN, K-means & K-mediods | 52 | DDoS | POX | Privet | DR of 94%, 90%, 86% and 88% |
| [21] | KNN | 23 | DDoS | SDN | NSL-KDD | AC of 91.27% & 0.99% of P |
| [22] | K-means++, AdaBoost, & RF | 6 | DDoS, DoS, U2R, & R2L | SDN | KDD-Cup | AC of 84% |

computational power to withhold and process the enormous amount of data coming into the SDN-based network intrusion system as malicious traffic. The features extracted in the model filter valuable data from nonvaluable data. The efficient packet processing is another relevant challenge in order to avoid bottlenecks of the system.

Sathya and Thangarajan [15] focused on security violations in an SDN environment and how the model can be identified to prevent attacks using anomaly-based detection methods. The authors presented an IDS to recognise DoS, Probe, U2R, and R2L attacks. The proposed model uses the NSL-KDD dataset, which includes various types of attack. The system achieved a 90.9%, 91.1%, 80.2%, and 98.1% detection rates and 0.111%, 0.249%, 0.69%, and 0.887% false alarm rates for the DoS, Probe, R2L, and U2R, respectively. However, this system did not achieve the highest accuracy, compared to the other approaches. It failed to minimise the features selected by the Binary Bat algorithm, when a large number of selected features were used.

Le *et al.* [16] presented an IDPS network-based detection method in SDN such as DoS and Probe attacks. The proposed system used the Decision Tree approach with the 1999 Darpa dataset. The C4.5 algorithm averted overfitting data and handled the missing attribute values of the training data. They claimed that the effect of DoS and Probe attacks in SDN can be mitigated by this method. The model was evaluated based on Precision and Recall measurements. The DoS attack results were 0.989 and 0.964, respectively, whereas those for the Prob attack were 0.984 and 0.921, respectively. However, they used the 1999 Darpa dataset for training, whereas many researchers no longer used this dataset because it does not contain features related to new types of DoS attacks. The attackers always create new behaviours for DoS and Prob attacks. The integration of the SDN controller and their IDP involved a high volume of control packets to monitor traffic. Therefore, the model overloads the controller, which this is another challenge that needs to be addressed in their work.

Hüseyin *et al.* [17] applied an IDS to identify DDoS in SDN networks. The main contribution of this research is to merge two DL approaches in parallel for traffic classification and SVM for feature extraction. The DL algorithms of LSTM and GRU have been utilised in the proposed system. The model was validated with SCADA topology. The experiments showed that the proposed model improved the performance by 5%. Moreover, the system was evaluated using accuracy measurement, and the result was 96.67%. However, the detection system needs to be extended to identify several attacks, not only DDoS attacks. The researchers employed only 420 non-attack and 3780 attack examples in the learning phase, which were considered too low for good training. Therefore, the obtained accuracy does not reflect the actual rate. Furthermore, the framework involves high complexity after using two DL methods to perform the traffic classification as these approaches use many computations operations. Hence the system consumes the network resources.

Jankowski and Marek [18] implemented five IDS models in SDN networks, using several ML algorithms: self-organising Maps (SOM), Learning Vector Quantization (LVQ1), and some modified editions. The machine learning algorithms that were used in this work are as follows: Self-Organising Maps (SOM), Multi-pass Self-Organizing Maps (M-SOM), Learning Vector Quantization (LVQ1), Multi-pass Learning Vector Quantization (M-LVQ1), and Hierarchical Learning Vector Quantization (H-LVQ1). The proposed models detect multi-level attacks, such as Prop, U2R, R2L, and DoS by classifying the incoming traffic. All the models implemented showed that they were achievable: the True Positive Rate was 94% on average. However, the authors generated a dataset containing features that were native and easily extracted from the packet header. Therefore, the model implemented will not detect real network attacks and does not cover the attack's behaviour.

Van *et al.* [19] introduced an anomaly-based detection method. The proposed model helps to prevent and detect known and unknown attacks in SDN networks. The J48-tree algorithm was used, which is a type of decision tree algorithm developed for classification purposes. The proposed model was implemented using a NetFPGA10G board. The system achieved a 91.81% detection rate and a 0.55% false alarm rate, and the KDD'99 public dataset was used in the training and test stages. However, they did not consider the high volume of data in a large-scale network. These require time and energy for efficient processing. An overload is generated on the controller and switches, where the OpenFlow switches check each incoming packet. Additionally,

the proposed system extracts too many features in the investigation stage, which causes increased consumption of the network's resources.

By classifying incoming traffic using the ML algorithms, Barki *et al.* in [20] introduced a method to deal with dynamic nature of the SDN to discover DDoS attack in the application plane. The selected ML algorithms were Naive Bayes, KNN, K-means, and K-medoids. The experiment used a private dataset obtained from a traced file that belongs to the real network to train and test the models. The detection rate metric was used to evaluate the algorithms implemented and the results were 94%, 90%, 86%, and 88%, respectively. However, the controller extracts 50 features for each incoming traffic event. This high volume of features requires more memory and lengthy processing, so when the attack starts, after a short time, the controller suffers from a substantial overload.

Lataha and Toker [21] attempted to prove that SDN can be part of a DoS attack solution. The proposed model separates the intrusion detection into two phases: flow based, and packet based. The disadvantage of high resource use emerges, when filtering and analysing packets in two states, similar to stateful and stateless firewalls. These problems cause the model to face performance degradation, owing to extensive incoming data rates. The proposed model can detect malicious flows by employing the KNN approach. However, such an algorithm is efficient in binary classification. As a result, the proposed approach has 91.27% accuracy and 0.99% precision, compared with algorithms, such as KNN, using the NSL-KDD dataset, neural networks, and others. The results were better under the same conditions. Even when the false positive rate improved, handling packet processing exhausted the controller.

Jiaqi *et al.* [22] proposed an ML approach in a 5G based SDN environment to identify DDoS, DoS, U2R, and R2L attacks. The K-means++ and AdaBoost algorithms have been used to classify traffic, and the Random Forest (RF) algorithm has been used for feature selection. The authors used the KDD Cup 1999 dataset, which is widely used in IDS evaluation. The model achieved an average classification accuracy of 84%. However, the Random Forest (RF) algorithm failed to select related features that cover the behaviours of U2R and R2L attacks, as it has achieved a low accuracy of detection. The selected features are insufficient to cover the behaviours of the attack. The authors did not consider the controller's bottleneck, where the controller at the large-scale network does not perform the functionality efficiently. Therefore, the system requires a lightweight method to process packets efficiently.

## III. MODEL DESIGN

Our model is a new IDS, designed for the SDN context called Naive Bayes - Protection System based Distribution Process (NB-PSDP). The proposed framework includes a Features Extractor, Analyser, and Decision Maker modules, as shown in Fig.2. Fig.3 depicts the NB-PSDP design across the SDN

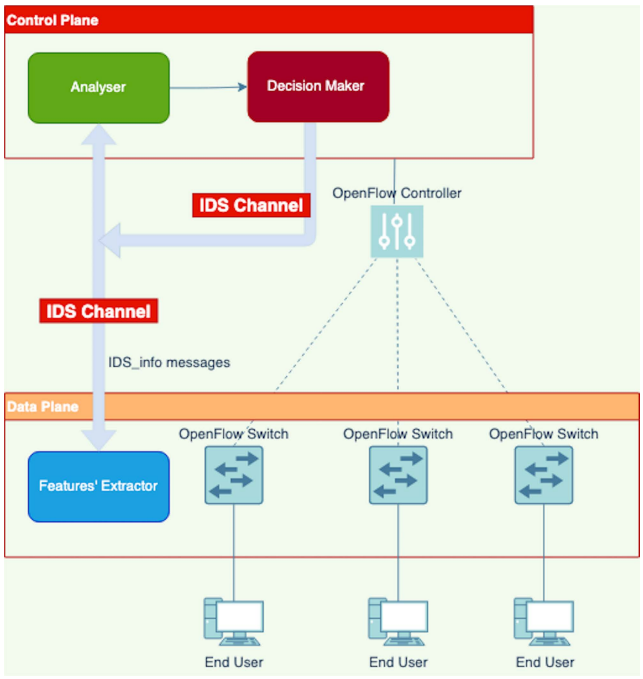


**FIGURE 2.** The model overview with the network topology.

planes. The model detects 13 types of attack in real time. These types of attack are listed in Table 2. Our proposed system distributes the processing steps across the SDN layers and utilises an appropriate number of features to transmit them via a new, trusted channel in order to avoid controller overload. Due to its lower complexity and negligible overload, the NB-PSDP is a reliable and flexible protection system for large SDN networks. Explanations of the modules are discussed below.

### A. FEATURES' EXTRACTOR

This module contains new terms that should be added to the OpenFlow switch. These terms include: *IDS_info_request* message, *IDS_info_reply* message, *IDS_info_ID*, and the IDS channel. The switch receives an *IDS_info_request* with an allocated *IDS_info_ID* from the controller. Then, the switch, in turn, sends *IDS_info_reply* with the assigned *IDS_info_ID* to the controller via the IDS channel. The terms above replace *ofp_flow_stats* request and reply messages.

If the controller sends an *IDS_info_request* message to the switch, this module extracts the selected features from the flow table. The switch sends the extracted features to the controller, using an *IDS_info_reply* message. These messages use the IDS channel to transfer data between the data and control planes. Since the traditional IDSs exhaust the OpenFlow channel by demanding extensive requests, we invented the IDS channel to transmit the *IDS_info_request* message, instead of the OpenFlow channel, in order to reduce overload in the OpenFlow channel, when installing such systems [2]. In addition, the OpenFlow channel transfers other data between the controller and switches, such

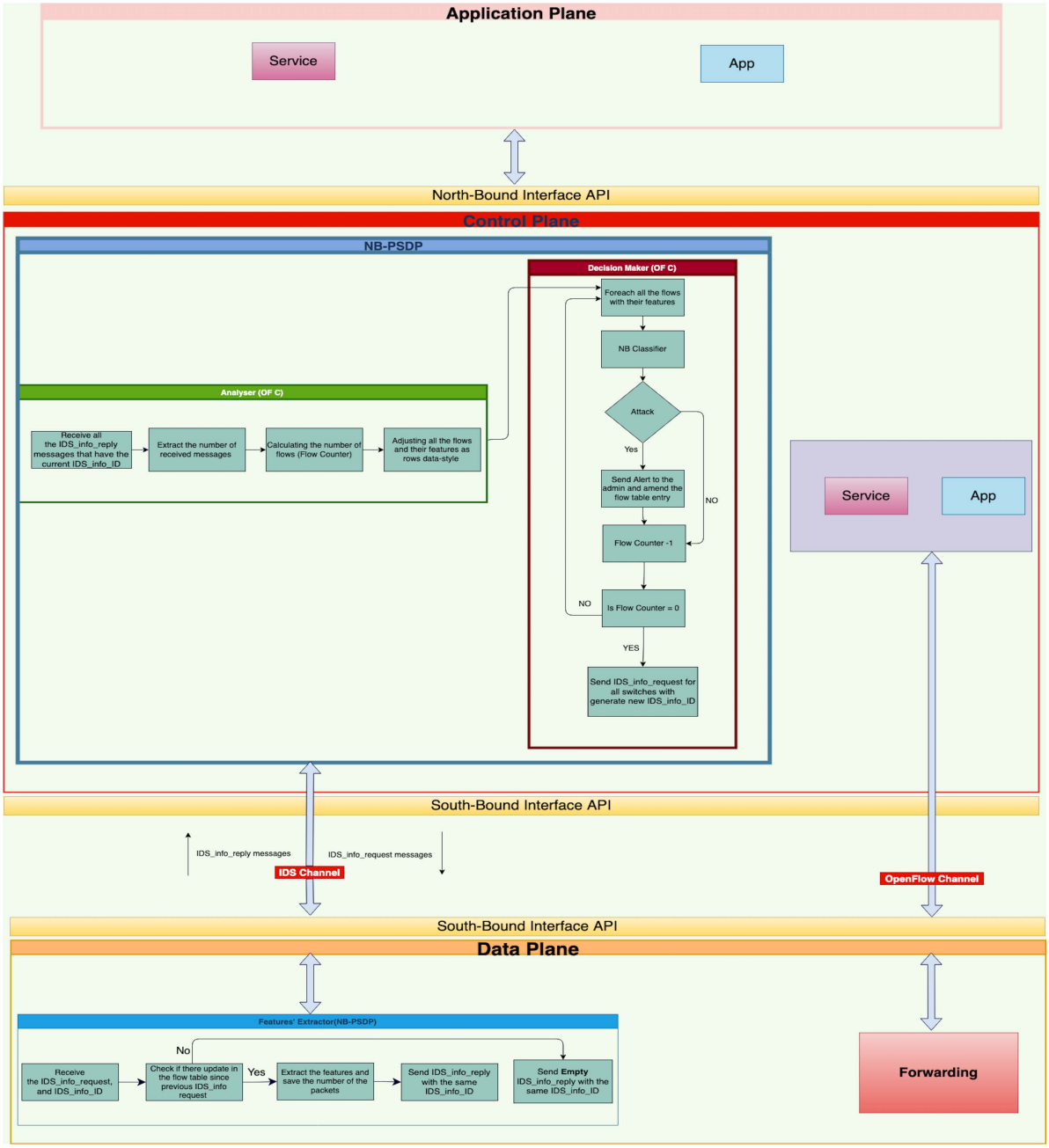


**FIGURE 3.** The system design.

as synchronise, hello messages, installing/amending flow entries, and *Packet_in*/ *Packet_out* messages [23]. Therefore, the load on the OpenFlow channel increases, when it is used to exchange IDS data. For this reason, the proposed model distributes the processing between the switches and the controller, instead of only the controller. At the same time, most of the current research merely used the controller to extract all the flow features. This module accomplishes the extraction features for each switch independently inside the OpenFlow switch, as depicted in Fig.3 and Algorithm 1. As a result, this module minimises the overload of the network.

### B. ANALYSER MODULE

At this stage, this module passes all *IDS_info_reply* messages and converts them into row-data style. Each row represents a single flow with its own features. The module then forwards all these features to the Decision Maker module, as shown in Fig.3. This module considers as pre-processing stage, it prepares the *IDS_info_reply* messages to be used by the following module.

### C. DECISION MAKER MODULE

This module employs the Naive Bayes classifier to forecast whether the flows are normal or attacked. NB is a

**Algorithm 1:** IDS Info Reply Message in Switch.

**Input:**
*IDS_info_ID*, *last_packet_count*
**Output:**
*last_packet_count*
**Procedure** `IDSReply` (*IDS_info_ID, last_packet_count*) **:**
  *Count_all_packets* ⟵ *count*(flows);
  **if** *Count_all_packets* ≠ *last_packet_count* **then**
    *features* = the 14 features for all Flows
    Send(IDS_info_reply(*features*), *IDS_info_ID*);
  **else**
    Send(IDS_info_reply(*Null*), *IDS_info_ID*);
  **end**
  **return** *last_packet_count*;
**End Procedure**

**Algorithm 2:** IDS Info Requests in Controller.

**Input:**
*IDS_info_ID*, *IDS_info_reply*
**Procedure** `IDSRequest` (*IDS_info_ID, IDS_info_reply*) **:**
  *Count* ⟵ *count*(IDS_info_reply);
  *features*[] = *IDS_info_reply*;
  **while** *Count* > *0* **do**
    trafficType = NB( *features*[Count]);
    **if** *trafficType* == *Attack* **then**
      Send(FlowEdit);
      Send(AlertToAdmin);
    **end**
    *Count* ⟵ *Count* − 1;
  **end**
  *IDS_info_ID* ⟵ *IDS_info_ID* + 1;
  Send(*IDS_info_request*, *IDS_info_ID*);
  **return** *IDS_info_ID*;
**End Procedure**

classification technique derived from Bayes' Theorem. The NB algorithm is considered a fast tool that assumes a low independence condition between the features [24]. Bayes' Theorem utilises to compute the posterior probability for the given classes. Equation 1 has been used to calculate the posterior probability [25]:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{1}$$

where P(c|x) indicates the probability of the predictor group c (x, features), P(c) represents the preceding probability of the class; P(x|c) represents the probability of a feature in the current class. P(x) defines the likelihood of a feature preceding it [25].

The Decision Maker module receives all current flows from the Analyser module. After receiving all the flows, this module stores the flows in a matrix and sends each flow to the NB algorithm. If NB categorised the flow as regular, the counter of the flows decreases. However, if the flow is classified as malicious, the controller instigates two events. The first event orders the controller to update the flow table of the predicted flow to teach the switch to drop any further matching packets. The second event warns the administrator to take further action. In this case, the counter of the flows will also be decreased and proceed to the following flow of the current request. The flow counter concept has already been published in [26].

At the end, if the counter of flows has elapsed, the Decision Maker module will generate a new *IDS_info_ID* and send new *IDS_info_request* messages to all connected switches, through the IDS channel, as shown in Algorithm 2 and Fig.3.

## IV. PERFORMANCE ANALYSIS AND RESULTS

### A. DATASET AND SELECTED FEATURES

We used the CSE-CIC-IDS2018 dataset, available online on [27]. This dataset consists of 13 types of attacks, as listed in Table 2.

**TABLE 2.** The targeted attacks in the NB-PSDP model.

| # | Attack name |
|---|---|
| 1 | Document infiltration |
| 2 | Brute Force -Web |
| 3 | Brute Force -XSS |
| 4 | DDOS attack-LOIC-UDP |
| 5 | DoS attacks-GoldenEye |
| 6 | FTP-BruteForce |
| 7 | SQL Injection |
| 8 | DoS attacks-Hulk |
| 9 | DoS attacks-Slowlori1 |
| 10 | DoS attacks-LOIC-HTTP |
| 11 | Bot |
| 12 | SSH-Bruteforce |
| 13 | DDOS attack-HOIC |

Many researchers only utilise basic flow features, which are insufficient for accurately identifying the attack type. Furthermore, the large number of features consumes the network's resources and causes saturation of the controller. Therefore, the T-Test method has been utilised to perform the feature selection. This method is kind of the univariate feature selection methods. T-Test examines every factor to determine the score of the feature's relationship with the classes [28]. This method calculates the scores for each feature individually, based on Equation 2:

$$t = \frac{(X_1 - X_2)}{\sqrt{\frac{(S_1)^2}{n_1} + \frac{(S_2)^2}{n_2}}} \tag{2}$$

where $X_1$ is the mean of feature 1. $X_2$ is the mean of group 2. $S_1$ and $S_2$ represent the standard deviation of group 1 and group 2 [29].

The 14 features with higher scores among the other features were selected by using T-Test method, as shown in Fig.4 and Table 3. Before applying the T-Test algorithm, we removed some features that cannot be used in the SDN environment. The T-Test method was selected to invest the

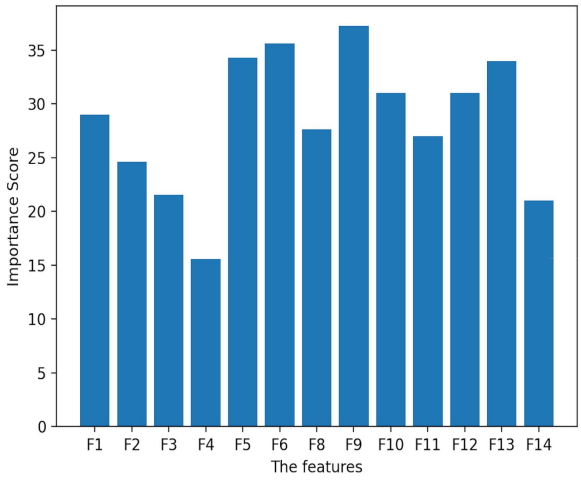


**FIGURE 4.** Features importance score using T-test.

**TABLE 3.** The selected features description [27].

| # | Feature | Description |
|---|---|---|
| 1 | folder | Flow duration |
| 2 | tot_fw_pk | Number of packets that sent to the direction |
| 3 | tot_l_fw_pkt | Total size of packets that sent to the direction |
| 4 | fw_pkt_l_avg | Average size of packets that sent to the direction |
| 5 | fl_iat_avg | Average time between two flows |
| 6 | pkt_size_avg | Average size of packet |
| 7 | idl_avg | The average of idle time out in that flow |
| 8 | fl_pkt_s | Flow packets rate that are sent per second |
| 9 | fl_byt_s | Flow byte rate that sent per second |
| 10 | fw_pkt_s | Average of the transmitted packets per second |
| 11 | atv_avg | Mean time a flow was active before becoming expire |
| 12 | down_up_ratio | Download and upload ratio |
| 13 | Dst Port | The port number of the flow destination |
| 14 | Protocol | The protocol type (UDP, TCP, etc..) |

maximum benefit of the NB algorithm, which both work perfectly with high independence features [24], [29]. Therefore, within this number of features, the model avoids long-time processing without exhausting the switches and controller. In addition, we adjusted the CSE-CIC-IDS2018 dataset to include just two labels: 0 for benign flows and 1 for all types of attacks.

### B. THE DETECTION TECHNIQUE ANALYSIS

We utilised various metrics to measure the efficiency of our NB-PSDP and the other detection techniques. These metrics are commonly used to evaluate the IDSs based ML approaches.

#### 1) EVALUATION METRICS

The following formulas were employed to assess the performance of the NB classifier and the other algorithms used in the comparisons.

- False Positive (FP): is the number of normal flows that are mispredicted.
- True Positive (TP): is the number of attacked flows that are expected correctly.

**TABLE 4.** The experiment results with CSE-CIC-IDS2018 dataset.

| Classifier | AC | P | R | F1 |
|---|---|---|---|---|
| RF | 89.12% | 89.04% | 89% | 89.02% |
| DNN | 82.65% | 83.87% | 82.41% | 83.13% |
| KNN | 92.27% | 92.68% | 90.17% | 91.41% |
| SVM | 74.74% | 73.95% | 74.29% | 74.12% |
| AdaBoost | 83.99% | 83% | 84.71% | 83.85% |
| XGBoost | 94.73% | 94.82% | 94.12% | 94.47% |
| **Our NB** | **98.46%** | **96.15%** | **100%** | **98.03%** |

- False Negative (FN): is the number of attacked flows that are mispredicted.
- True Negative (TN): is the number of normal flows that are expected correctly.
- Accuracy (AC): is the accurate rate of predictor over the total flow. The following formula calculates the accuracy:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{3}$$

- Precision (P): is utilised to get the percentage of attack detection rate. P computes by using formula 4:

$$P = \frac{TP}{TP + FP} \times 100\% \tag{4}$$

- Recall (R): is represent the rate of the number of classified attacks to the total number attack flows. R calculates by utilising equation 5.

$$R = \frac{TP}{TP + FN} \times 100\% \tag{5}$$

- F1-measure (F1): it evaluates the more reliable model accuracy rate based on R and P rates. Formula 6 utilises to compute F1 score.

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \times 100\% \tag{6}$$

The classifiers in Table 4 have been selected based on their complexity in order to conduct fair evaluation. Furthermore, all these algorithms have examined under a similar conditions, same dataset, and same parameters that utilised with our NB-PSDP.

#### 2) EXPERIMENTAL SETUP AND RESULTS

All the selected classifiers including our NB were evaluated using labelled datasets. Each classifier was utilised to categorise the flows in real-time. TensorFlow and Keras libraries were employed to perform the ML algorithms. TensorFlow [30] and Keras [31] are modern libraries used in Python to implement the ML and DL applications.

The results of the detection techniques are presented in Table 4 and the confusion matrix of our model is shown in Fig.5. The NB algorithm has been achieved the highest rate and best performance among other classifiers. The proposed IDS reached a maximum accuracy, precision, recall, and

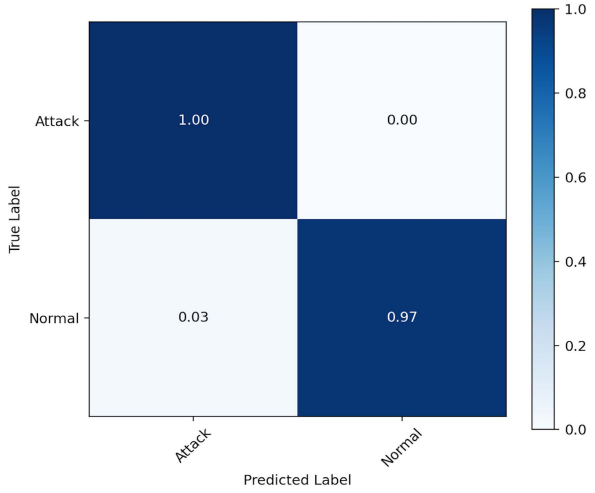


**FIGURE 5.** Confusion matrix of our NB.

F-Score of 98.46%, 96.15%, 100%, and 98.03%, respectively. The NB classifier predicted the attack with 100% accuracy, as illustrated in the confusion matrix shown in Fig.5. Moreover, all the experiments have been validated using the 5-fold cross-validation technique. Hence, these experiments prove that the NB classifier with the univariate selection method produced an efficient approach for detecting several attacks in real time.

### C. NETWORK PERFORMANCE ANALYSIS

The integration between a controller and an IDS is essential, especially when implementing an IDS in large networks. Throughput and latency benchmarks were applied to assess the performance of the NB-PSDP. Furthermore, CPU utilisation has been used to evaluate the controller overload and show how the proposed model reduces the stress on network resources.

#### 1) EVALUATION METRICS

To evaluate the network performance, we used the Cbench benchmark, which was developed to estimate the throughput and latency of the SDN controllers [32]. Cbench is an open-source tool, programmed by utilising the C programming language [33].

- Throughput: represents the number of processes that a system can handle within a fixed duration. Cbench counts the number of *packet_in* events handled per second.
- Latency: determines the time required for a single process to be completed by a system, and is also known as delay. Cbench calculates the time required to complete a single process in latency mode.
- CPU Utilisation: specifies the percentage of the total CPU capacity of the controller used during the IDS implementation.

#### 2) EXPERIMENTAL SETUP AND RESULTS

The Mininet platform and POX controller were employed to execute NB-PSDP. In the throughput evaluation, Cbench employs the switches to forward a significant number of *packet_in* events to the control plane, and determines the number of *packet_out* replies handled during the time unit. Simultaneously, in latency mode, the switches forward a single *packet_in* event to the control plane and calculate the duration used to process this request.

The experiments were conducted using six scenarios, all of which involved the following numbers of switches: 8, 16, 32, 64, 128, and 256. Switches were attached to the POX controller and each switch was connected to 500 MAC addresses. These scenarios have been executed individually: the POX controller only, NB-PSDP, and state-of-the-art methods. At the same time, the NB-PSDP model and state-of-the-art methods were implemented in the POX controller.

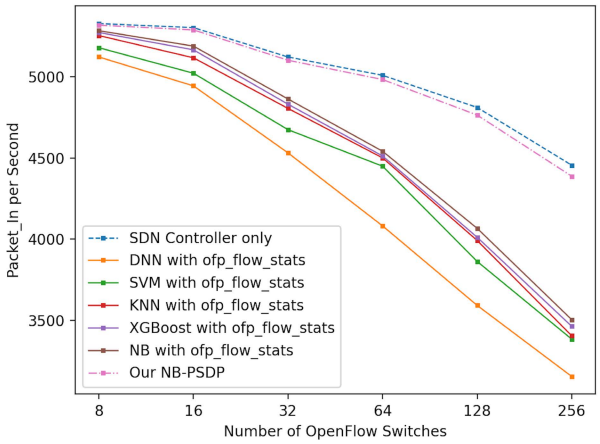


**FIGURE 6.** Throughput evaluation.

The results of the throughput scenarios are shown in Fig.6. In all cases, the throughput decreased, when the network expanded. This behaviour is expected, as the size of the network broadly affects the network performance. After obtaining the results from Cbench tool, we computed the decreasing rate of the controller in each scenario using the following formula:

$$Decreasing\ rate = \frac{newvalue - oldvalue}{oldvalue} \times 100\% \quad (7)$$

The decreasing rate values were calculated relative to SDN controller throughput. The results of decreasing NB-PSDP throughput rates were approximately between 0.18% and 1.5%. The 1.5% rate was observed in the case of 256 switches. The previously mentioned rates are not significant and do not affect the network performance. In contrast, as shown in Fig.6, we conducted other experiments using state-of-the-art methods and found all scenarios had a high decreasing rate of throughput: between 1.4% and 29.5%. The 29.5% rate is considered to be too high and consumes the network's resources.

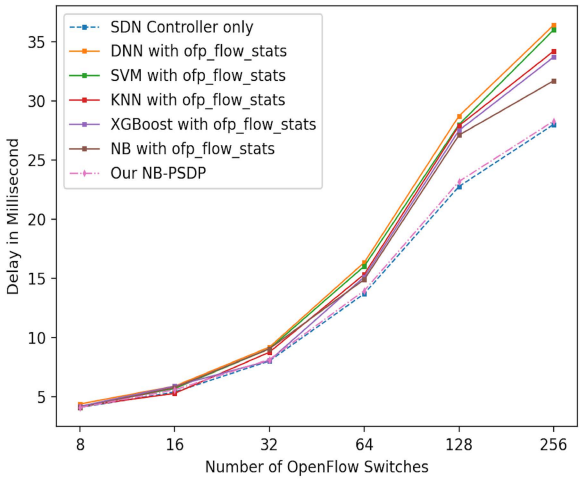


**FIGURE 7.** Latency evaluation.

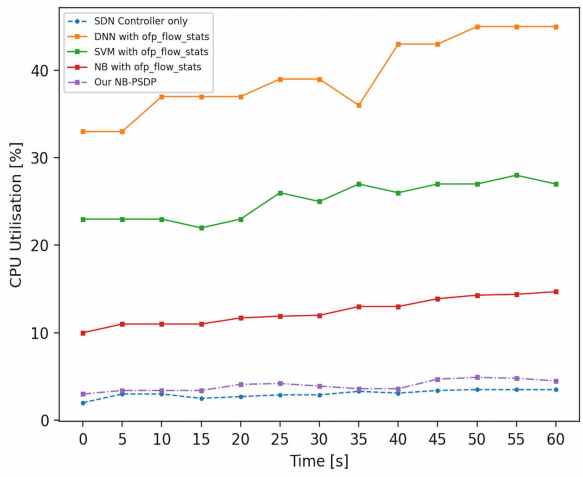


**FIGURE 8.** CPU utilisation.

As shown in Fig.7, the NB-PSDP has a low decreasing rate in latency. The maximum decreasing rates were between 0.18% and 0.7% when the controller was connected to 256 switches. These values do not impact the network performance. Compared with the selected state-of-the-art methods, we can observe that the decreasing rates of the high latency were between 1.2% and 30%. The 30% rate is considered too high and negatively affects the network performance.

The CPU usage average in terms of the SDN controller was only around 3%. While in the case of applying state-of-the-art methods, the CPU usage rates were between 15% and 45% of the total CPU capacity. However, these rates are relatively high and reflect the way in which traditional approaches consume resources. Yet, the usage rate of our model was around 4% of the total CPU capability, so the decentralisation approach in the proposed model prevents resources from being exhausted. Therefore, the experiments prove that the NB-PSDP can be efficiently implemented on large-scale networks.

These improvements indicate that our NB-PSDP uses a novel framework, utilising a new independent channel to transmit the required data. The model distributes the proceeding stages of handling data between the controller and the switches. In contrast, most studies simply used the controller to extract and sort the requested data; this process exhausts the OpenFlow channel and controller, when implementing an IDS. Most researchers use *ofp_flow_stats* events to collect flow tables for each fixed time. These events cause congestion and overload in the controller and hinder the network performance. Therefore, we designed a novel framework to solve these issues and allow sufficient time for the controller to handle all the requests. Moreover, the NB-PSDP uses only 14 features. Latency, throughput and CPU utilisation evaluations prove that our model can be successfully implemented in large-enterprise networks.

## V. CONCLUSION

This paper has offered a brief literature review of IDSs and proposed a novel security model for SDN networks, called NB-PSDP, which can categorise network flows in real time, using the NB classifier. The system utilises a new independent channel to transmit data between switches and the controller, employing a novel extension called *IDS_info*, in order to avoid requesting unwanted data from the flow tables, and it distributes the processing steps over the SDN planes. Moreover, the proposed model employs the Univariate Selection algorithm to only select the relative features that cover attack behaviours. Furthermore, NB-PSDP protects SDN networks from various attacks, including document infiltration, SSH-Bruteforce, Brute Force-XSS, and DoS attacks-Slowloris.

The experiments proved that our NB-PSDP has the highest accuracy: 98.46% in flow classification and 100% recall, compared with other ML algorithms. Moreover, network performance evaluation shows that the proposed system has high reliability and integrity in both large and small enterprise networks. In future work, we intend to extend the framework and add a new dynamic module to conduct a balanced real time trade-off between the security level and network performance. Finally, we will utilise other performance metrics to perform a comprehensive evaluation.

## VI. DATA STATEMENT

No new data were created during this study. Pre-existing data underpinning this publication were obtained from CSE-CIC-IDS2018 dataset (https://www.unb.ca/cic/datasets/ids-2018.html) and are subject to licence restrictions.

## REFERENCES

[1] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in SDN-based networks: Deep recurrent neural network approach," in *Deep Learning Applications for Cyber Security*. Cham, Switzerland: Springer, 2019, pp. 175–195.

[2] M. Ahmed, S. Shatabda, A. K. M. Islam, M. Robin, and T. Islam, "Intrusion detection system in software-defined networks using machine learning and deep learning techniques—A comprehensive survey," Tech. Rep., 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/Intrusion_Detection_System_in_Software-Defined_Networks_Using_Machine_Learning_and_Deep_Learning_Techniques_A_Comprehensive_Survey/17153213

[3] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "A survey and classification of the security anomaly detection mechanisms in software defined networks," *Cluster Comput.*, vol. 24, no. 2, pp. 1235–1253, Jun. 2021.

[4] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.

[5] M. Hanselmann, T. Strauss, K. Dormann, and H. Ulmer, "CANet: An unsupervised intrusion detection system for high dimensional CAN bus data," *IEEE Access*, vol. 8, pp. 58194–58205, 2020.

[6] R. S. Krishnan, E. G. Julie, Y. H. Robinson, R. Kumar, L. H. Son, T. A. Tuan, and H. V. Long, "Modified zone based intrusion detection system for security enhancement in mobile ad hoc networks," *Wireless Netw.*, vol. 26, no. 2, pp. 1275–1289, Feb. 2020.

[7] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based DDoS defense mechanisms," *ACM Comput. Surv.*, vol. 52, no. 2, pp. 1–36, Mar. 2020.

[8] R. Palanikumar and K. Ramasamy, "Software defined network based self-diagnosing faulty node detection scheme for surveillance applications," *Comput. Commun.*, vol. 152, pp. 333–337, Feb. 2020.

[9] D. P. Isravel, S. Silas, and E. B. Rajsingh, "Centrality based congestion detection using reinforcement learning approach for traffic engineering in hybrid SDN," *J. Netw. Syst. Manage.*, vol. 30, no. 1, pp. 1–22, Jan. 2022.

[10] S. Kaur, K. Kumar, N. Aggarwal, and G. Singh, "A comprehensive survey of DDoS defense solutions in SDN: Taxonomy, research challenges, and future directions," *Comput. Secur.*, vol. 110, Nov. 2021, Art. no. 102423.

[11] N. Ravi and S. M. Shalinie, "BlackNurse-SC: A novel attack on SDN controller," *IEEE Commun. Lett.*, vol. 25, no. 7, pp. 2146–2150, Jul. 2021.

[12] W. Ren, Y. Sun, H. Luo, and M. Guizani, "SILedger: A blockchain and ABE-based access control for applications in SDN-IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4406–4419, Dec. 2021.

[13] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A DDoS attack detection method based on SVM in software defined network," *Secur. Commun. Netw.*, vol. 2018, pp. 1–8, Apr. 2018.

[14] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *Proc. 26th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2017, pp. 1–9, doi: 10.1109/ICCCN.2017.8038436.

[15] R. Sathya and R. Thangarajan, "Efficient anomaly detection and mitigation in software defined networking environment," in *Proc. 2nd Int. Conf. Electron. Commun. Syst. (ICECS)*, Feb. 2015, pp. 479–484.

[16] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible network-based intrusion detection and prevention system on software-defined networks," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, Nov. 2015, pp. 106–111.

[17] H. Polat, M. Türkoğlu, O. Polat, and A. Şengür, "A novel approach for accurate detection of the DDoS attacks in SDN-based SCADA systems based on deep recurrent neural networks," *Expert Syst. Appl.*, vol. 197, Jul. 2022, Art. no. 116748.

[18] D. Jankowski and M. Amanowicz, "On efficiency of selected machine learning algorithms for intrusion detection in software defined networks," *Int. J. Electron. Telecommun.*, vol. 62, no. 3, pp. 247–252, Sep. 2016.

[19] N. T. Van, H. Bao, and T. N. Thinh, "An anomaly-based intrusion detection architecture integrated on OpenFlow switch," in *Proc. 6th Int. Conf. Commun. Netw. Secur.*, Nov. 2016, pp. 99–103.

[20] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2016, pp. 2576–2581.

[21] M. Latah and L. Toker, "Minimizing false positive rate for DoS attack detection: A hybrid SDN-based approach," *ICT Exp.*, vol. 6, no. 2, pp. 125–127, Jun. 2020.

[22] J. Li, Z. Zhao, and R. Li, "Machine learning-based IDS for software-defined 5G network," *IET Netw.*, vol. 7, no. 2, pp. 53–60, Mar. 2018.

[23] L. C. Costa, A. B. Vieira, E. de Britto e Silva, D. F. Macedo, L. F. M. Vieira, M. A. M. Vieira, M. da Rocha Miranda, G. F. Batista, A. H. Polizer, A. V. G. S. Gonçalves, G. Gomes, and L. H. A. Correia, "OpenFlow data planes performance evaluation," *Perform. Eval.*, vol. 147, May 2021, Art. no. 102194, doi: 10.1016/j.peva.2021.102194.

[24] K. Rrmoku, B. Selimi, and L. Ahmedi, "Application of trust in recommender systems—Utilizing naive Bayes classifier," *Computation*, vol. 10, no. 1, p. 6, Jan. 2022, doi: 10.3390/computation10010006.

[25] H. Asgharnezhad, A. Shamsi, R. Alizadehsani, A. Khosravi, S. Nahavandi, Z. A. Sani, D. Srinivasan, and S. M. S. Islam, "Objective evaluation of deep uncertainty predictions for COVID-19 detection," *Sci. Rep.*, vol. 12, no. 1, pp. 1–11, Dec. 2022.

[26] A. H. Janabi, T. Kanakis, and M. Johnson, "Convolutional neural network based algorithm for early warning proactive system security in software defined networks," *IEEE Access*, vol. 10, pp. 14301–14310, 2022, doi: 10.1109/ACCESS.2022.3148134.

[27] University of New Brunswick. *CSE-CIC-IDS2018 Dataset*. Accessed: Dec. 2, 2021. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2018.html

[28] S. Jain and A. Saha, "Rank-based univariate feature selection methods on machine learning classifiers for code smell detection," *Evol. Intell.*, vol. 15, no. 1, pp. 609–638, Mar. 2022.

[29] Q. Liu and L. Wang, "*t*-test and ANOVA for data with ceiling and/or floor effects," *Behav. Res. Methods*, vol. 53, no. 1, pp. 264–277, Feb. 2021.

[30] TensorFlow. *TensorFlow 2 Quickstart for Experts*. Accessed: Jan. 7, 2022. [Online]. Available: https://www.tensorflow.org

[31] Keras. *Developer Guides*. Accessed: Jan. 7, 2022. [Online]. Available: https://keras.io

[32] R. Jawaharan, P. M. Mohan, T. Das, and M. Gurusamy, "Empirical evaluation of SDN controllers using Mininet/Wireshark and comparison with cbench," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–2.

[33] (2013). *CBench. GitHub*. Accessed: Jan. 26, 2021. [Online]. Available: https://github.com/mininet/oflops/tree/master/cbench