Intrusion Detection System for Cloud Based Software-Defined Networks

Omar Jamal Ibrahim*, Wesam S. Bhaya*
University of Babylon, College of Information Technology, Iraq.

omarjamal@itnet.uobabylon.edu.iq
wesambhaya@uobabylon.edu.iq

Abstract. Software-Defined Networks is a programmable network architecture for the cloud programmable control plane, decoupled from its data plane, offers new possibilities for creative security measures for overall visibility of the status network. This paper leverages these capabilities of SDN and presents the software-enabled Intrusion Detection System (IDS) architecture using the consideration of SDN. It combines the advantages of machine learning with IDS to ensure a high detection rate and protect the network from attacks. The Python script was utilized the Mininet emulator to create a virtual network. Also, it has been used as an OpenDaylight software as an SDN controller hosted at a Google cloud. The proposed IDS uses a GridSearch technique with Support Vector Machine (SVM) to detect anomaly of attack. The proposed work was trained on UNSW-NB15 and NSL-KDD datasets. The results show that the proposed system offers a high detection rate. With the proposed machine learning model, the detection rate becomes more than 99.8 percent of accuracy. The results show positive progress in detecting almost all possible network attacks in the SDN-based cloud environment.

Keywords. IDS, Support Vector Machine with Grid Search, OpenDaylight, Google Cloud, and Mininet.

1. Introduction
Due to versatility, high availability, scalability, performance, and so on[1], [2] cloud computing systems have been commonly introduced and perform [3]. Protection has been identified as one of the most important issues when cloud abuse and malicious attacks are among the top threats in existing cloud systems [4]. Based on affected cloud resources, attackers can spam, spread malicious code, crack passwords and security keys, compromise vulnerable VMs, and then launch DDoS attacks, use and control botnets, etc. Existing countermeasures generally involve additional and flexible protection models and enable the cloud to provide the necessary tools. The ONF represents the SDN framework as a physical separation between the control plane and the forwarding plane. SDN is an architecture that separates the functions of control and transmission. It helps make your network scalable, adaptable, open, programmable, and easy to manage [5]. The advent of cloud computing enables the logical centralization and distribution of the control plane on cloud platforms. Complementary reactive safety mechanisms such as IDS is, therefore, wanted [2]. The IDS detects intrusion through the processing of data from the network. IDS (NIDS), data is obtained immediately from the network by packet inspection. IDSs conventional such as Snort [3] and Suricata [4] evaluate all packets. The large volume of data
causes this to reduce the execution of the network and increases network delays and more infrastructure processing costs [4].

Inside an SDN, the idea of a stateful heart may be used for flows and central control for better safety features. Based on these advantages of SDN, this work reveals the design of an IDS system that is lightweight and flexible and functions in the same way as a conventional packet-based system. The developed solution consists of a light flow-dependent IDS that implements a detection scheme dependent on an anomaly trained by an SVM with a grid quest and a Random Forest processor. On the next step, a suspicious flow is transmitted through a packet dependent IDS for more analysis utilizing a signature-based identification. The scheme combines the advantages of SDN and the recent progress of machine learning to deliver a scalable approach that can be periodically modified with a new data set by shaping the algorithm.

2. Related Works
This section discusses in-depth the related research on Machine Learning network attack detection, in particular the methods and steps taken to address the issue of NIDS attacks classification. Illustrated all of the works studied in this paper. This work each consists of a data set and all steps that have been used in each proposed solution, including Preprocessing Data, Feature Engineering, Preprocessing Data, Hyperparameter Selection, Feature Selection, Machine Learning Method, and Performance Metrics, and the full detection procedure. Finally, we illustrate the various measures used in this study to evaluate the proposed technique.

The research done in [6] revealed weaknesses in various previous work on the classification of network attacks when considering an obsolete dataset, such as KDDCup’99. Other similar studies usually reach high classification rates but are mainly due to limitations in the KDD’99 dataset. This presumption is verified by comparing the accuracy of [7]classical ML algorithms with that obtained using the same ML method. While the same ML algorithms were used for comparison, the method followed by the authors did not provide detailed information. In[7] the authors recommended IDS architecture, which addressed new issues related to the control of high network traffic volumes in real-time. The system was evaluated using a well-known KDDCup’99 Machine Learning algorithm designed to test the performance of the attack classification. However, better results are possible for the proposed FS method in the designing and detective timeframe, with a manual exploratory study of two techniques called Backward Selection Rankings and Forward Selection Rankings. The writers said little about FE, DP, and HS, making it difficult to reproduce and compare reported results.

It’s Sharafaldin et al. [8] offers a realistic network data collection consisting of modified attacks and standard traffic patterns in an emulated network environment. We reported that the data set can be used with up to seven classic machine learning algorithms to detect network attacks. Finally, a range of network-based flow features has been defined by a specialized tool. Then they carried out an FS procedure, known as the Random Forest Regressor, which weighted by the weight of an attack. However, neither HS nor DP method have been identified.

Li et al. provide a deep learning approach to the identification of network anomalies[9]. For this study, the authors tested a neural LSTM (Long Short-Term Memory) and a basic learning technique (BLS) based on anomaly detection methods. The NSL-KDD and the Border Gateway Protocol data gathering are included in two network data sets. The optimal neural network architecture was selected using the HS manual method. Further, standardization of datasets and a dummy FE strategy are part of their methods. Finally, the BLS approach was promoted as a feasible alternative to traditional approaches to deep education as offered a similar rate of success with less training time.

In[10], a new rapacious FS approach was proposed to improvement the efficiency of attack detection in communications networks. The evaluation of the NSL-KDD and ICSX12 data sets was deemed along with a wide variety of classical the Machine Learning and Deep Learning methods. The authors complete that the approach was feasible in terms of the classification of production and energy usage. Nevertheless, no FE, DP, and HS steps have been referred to or proposed in this study.
Cordero et al. [11] have implemented a method for boosting previously created network data sets using the I2dt tools. I2DT can apply a variety of attacks to present datasets, thereby supplying a valuable approach that does not reduce the time spent on constructing an efficient NIDS evaluation dataset, according to the authors. In this way, I2DT was utilized to update the MAWI data gathering and lower utilization in the port attack detection port and Replicator Neural Networks DoS [12] process. RNN solutions have an anomaly score that suggests an abnormality at a certain threshold. As with other plays, DP and FS methods have not been listed.

The [13] Optimal Allocation Least Square SVM for incremental data sets was proposed. When dealing with incremental data sets, the data set size can inevitably be also big to train models over a reasonable period. The proposed approach utilized the Optimal Allocation method to pick a rep model from the dataset first and thereafter the LS-SVM model, using selective samples only, to learn the problem of classification. This methodology has been tested for anomalies in the KDD’99 data set [14]. The writer contrasts two variants of the method utilize all the features in the dataset or using some of them obtained by the PCA [15]. This is not recommended to avoid crucial stage DP, FE, and HS for appropriate NIDS comparisons.

In [16] researchers proposed a crossbred IDS way consisting of two various evolutionary algorithms, Artificial Fish Swarm and Artificial Bee Colony. The crossbred algorithm was utilized to make anomaly detection rules depend on a small subset of features created using the primary combined process. The inspired alternatives that too beat classical and profound learning strategies for detecting network attacks, but comparisons are more complicated due to the lack of knowledge on methods or steps.

It's Divekar et al. [17] recognized many shortcomings in the study of modern NIDs in KDDCup’99 [14] and recommended an examination of the performance of other classic ML models to test their suitability in the case of modern NIDS for two alternatives to this benchmark. Mean Impurity Decrease approach has been used to eliminate unwanted functionality. In addition to SMOTE, a random sampling technique was used to generate an acceptable data set with a reasonable number of samples in each class. Randomized GridSearch and five-fold cross-validation were used to configure the hyperparameters for the ML models. Again, there is no mention in the article of the FE system.

The authors proposed to apply several classic NIDS ML models to the UNSW-NB15 dataset in [18], taking into consideration all available features. Apache Spark analyzed the effectiveness of the methods tested. Naive Bayes was the quickest method of plan, however, Random Forest made predictions more precise. The writers presented many occasions obtained but did not include detail on the computational system utilized or any of the recommended steps (FE, FS, etc.).

3. Methodology
Throughout this section, the recommended methodology followed in this work is described in order to evaluate ML-based network IDSs and detect different kinds of network communications attacks, which was explained as follows:

3.1. Cloud Computing
In cloud computing, application layers have been divided into several working layers for resource allocation and application provision [19]. There are therefore four types of services: the network layer (IaaS), the platform layer (PaaS), the application layer (SaaS), and the security as a service layer (SecaaS). Every one of these levels offers a particular function to customers and is explained appropriately [20].

3.2. OpenDaylight Controller
An OpenDaylight controller is an SDN tool used for this project. It depends on the Abstraction Layer Services (SAL) that allow protocols other than OpenFlow. It is developed in java and applies to any Java frame. The OpenDaylight consortium founded OpenDaylight in 2013. This funding from a variety of organizations allows OpenDaylight equal to suppliers [22].
Figure 1 displays the architecture of the OpenDaylight controller. The controller uses programmable application interfaces (APIs) such as REST technology for communication with the orchestration and service layer of network applications. This will contain OpenStack Neutron, the Virtual Tenants Network administrator[23][24]. The controller layer operates a variety of services, including software abstraction layer (SAL), OpenStack applications, basic network service, and a variety of others. The controller utilizes southbound interfaces and protocol plugins including OpenFlow, OVSDB[25], network management Protocol, NetCONF[26], etc. to communicate with data plan elements[27].

3.3. Mininet
The container-based emulator is Mininet [28]. This authorizes you to run not modified code interactively on a standard machine with simulated hardware. Adds rest and originality at low hardware prices. If programs are added to actual live networks, simulator programs need little to no modification [29]. Mininet uses a not modified Linux container network technology to obtain its scalability and accuracy. Mininet backing applications based on OpenFlow (SDN). It offers an elastic and economic forum for designing, reviewing, and evaluating OpenFlow applications. The processes and software processes of virtual hosts run inside Mininet within the container. It allows them to use system resources separately and to participate in the kernel by other containers [30].

Mininet backing five interconnected network topologies. Such integrated topologies are Single, Minimal, Tree, Linear, and Reversed. The Mininet network topology can be changed by CLI [31].

SDN switch, controller, host, and links can be generated through coming in instructions via the Mininet command-line interface. Many Linux commands are given in the Mininet command-line interface (CLI). The most frequently utilized commands are nodes listing all of the produced nodes, dump lists network information, networks show how network elements are connected. CLI provides routine troubleshooting command that is used for computer networks. These commands contain pingall which outputs the outcome of the communication trial across all nodes. Ifconfig is as well allowed to view the Internet Protocol node information. Iperf is also propped and is an observation device for network efficiency. It utilizes a client/server model, anywhere traffic of the client begins and the network is passed to the server. Iperf produces network-supported datum test flow with time stamps and tracks data transmitted and measured throughput. Two types of transport protocols are provided in Iperf: UDP and TCP. Most programs such as File Transfer Program As transport protocol SMTP and Hypertext Transmission Protocol use TCP. Utilize TCP mode, Iperf checks the full transport layer TCP bandwidth. the UDP mode, Iperf tracks jitter, packet loss, and bandwidth. The UDP mode is good for tracking service fineness for phoneme and video participation applications.
3.4. Dataset

The following subsections show the attacks datasets which used to validate the proposal.

3.4.1. NSL-KDD

The NSL-KDD dataset includes four types of attacks: DoS, Probe, R2L, and U2R. The 41 features are classified into three groups: Basic, Traffic, and Content features. This data collection contains a minimum of 148,517 train and test sets [41].

3.4.2. UNSW-NB15

The UNSW-NB15 dataset contains nine types of new attacks and modern patterns of normal traffic. It has 49 features split to five groups namely Flow, Basic, Content, Time, and Additional generated features. This dataset contains an overall number of 257,705 records labeled whether by an attack-type or a normal label [42].

3.5. Intrusion Detection System (IDS)

The IDS aims at recognizing and detecting any infringements of the laws, aggressive activity, abuse, or abuse of computer systems. While the IDS may be extended to a server (Host-based IDS) or network (Network-based IDS), a network-based IDS analysis is oriented. Some of the popular network-based open-source IDSs include Snort, Suricata, and zeek [32]. The various types of IDS can be categorized according to the types of the analyzer and detection algorithms used. Detection algorithms can be divided into misused algorithms and behavioral algorithms. In the former situation, suspicious activity is detected depend on a string pattern that fits the predefined pattern of documented attacks stored as signatures in the database. Such detection will only identify known attacks that have been extremely reliably predefined, meaning that a new attack cannot be detected before the signature database is modified. IDS based on signatures has a very small, but false-negative rate. It also includes that each packet is connected to a serialized list of signatures. It can be a resource-intensive and sluggish process depending on the amount of the traffic and signature database. The abstraction of the flow level given in the SDN cannot be used by signature-based systems where only those flows that need moreover, analysis is to be sent to the controller [33].

3.6. Support Vector Machine (SVM) Classifier

All classification problems are false-positive reductions and an imbalance between normal and invasive interactions. In the problem of classification, an unknown pattern is assigned to a predefined class in the form of a vector according to the characteristics of the pattern. There are various technical classifications. We used SVM to detect intrusion. In our situation, we are concerned with a binary distinction, which implies a normal or invasive partnership. Classifiers SVM [34][35] are typically More advanced classifications to solve problems in binary classification and therefore fully meet our requirements. SVM considers an ideal high-dimensional hyperplane separating data with broad margins from different classes [36]. The margin is known as the total of the space from the decision boundary (hyperplane) of the near points of the two classes (support vectors). The method SVM depends on the statistical theory of learning and has an enticing capacity to generalize problems of linear and non-linear choice[35][37]. By the possibility of misclassification of an unknown pattern at randomly generated from a constant and uncertain distribution, SVM uses structural risk management rather than empirical risk minimization [34][35]. SVM calculates a hyperplane that represents the margin between the outstanding training and the class limit when the results are dimensional. If the information is not linearly divided, the instances are tracked to a large area where a splitting hyperplane may be found. The kernel function is the mechanism for defining the mapping method.

SVMs are efficient classifiers for intrusion detection with good efficiency. This can be applied to data with several apps, although the number of features has been limited to increase their performance. The mathematical handling and geometric representation of SVM are main features. This has led to a
rapid increase in interest among SVMs in recent years, showing remarkable progress in several areas[38][39][40][34][41].

4. Proposed system
The proposed system suggested the network that consisted of a server cloud, OpenDaylight controller, intrusion detection system, and SDN emulates on Mininet shown in the following Figure (2).

![Logical network used on this paper](image)

Figure 2. Logical network used on this paper

4.1. Implementation of OpenDaylight (ODL) SDN Controller
We used the six releases of the ODL controller known as carbon. We have downloaded the program from the ODL web website and mounted the device on the Ubuntu 18.04 server in the cloud domain. The Open Source program called Apache Karaf, which is used to install and allow the OpenDaylight controller to use the necessary functionality. Karaf is a scalable Open Services Gateway Initiative (OSGi) offering the materials and functions needed to run the program. OSGi is a compilation of standards for the development and implementation of modular software programs and libraries packaged in packages. Karaf helps load, continue, end, upgrade, and uninstall modules without rebooting. The OpenDaylight controller does not have any features allowed by default. In this proposed system, we have the following functions installed and activated on the control (However, many functions may be built and enabled or disabled):

- odl-restconf – Representational State (REST) as a network offering the HyperText Transmission Protocol (HTTP) programmatic mechanism for accessing data on port 8181 for HTTP queries.
- odl-l2switch-switch-ui - Provides L2 (Ethernet) forwarding across connected OpenFlow switches and support for host tracking.
- odl-ovsdb-library - Encode/decoder library for OVSDB and Hardware vTEP schema.
- odl-ovsdb-southbound-api - This feature provides the YANG models for northbound users to configure the OVSDB device. These YANG models are designed based on the OVSDB schema.
odl-ovsdb-southbound-impl - This feature is the main feature of the OVSDB Southbound plugin. This plugin handles the OVS device that supports the OVSDB schema and uses the OVSDB protocol.

odl-ovsdb-southbound-impl-rest - This feature is the wrapper feature that installs the odl-ovsdb-southbound-api & odl-ovsdb-southbound-impl feature with other required features for restconf access to provide a functional OVSDB southbound plugin.

odl-l2switch-all - The L2 Switch project provides Layer2 switch functionality.

The ODL controller used in this article is based on the google cloud framework Computing Engine. In this proposal, the Elastic Computing Cloud (VM instances) was used because it is a stable, resizable machine node and enables us to achieve and customize power in a few minutes. The Mininet simulator is also installed and configured to make a host system, servers, and Open VSwitch on the Ubuntu machine VM link with the ODL controller in the cloud. Furthermore, IDS is installed and configured on the Ubuntu VM to with Mininet equipping network traffic observation, attacks intrusion detection utilizing Network IDS by interfaces ens33, these are utilizing via Mininet switch s1. The designed device has no IP addresses such that the framework may link to the s1 OVS connection, such as the IDS, between devices. An IDS is Posted on the Virtual machine. Incoming and outgoing network traffic flow is monitored by mirroring. Communication is made through all the Open Flow switch, which is were created the Mininet machine through the use of the mirror traffic. The IDS is linked by ens33 to the Open Flow switch s1 and s2.

4.2. Mininet Implementation
This paper hosts VMware's Mininet Virtual Machine (VM). Mininet was installed on the Ubuntu 18.04 operating system. Mininet has created an emulated software-defined network using a Python script. Topo, Switch, and Controller are the three key functions of this script. Topo: This is the basic class of Mininet Topology. Provides network representation of data centers for organized multi-trees. A function has been built using Python to build a custom network. Mininet has been stopped from building a network using default values to create a network. The controller class must define the remote device and determine the name and IP addresses of the ODL device IP address in the Google Cloud. Port 6633 has been set.

The subclass (OVS) was generated using the OVSKernel Switch. There were also two network hosts and the networking tools were given. The scripts specify the network subnet to be used by the controller along with the connection between the switch and the hosts. The scripting codes the controller such that the external interface is attached to the switch after network creation. Mininet utilizes this guide to navigate the business domain Linux service.

4.3. Proposed IDS Implementation
The proposed system is shown in Figure (3). This model has several parts. The first part is the representation consists of dividing the dataset to sub dataset one for training and testing data, in that work has been used two data sets (NSL-KDD, UNSW-NB15). It is as follows:
4.3.1. **Preprocessing**

Machine Learning algorithms do not work too well for raw data processing. We need to preprocess this data before we can feed this data to an ML algorithm. In other words, some transformations need to be applied. We transform raw data into a clean preprocessing data collection. Some ML models require information in a specified format.

4.3.2. **Encoding**

Some of the scripts may be content words or numbers, for each script. Training data is usually labeled with words to make it readable. Label encoding converts word labels into numbers that allow algorithms to operate on them.

4.3.3. **Feature Engineering & Drop Object Feature**

Feature engineering technique to construct a well-defined, ML-based Network IDS-enabled dataset. Feature-as-a-Counter (FaaC) is therefore utilized as a practical solution to the problem of learning from big heterogeneous data set. It comprises the fusion and transformation of different data source and their

---

**Figure 3. Proposed Machine Learning IDS**
parameters in new variables in a specified time interval, that are only counters to the original variables as shown in figure (4). When the engineering feature is used there are original features such as (protocol type, status, and service) that are removed this method is called Drop Object Feature. For example, when converting patterns within a protocol type feature that contains the text to new features such as (Protocol_type_3pc, Protocol_type_a/n, etc...). It has been dropped the original feature, for two data sets (NSL-KDD and UNSW-NB15) for the emergence of new features derived from the original features.

4.3.4. Feature Grouping
The size of the data is large with the implementation of the algorithm support vector machine, so the results of accuracy are few. Therefore, It has been suggested to divide the types of attacks into groups and apply SVM to each group to obtain high accuracy as shown in the following table (1, and 2).

![Figure 4. The FaaC approach as a feature engineering process.](image)

| Table 1. Grouping UNSW-NB15 Dataset |
|--------------------------------------|
| Label | Normal | Fizzers | Analysis | Backdoor | Dos | exploits | generic | reconnaissance | shellcode | worms |
| No of row | 56000 | 18184 | 2000 | 1746 | 12264 | 33394 | 40000 | 10491 | 1133 | 130 |
| Gropes | G 1 | G 2 | G 3 | G 4 | G 5 | G 6 | G 7 | G 8 | G 9 |
| Normal +Attack type | 74184 | 58000 | 57746 | 68264 | 89394 | 96000 | 66491 | 57133 | 56130 |

| Table 2. Grouping NSL-KDD Dataset |
|-----------------------------------|
| Label | Normal | DoS | Probe | R2L | U2R |
| No of row | 67343 | 45927 | 11656 | 995 | 52 |
| Gropes | G 1 | G 2 | G 3 | G 4 |
| Normal +Attack type | 113270 | 78999 | 68338 | 67395 |
4.3.5. Features standardization
Features standardization is a common requirement of machine learning methods, to avoid that features with large values may weight too much on the final results. For each feature, calculate the average, subtract the mean value from the feature value, and divide the result by their standard deviation. After scaling, each feature will have a zero average, with a standard deviation of one.

4.4. Support Vector Machine Classifier
Classification is a supervised learning technique that learns the structure of the training data set consisting of input/attributes and output category. This function is used to predict the class mark of any valid input vector. The main goal of classification is to use machine learning algorithms to achieve the highest predictive accuracy. The issue of classification can be seen as an optimization issue, to find the best model for predictive relationships in the data. In addition to conventional data mining techniques such as naive Bayes, decision tree, rule induction, etc. Support Vector Machine (SVM) has become more interested in finding an acceptable solution and has been adopted on data classification issues. Most machine learning algorithms usually do not produce optimal results if their parameters are not properly tuned. It is very important to choose a powerful machine learning algorithm and to adjust its parameters to construct a highly accurate classification model. Optimization of parameters can take a lot of time if performed manually, especially if the learning algorithm has many parameters. The biggest problem with SVM is how to select the kernel function and its parameter values. Inappropriate parameter settings contribute to poor classification results. In this paper, we used a method to adjust the SVM parameter: GridSearch with cross-validation.

4.4.1. Parameter Optimization using GridSearch
The GridSearch is a thorough search based on a given hyper-parameter space subset. Hyperparameters are defined by the minimum (lower bound), maximum (upper bound), and several steps. The search grid optimizes the SVM parameters (C, γ, grade, etc.) utilizing a Cross-Validation performance metric (CV). The aim is to identify good combinations of hyperparameters so that the classifier can accurately predict unknown data. The over-fitting problem can be avoided through a cross-validation technique. To select C and γ with k-fold CV, we first divide the available data into k subsets (in most experiments we set k=10). One subset is used as test data and is tested using the other k-1 training subsets. Then we calculate the CV error with this SVM classifier split error by using the specific parameters C, γ, and other values. Various hyper-parameter value combinations are entered and one with the best cross-validation accuracy (or the lowest CV error) is selected and used to train the SVM in the entire dataset.

Moreover, it is the best solution to the class problem, and it is more suitable in this work of two classes normal and abnormal. The third section is the training and testing of the system. Training is tuning the system to detection optimal parameters, and finally, testing is the evaluation of the trained system.

5. Results
It is significant to utilize a standard benchmark when assessing the performance of our proposed model. The parameters used for NIDS are Accuracy, Precision, Sensitivity, and F-Measure. Within this test, the parameters specified are utilized to test the qualification of our model. The Confusion Matrix was used to determine the parameters to check this. Also, the Confusion Matrix includes follow up parameters correctly identified True Positive (TP) record numbers, correctly rejected True Negative (TN) record number, incorrectly identified False Positive (FP) attack number, and incorrectly rejected False Negative (FN) attack record number. The following equation is derived from the confusion matrix for our assessment parameters.

\[
\text{Accuracy (ACC): test the proportion of true traffic identification.} \\
\text{ACC} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)
\]
Sensitivity (SNS): the percentage of expected attacks against all current attacks is often referred to as recall or true positive scores.

\[
SNS = \frac{TP}{TP + FN}
\]  
(2)

False Alarm Rate (FAR) allow a fake alarm metric and is measured as:

\[
FAR = \frac{FP}{TP + FP}
\]  
(3)

F–Measure (F1): is a calculation of the accuracy of the model, taking into account its precision and sensitivity.

\[
F1 = \frac{2TP}{2TP + FP + FN}
\]  
(4)

The proposed IDS model was evaluated using a machine learning algorithm SVM with a GridSearch. The resulting view in tables (3, 4) indicates Pattern Recognition has the best performance accuracy of detecting abnormality with a 99.99% discovery rate. Moreover, SVM with a GridSearch method recorded the strongest results.

**Table 3. Accuracy of UNSW-NB15 dataset**

| criteria | Fuzzers | Analysis | Backdoor | DoS | Exploits | Generic | Reconnaissance | Shellcode | worms |
|----------|---------|----------|----------|-----|----------|----------|---------------|-----------|-------|
| TN       | 73000   | 37000    | 36997    | 37000| 36998    | 36998    | 37000         | 37000     | 37004 |
| TP       | 6062    | 677      | 584      | 4089| 11132    | 18871    | 3496          | 378       | 44    |
| FN       | 0       | 0        | 0        | 0   | 0        | 0        | 0             | 0         | 0     |
| FP       | 4       | 4        | 7        | 4   | 6        | 6        | 4             | 0         | 0     |
| FAR      | 0.01%   | 0.01%    | 0.018    | 0.01| 0.016    | 0.016    | 0.01          | 0         | 0     |
| Acc      | 99.99%  | 99.98%   | 99.98    | 99.99| 99.98    | 99.98    | 99.99         | 100       | 100   |
| F1       | 99.96%  | 99.7%    | 99.4     | 99.95| 99.97    | 99.98    | 99.94         | 100       | 100   |
| Recall   | 100%    | 100%     | 100%     | 100 | 100      | 100      | 100           | 100       | 100   |

**Table 4. Accuracy for NSL-KDD dataset**

| Criteria | DoS | Probe | R2L | U2R |
|----------|-----|-------|-----|-----|
| TN       | 9683| 9640  | 9461| 9701|
| TP       | 7422| 2382  | 2795| 45  |
| FN       | 38  | 39    | 90  | 22  |
| FP       | 28  | 71    | 250 | 10  |
| FAR      | 0.003| 0.028| 0.082| 0.18|
| Acc      | 99.61| 99.9 | 97.3 | 99.67|
| F1       | 99.55| 97.74| 94.26| 73.77|
| Recall   | 99.49| 98.38| 91.78| 67.16|

6. Conclusion
Software-Defined networks add flexibility to networking, decoupling the control plane and data plane, and removing network architectural privacy for opening and scheduling networks. Thanks to the several advantages, many businesses are switching from the existing network system to the modern SDN design. Nonetheless, SDN, as an emerging network, presents a challenge to the technological future. Safety is one of the key threats to the viability of SDN technology. Mininet has been used to build a fixed network of software. The SDN network was integrated into the real-life network with the ODL controller. Open Flow Protocol was designed and also utilized to connect with the ODL controller hosted with a virtual switch on the Mininet cloud network. The ODL controller administers the transfer of knowledge through
Mininet to the real network. A proposed classifier (SVM with modified GridSearch parameters) improves the accuracy of anomaly detection in the SDN networks. Two data sets, (NSL KDD, UNSW-NB15) were involved in the experiment. Propose work revealed that SVM with grid search was statistically significant compared to other classifiers. Furthermore, the proposed model outperformed other methods found in the literature with respect to accuracy and False Alarm Ratio metrics [43], [44]. In the proposed machine learning SDN IDS model, the detection rate was 99.8 percent of the accuracy of the attack detection.

References
[1] T. Xing, D. Huang, S. Ata, and D. Medhi, “MobiCloud: A geo-distributed mobile cloud computing platform,” in Proceedings of the 2012 8th International Conference on Network and Service Management, CNSM 2012, 2012.
[2] T. Xing et al., “Constructing a virtual networking environment in a Geo-distributed programmable layer-2 networking environment (G-PLaNE),” in IEEE International Conference on Communications, 2012, doi: 10.1109/ICC.2012.6364943.
[3] D. Huang, T. Xing, and H. Wu, “Mobile cloud computing service models: A user-centric approach,” IEEE Netw., 2013, doi: 10.1109/MNET.2013.6616109.
[4] C. S. Alliance, “Top threats to cloud computing v1.0,” White Pap., vol. 23, 2010.
[5] P. Alto, “Software-Defined Networking: The New Norm for Networks [white paper],” ONF White Pap., 2012.
[6] K. Siddique, Z. Akhtar, F. Aslam Khan, and Y. Kim, “KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research,” Computer (Long. Beach. Calif.), 2019, doi: 10.1109/MC.2018.2888764.
[7] M. M. Rathore, A. Ahmad, and A. Paul, “Real time intrusion detection system for ultra-high-speed big data environments,” J. Supercomput., 2016, doi: 10.1007/s11227-015-1615-5.
[8] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in ICISSP 2018 - Proceedings of the 4th International Conference on Information Systems Security and Privacy, 2018, doi: 10.5220/0006639801080116.
[9] Z. Li, A. L. G. Rios, G. Xu, and L. Trajković, “Machine learning techniques for classifying network anomalies and intrusions,” in Proceedings - IEEE International Symposium on Circuits and Systems, 2019, doi: 10.1109/ISCAS.2019.8702583.
[10] T. T. H. Le, Y. Kim, and H. Kim, “Network intrusion detection based on novel feature selection model and various recurrent neural networks,” Appl. Sci., 2019, doi: 10.3390/app9071392.
[11] C. G. Cordero, E. Vasilomanolakis, A. Wainakh, M. Mühlhäuser, and S. Nadjm-Tehrani, “On generating network traffic datasets with synthetic attacks for intrusion detection,” arXiv Prepr. arXiv1905.00304, 2019.
[12] C. G. Cordero, S. Hauke, M. Mühlhäuser, and M. Fischer, “Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks,” in 2016 14th Annual Conference on Privacy, Security and Trust, PST 2016, 2016, doi: 10.1109/PST.2016.7906980.
[13] E. Kabir, J. Hu, H. Wang, and G. Zhuo, “A novel statistical technique for intrusion detection systems,” Futur. Gener. Comput. Syst., vol. 79, pp. 303–318, 2018.
[14] STOLFO, S. J., et al. KDD cup 1999 dataset. UCI KDD repository. http://kdd.ics.uci.edu, 1999.
[15] K. K. Gupta, B. Nath, and R. Kotagiri, “Layered approach using conditional random fields for intrusion detection,” IEEE Trans. dependable Secur. Comput., vol. 7, no. 1, pp. 35–49, 2008.
[16] V. Hajisalem and S. Babaie, “A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection,” Comput. Networks, 2018, doi: 10.1016/j.comnet.2018.02.028.
[17] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, “Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives,” in 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018, pp. 1–
8. [18] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, “Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,” Expert Syst. Appl., vol. 67, pp. 296–303, 2017, doi: 10.1016/j.eswa.2016.09.041.
[19] C. Emmanouilidis, R. A. Koutsianamanis, and A. Tasidou, “Mobile guides: Taxonomy of architectures, context awareness, technologies and applications,” Journal of Network and Computer Applications. 2013, doi: 10.1016/j.jnca.2012.04.007.
[20] D. Armstrong and K. Djiemame, “Towards quality of service in the cloud,” in Proc. of the 25th UK Performance Engineering Workshop, 2009.
[21] OpenDayLight, “OpenDayLight.” https://www.opendaylight.org/.
[22] W. Zhou, L. Li, M. Luo, and W. Chou, “REST API design patterns for SDN northbound API,” in Proceedings - 2014 IEEE 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2014, 2014, doi: 10.1109/WAINA.2014.153.
[23] L. Li, W. Chou, W. Zhou, and M. Luo, “Design Patterns and Extensibility of REST API for Networking Applications,” IEEE Trans. Netw. Serv. Manag., 2016, doi: 10.1109/TNSM.2016.2516946.
[24] M. Brandt, R. Khondoker, R. Marx, and K. Bayarou, “Security Analysis of Software Defined Networking Protocols — OpenFlow, OF-Config and OVSDB,” Fifth IEEE Int. Conf. Commun. Electron. ICCE 2014, 2014, doi: 10.1145/2684793.2684796.
[25] J. Schonwalder, M. Bjorklund, and P. Shafer, “Network configuration management using NETCONF and YANG,” IEEE Commun. Mag., vol. 48, no. 9, pp. 166–173, 2010.
[26] Z. K. Khattak, M. Awais, and A. Iqbal, “Performance evaluation of OpenDaylight SDN controller,” in Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS, 2014, doi: 10.1109/PADSW.2014.7097868.
[27] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, “Reproducible network experiments using container-based emulation,” in CoNEXT 2012 - Proceedings of the 2012 ACM Conference on Emerging Networking Experiments and Technologies, 2012, doi: 10.1145/2413176.2413206.
[28] T.-Y. Huang, V. Jeyakumar, B. Lantz, N. Feamster, K. Winston, and A. Sivaraman, “Teaching computer networking with mininet,” in ACM SIGCOMM, 2014.
[29] J. Yan and D. Jin, “VT-mininet: Virtual-time-enabled mininet for scalable and accurate software-defined network emulation,” in Symposium on Software Defined Networking (SDN) Research, SOSR 2015, 2015, doi: 10.1145/2774993.2775012.
[30] R. K Jha, P. Kharga, I. Z. Bholebawa, S. Satyarthi, and S. Kumari, “OpenFlow technology: A journey of simulation tools,” Int. J. Comput. Netw. Inf. Secur., vol. 6, no. 11, p. 49, 2014.
[31] Zeek, the zeek network security monitor, 2020 “Zeek.” https://zeek.org/.
[32] M. Alenezi and M. J. Reed, “Methodologies for detecting DoS/DDoS attacks against network servers,” in The Seventh International Conference on Systems and Networks Communications ICSNC, 2012, pp. 92–98.
[33] M. Hussain, S. K. Wajid, A. Elzaart, and M. Berbar, “A comparison of SVM kernel functions for breast cancer detection,” in Proceedings - 2011 8th International Conference on Computer Graphics, Imaging and Visualization, CGIV 2011, 2011, doi: 10.1109/CGIV.2011.31.
[34] V. Vapnik, The nature of statistical learning theory. Springer science & business media, 2013.
[35] B. E. Boser, I. M. Guyon, and V. N. Vapnik, “Training algorithm for optimal margin classifiers,” in Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, 1992, doi: 10.1145/130385.130401.
[36] C. J. C. Burges, “A tutorial on support vector machines for pattern recognition,” Data Min. Knowl. Discov., 1998, doi: 10.1023/A:1009715923555.
[37] S. J. Horng et al., “A novel intrusion detection system based on hierarchical clustering and support vector machines,” Expert Syst. Appl., 2011, doi: 10.1016/j.eswa.2010.06.066.
[38] F. E. Heba, A. Darwish, A. E. Hassanien, and A. Abraham, “Principle components analysis and
support vector machine based intrusion detection system,” in 2010 10th international conference on intelligent systems design and applications, 2010, pp. 363–367.

[39] Z. Sun, G. Bebis, and R. Miller, “Object detection using feature subset selection,” Pattern Recognit., 2004, doi: 10.1016/j.patcog.2004.03.013.

[40] D. S. Kim, H. N. Nguyen, and J. S. Park, “Genetic algorithm to improve SVM based network intrusion detection system,” in Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2005, doi: 10.1109/AINA.2005.191.

[41] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set in Computational Intelligence for Security and Defense Applications,” Comput. Intell. Secur. Def. Appl., no. Cisda, pp. 1–6, 2009.

[42] N. Moustafa and J. Slay, “The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set,” Inf. Secur. J., vol. 25, no. 1–3, pp. 18–31, 2016, doi: 10.1080/19393555.2015.1125974.

[43] I. Syarif, A. Prugel-Bennett, and G. Wills, “SVM parameter optimization using grid search and genetic algorithm to improve classification performance,” Telkomnika (Telecommunication Comput. Electron. Control., vol. 14, no. 4, pp. 1502–1509, 2016, doi: 10.12928/TELKOMNIKA.v14i4.3956.

[44] I. S. Thaseen and C. A. Kumar, “Intrusion detection model using fusion of PCA and optimized SVM,” Proc. 2014 Int. Conf. Contemp. Comput. Informatics, IC3I 2014, no. January, pp. 879–884, 2014, doi: 10.1109/IC3I.2014.7019692.