2017 Seventh International Conference on Emerging Security Technologies (EST)

Machine Learning Based Intrusion Detection System for Software Defined Networks

Atiku Abubakar and Bernardi Pranggono*
Department of Engineering and Mathematics,
Sheffield Hallam University, Sheffield, S1 1WB, U.K.
*B.Pranggono@shu.ac.uk

***Abstract* —Software-Defined Networks (SDN) is an emerging area that promises to change the way we design, build, and operate network architecture. It tends to shift from traditional network architecture of proprietary based to open and programmable network architecture. However, this new innovative and improved technology also brings another security burden into the network architecture, with existing and emerging security threats. The network vulnerability has become more open to intruders: the focus is now shifted to a single point of failure where the central controller is a prime target. Therefore, integration of intrusion detection system (IDS) into the SDN architecture is essential to provide a network with attack countermeasure. The work designed and developed a virtual testbed that simulates the processes of the real network environment, where a star topology is created with hosts and servers connected to the OpenFlow OVS-switch. Signature-based Snort IDS is deployed for traffic monitoring and attack detection, by mirroring the traffic destine to the servers. The vulnerability assessment shows possible attacks threat exist in the network architecture and effectively contain by Snort IDS except for the few which the suggestion is made for possible mitigation. In order to provide scalable threat detection in the architecture, a flow-based IDS model is developed. A flow-based anomaly detection is implemented with machine learning to overcome the limitation of signature-based IDS. The results show positive improvement for detection of almost all the possible attacks in SDN environment with our pattern recognition of neural network for machine learning using our trained model with over 97% accuracy.**

***Keywords* — Software-defined Network; Intrusion Detection System; OpenFlow; Machine Learning; Neural Network;**

I. INTRODUCTION

Software-Defined Networking (SDN) is an emerging area that promises to change the way we design, build, and operate the networks. Shifting from the traditional network architecture of proprietary based to the open, simple, and programmable network architecture. Open networking foundation defines SDN as "an evolving architecture that is dynamic, manageable, cost-effective, and adaptable. An ideal for the high bandwidth requirement and dynamic nature of today's application. The architecture decouples the network control and forwarding functions. This is enabling the network control to become directly programmable, and allowing the underlying infrastructure to be abstracted for applications and network services" [1].

Today network has become an essential part of public infrastructures with the inception of public and private cloud computing. The traditional networking approach has become too complex. This complexity has resulted in a barrier for creating new and innovative services within a single data center, difficulties in interconnecting data centers, interconnection within enterprises, and bigger barrier in the continued growth of the Internet in general.

Furthermore, current network architecture has many limitations, which were resolved with the emergence of new SDN architecture. These include but are not limited to: inability to optimize network for WAN and Data Centre to generate more revenue and reduce expenses. With SDN more revenue can be generated by monitoring network devices and optimizing device utilization with a dynamic feature of SDN. The increase in capital and operational cost with SDN automation reduces human involvement in managing resources to a minimum which significantly reduces the cost.

The SDN comprise three-tiered architecture that is designed to simplify network management [2]:

- The Application layer: contains application that delivers services.
- The SDN Controller: the main decision-making component separated originally from data plane which facilitates automated network management.
- The Infrastructure layer: a hardware layer that requires command line interface (CLI), but it does not need a programming language, unlike other layers.

II. BACKGROUND AND RELATED WORK

The key technology advantages of SDN are network flexibility, efficiency, speedy service provisioning, and lower operation cost considering the gain over the traditional network technology. Traditional network technologies are proprietary and restricted to specific devices. SDN has the ability of been programmable, configurable and manageable. It is also open for the user to use devices from different vendors. SDN architecture is characterized by the separation of the control plane from data plane [2]. With the logically centralized control plane, the controller has the global view of the entire network where the forwarding entries are programmed based on the policies defined. This centralization can result in efficient support for traffic engineering, and maintain reliable security and policy implementation to the entire network [3].

Despite the security consideration in designing SDN architecture, the SDN environment still has security issues that need to be addressed. Some of these problems are inherited

978-1-5386-4018-0/17/$31.00 ©2017 IEEE

from traditional network environment, while some are specific to the SDN architecture [3].

The security threat has become so frequent from within, the effect of these attacks ranges from mild to critical. The security breach usually alters the credibility, integrity, or availability of hardware, software or an information resource. The attack on these components can bring considerable damage to the organization. The damages can be a loss in monetary or reputation which may lead to the total or partial collapse of the organization. Therefore, an effective measure must be put in place to avoid the damage.

Although the architecture of SDN tried to contain the security prone in the network management, but the separation of the control plane from data plane bring another form of security threat to the SDN architecture that can be found in any of it three layers: application, control, and infrastructure layer. The consequences this security prone can lead to data modification, unauthorized access to the network, data leakage, denial of service (DoS) [4]. Many of the attacks are possible due to the centralized control introduced by the SDN architecture. In [5] demonstrated the possibility of an attack gaining access to the SDN controller. Once the controller is compromised the attacker can alter the rules in the devices and deny a legitimate user access to the available resources (DoS attack). DoS attacks are not the only attacks for SDN but among the common attacks there are other attacks like port proves, vulnerability scan, man-in-the-middle (MITM), and side-channel.

Integrating an intrusion detection system (IDS) into SDN architecture is potentially one of the best approaches to build a secure SDN environment. IDS is a system purposely designed to detect and alert unauthorized or unwanted access attempts, changes, or/and restricts computer system resources [6]. The system typically detects malicious traffic and attack against the network or a single host computer.

Basically, there are two most common types of IDS: host IDS (HIDS) and network IDS (NIDS). HIDS is usually installed and run on each system or network as individual device monitoring the incoming and outgoing packet within the system or network and notifies the user or administrator if the system is under any potential or actual attack or any unusual activities detected. HIDS normally operates by taking the snapshot of the existing files and compares it with the previous snapshot of system files, with this the unauthorized activities can be identified.

On the other hand, NIDS is a system that identifies unauthorized, anomalous behavior, and attack in the network by examining network traffic and monitoring different hosts over the network environment. NIDS generally gain access to the network traffic by linking to a hub, network tap, and configured switch for port mirroring. In this work, the purpose is to implement IDS for SDN environment, therefore IDS in this work refers to the NIDS throughout the project unless it is specified otherwise.

In this work, we used signature-based detection technique and Snort in specific to implement IDS for SDN. We also develop flow-based IDS model that can provide scalable security and threat management solution using pattern recognition of neural network with machine learning.

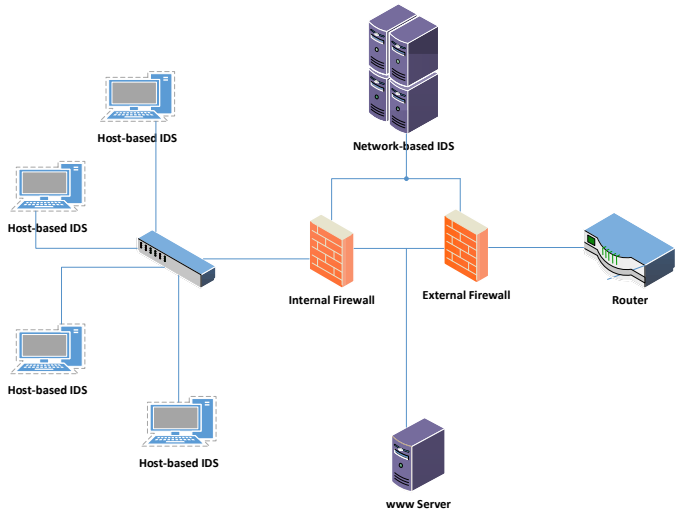

Fig. 1. Signature-based IDS

III. Experiment Design

A virtual testbed is developed where various attacks are performed by means of simulation. Initially, different attacks techniques are implemented to observe the impact of DoS, Probe, U2R, and R2L attacks on SDN environment on both the servers and normal users accessing resources on the server.

As signature-based IDS cannot be the solution to all type of attacks, it is necessary to provide alternative approaches that complement its work. A flow-based anomaly-based system is developed as an anomaly-based IDS. This is due to the nature OpenFlow protocol as the communication protocol between controller and infrastructure layer: it uses flow for identifying the network traffic, and also records its information by counters. The flow is a sequence of IP packets with common characteristics, going through monitoring point within a period of time.

The work follows two approaches to provide a solution to this problem. The first is developing a virtual testbed that mimics the real scenario and provides a solution to signature-based attacks. The second method is designing the model that will provide anomaly-based detection. This would be integrated into signature-based architecture for detection of unknown attack undetected by signature-based IDS.

A. Virtual Testbed

OpenDayLight controller (ODL) is installed and configured on Ubuntu Desktop 16.04 OS. ODL manages the Open Virtual Switches (OVS) based on OpenFlow protocol through a remote connection to be established by Mininet simulator.

The Mininet network simulator is also installed and configured to create host system, servers, and OVS on the same OS with ODL. The Metasploitable2 server is hosting four services that are left vulnerable intentionally for penetration testing purpose, while the Parrot security will be generating attack scenario on Metasploitable2.

Furthermore, Snort IDS is installed and configured on separate Ubuntu Desktop VM to provide network traffic monitoring, attacks intrusion detection by means of NIDS.

The Mininet+ODL deployed on VM with three additional interfaces eth1, eth2 and eth3, these are used by Mininet switch s1. The three interfaces are configured with no IP addresses to enable the system to provide bridging between systems connecting to the s1 ovs-switch such as Parrot Security, Metasploitable2 server and Snort IDS.

Parrot Security is deployed on the VM and installed, it has special attacking tools, the default eth0 will connect to the s1 ovs-switch in the Mininet+ODL environment through eth1 interface. The Parrot Security IP is configured to be in the same network with Metasploitable2 server. The Metasploitable2 server is deployed on the VM as a server connecting to s1 ovs-switch through eth2 interface of Mininet VM with it default eth0.

Snort IDS is deployed on Ubuntu machine VM. The incoming and outgoing network traffic flow is monitored by the Snort by means of mirroring. The communication is made through OpenFlow switch created on Mininet machine via mirror traffic. Snort is connected to OpenFlow switch s1 by eth3. Figure 2 present architecture of proposed IDS for SDN virtual testbed environment.

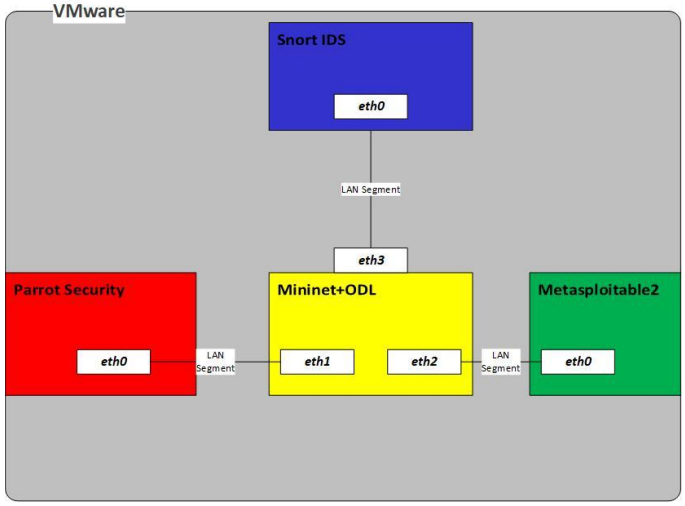

Fig. 2.. Virtual Testbed Architecture

B. Network Topology

The star topology is used for setting up laboratory network because it is easy to setup looking at the nature of the research and the combination of systems involved. Also, OVS-switch as a central hub is expected to provide optimal performance of the network traffic without overhead in providing centralize network monitoring. Therefore, failure of a single node will not affect the entire network.

Fig. 3 presents four independent VMs as their configuration seen in Figure 2 the Mininet VM is centralized. Inside Mininet, a network is created with fifteen VM hosts, five generating malicious traffic internally using manual attack procedure by attacking the server and other internal external server hosts. The ten hosts VMs generate normal or benign traffic between each other and the servers. All the hosts VM are connected to OVS-switch.

PENTMENU penetration testing tool is installed on both Parrot Security and Mininet+ODL machine with aim of attacks demonstration using created hosts for internal attacks.

The Wireshark services is on installed Mininet Simulator lunch, where the Wireshark will be monitoring the network traffic through the traffic filter any option. The purpose of using Wireshark is to observe MITM attacks on the controller.

The connection between OpenFlow ovs-switch with ODL controller is remote when creating the topology, a remote connection is specified with the loopback IP address of Ubuntu machine where ODL controller is installed. The Parrot Security, Metasploitable2 server, and Snort IDS are connected to OpenFlow ovs-switch through the Mininet+ODL VM interface eth1, eth2 and eth3 respectively.

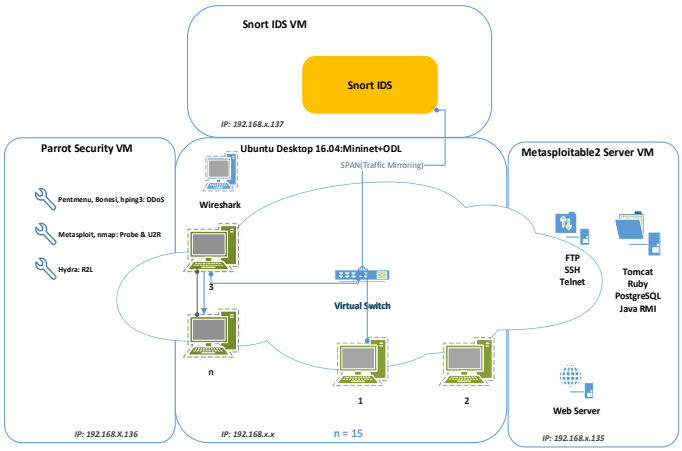

Fig. 3. Signature-based Network Topology

C. Pattern Recognition of Neural Network

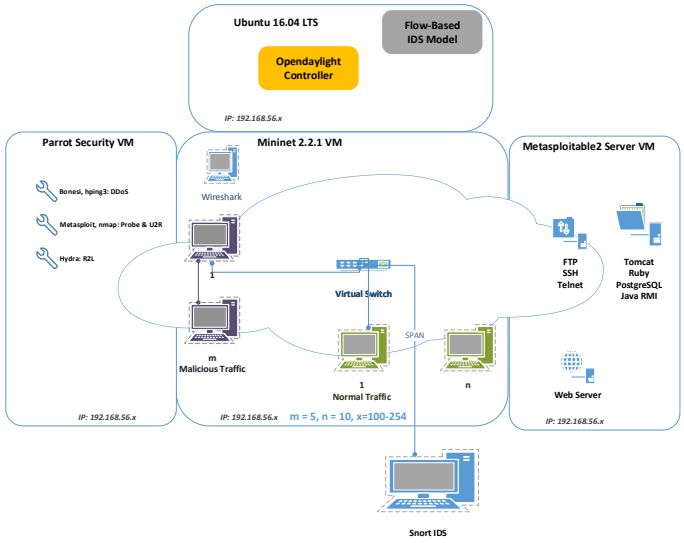

Fig. 4. Flow-based IDS Model Network Topology

In addition to the existing signature-based IDS, a Neural Network-based model is designed to be integrated into the system. This second method proposed in this work is

flow-based anomaly detection using machine learning approach to compliment the signature-based, since the signature-based cannot detect the unknown or zero-day attack. Furthermore, attack demonstration on the virtual testbed is limited to specific type of attacks under each category of attack. Therefore, a model that can detect a wide number of attacks is proposed.

The flow-based IDS model illustrates in **Fig. 4** will be implemented in the future, as a module using Restful API or Java and hosted over ODL controller. As an application layer model, the network policies of traffic flow is controlled by the application, in such a way that some rules will be imposed that will be responsible for attack detection.

Typically the flow statistic request is sent to the switch by the controller over a certain time interval. When the statistics are available on the controller, the module will used it to detect anomaly behavior in the flow. The detected anomaly traffic will be mitigated appropriately through flow modification, hence result in new network impose by the module IDS.

Pattern recognition of neural network is implemented in this model. It usually classifies inputs into a set of target categories. The network architecture consists of three layers: an input layer, hidden layer, and an output layer. Backpropagation algorithm is used to trained the network.

Backpropagation algorithm is a training method used in classification by propagation and updating the weight of a network. When an input is received from the input layer, it is passed to the next layer, then to the output layer. The output is compared with the given targets or desired output, each output result of the neuron is calculated using a function and error value at the output layer. If the output matches the target or roughly closed, then it is presented as final output, otherwise an error is fired backwards from the output layer toward previous layers until desired output is obtained.

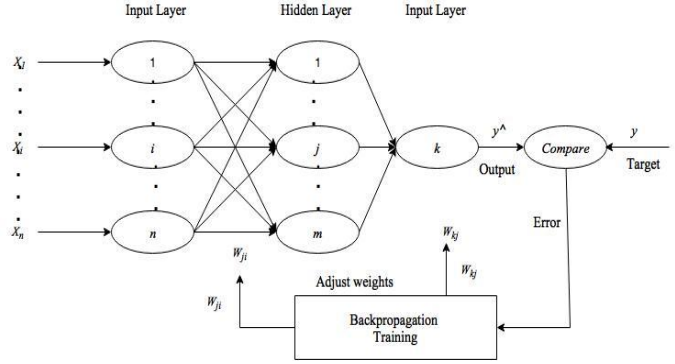

Fig. 5. Backpropagation Algorithm

The **Fig. 5** illustrates the network architecture of backpropagation algorithm uses by the model. X is the input connected to the hidden layer W weights, also the same for hidden connected to the output layer. The input variables are transformed by the nonlinear activation function, the algorithm is expressed in the following equations [7]:

$$U = \sum_{i=1}^{n} w_{ji} + w_{j0} \quad (1)$$

$$V = \sum_{j=1}^{m} w_{kj} + f_h U + w_{ko} \quad (2)$$

$$\hat{y} = f_o V \quad (3)$$

D. NSL-KDD Dataset for Training Model

The NSL-KDD dataset is used in this research to implement training and evaluation of the proposed model. The NSL-KDD dataset is the refine version of KDD-Cup 99. KDD-Cup 99 dataset is originally used in Knowledge Discovery and Data mining competition, it is the leading data mining competition in the world [8]. KDD-Cup 99 dataset has the problem of redundant record which may result in degrade the quality of inputs and cause learning algorithm to be biased to the more frequent record [9]. The NSL-KDD is proposed to solve this problem and made publicly available to the researchers. Although NSL-KDD has inherent some problem of KDD-Cup 99 but the data is still used by many researchers [10, 11]. These can be a standard benchmark for comparing our model with another detection method.

The work obtained the dataset from [12] which are partly pre-processed and categorize into four main categories: DOS, U2R, R2L, and Probes both composing the training and testing data. **TABLE** illustrates categorization of the attacks based on four categories with type of attacks in both training and testing. The attacks in Testing set that are italic and bold are only introduce in the testing stage and not available at the time. The categorization of training and testing dataset is predetermined from the original dataset source, with objective of obtaining good results in real-time. The dataset has forty-two features together with target feature; all the features are in numerical values against some that are originally nominal in order to train our model. This conversion is made from the data source.

TABLE I. ATTACK CATEGORY

Category	Training	Testing
DoS	back, land, Neptune pod, smurf, teardrop,	***apache2***, back, land, ***mailbomb***, Neptune pod, ***processtable***, smurf, teardrop, ***udpstorm***
U2R	Bufferoverflow, loadmodule, perl rootkit,	Bufferoverflow, loadmodule, perl ***ps***, rootkit, ***snmpguess***, ***sqlattack***,***worm*** ***xterm***
R2L	Spy, warezclient ftp_write, guesspasswd, imap, multihop, phf warezmaster	Spy, warezclient ftp_write, guesspasswd, ***httptunnel***, imap, multihop, ***named***, phf, ***sendmail*** ***snmpgetattack***, warezmaster,***xlock*** ***xsnoop***
Probes	Ipsweep, nmap, portsweep, satan	Ipsweep, ***mscan***, nmap, portsweep, **saint**, satan

To make the model more realistic and simplify real implementation of the module on ODL controller, specific features are selected which is commonly obtainable in the SDN environment. Seven features were selected from the existing forty-one features and illustrated in **TABLE** .

TABLE II. FEATURE SELECTION

Feature	Description
duration	Length (number of seconds) of the connection
protocol_type	Type of protocol such as TCP, UDP, etc.
service	Network service on the destination, such as HTTP, telnet, ssh, etc.
src_bytes	Number of data bytes from source to destination
dst_bytes	Number of bytes from the destination to source
count	Number of connections to the same host as the current connection in the past two seconds
srv_count	Number of connections to the same service as the current connection in the past two seconds

E. Evaluation Matrix and Procedure

In evaluating the performance of our proposed model, it is important to use a standard benchmark for evaluation criteria. Accuracy (ACC), Precision (P), Sensitivity (SNS), and F-Measure (F1-score) are commonly used parameters in performance evaluation criteria for NIDS models [10]. In this experiment, the specified parameters are used in evaluating the performance of our model. To achieve this we used the confusion matrix to calculate the parameters. Moreover, the confusion matrix contains the following parameters: True Positive (TP) number of attack record correctly identified, True Negative (TN) number of attack record correctly rejected, False Positive (FP) number of attack record incorrectly identified, and False Negative (FN) number of attack record incorrectly rejected. The following equation derived from confusion matrix to obtain our evaluation parameters [11].

Accuracy (ACC): measures the percentage of true detection over the total traffic trace.

$$\mathrm{ACC} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}} \tag{4}$$

Sensitivity (SNS): also call recall or true positive rate measures the percentage of predicted attacks against all the attacks presented.

$$\mathrm{SNS} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{5}$$

Precision (P): measures the number of attacks predicted by IDS that are actual attacks.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+\ FP}} \tag{6}$$

F-Measure (F1): is a measure of test accuracy in the model by considering Precision and Sensitivity.

$$\mathrm{F1} = \frac{2\mathrm{TP}}{2\mathrm{TP+FP+FN}} \tag{7}$$

IV. RESULTS AND DISCUSSION

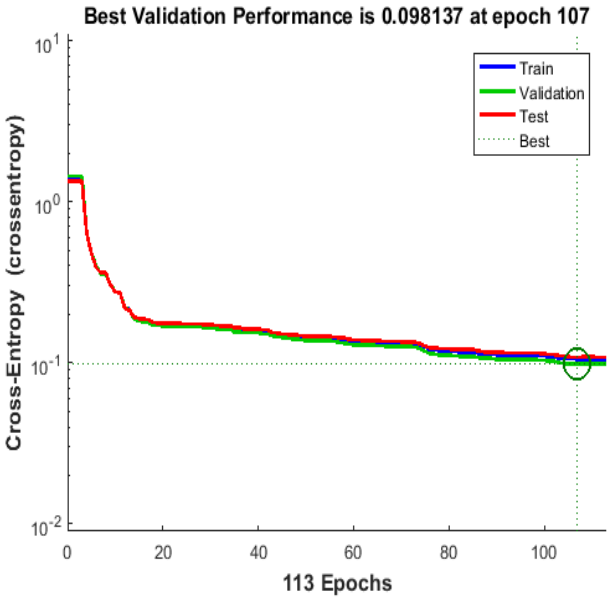

Fig. 6. Performance Measurement Graph

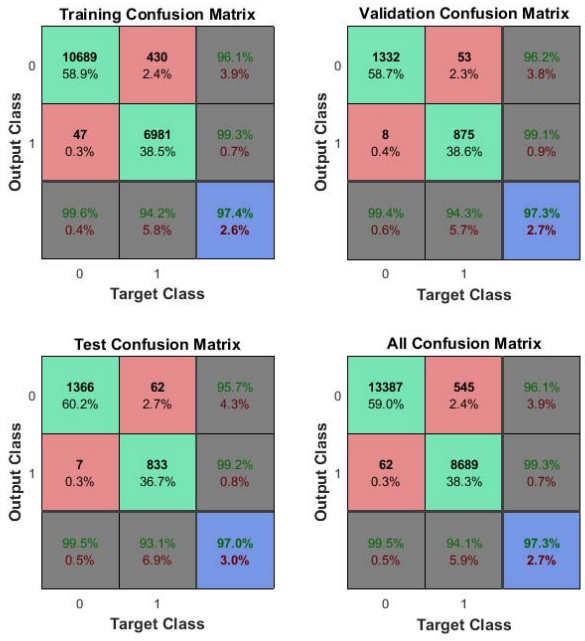

Fig. 7. Accuracy measure from Confusion Matrix

The performance of the model validation is best on 0.098137 at 107 iterations as shown in **Fig. 6**. The algorithm normally stops training when the performance of the training data stops improving, by doing that the best number of an epoch

is selected. The validation set is used to measure this performance because of its ability to generalize network model and serve as the basis for the evaluation.

The Performance Measurement Graph **Fig. 6** illustrates the results obtained for the performance evaluation of the designed IDS model using confusion matrix (**Fig. 7**). The model shows high detection accuracy of 97.4% in detecting attacks with training set and 97% on testing set, while overall accuracy is 97.4%. The Figure 8 shows model performance on the dataset on graph of True Positive rate (sensitivity) and False Positive rate (specificity), ROC curve is a plot on Sensitivity against Specificity, for the three portions of data on the training set the curve. The overall ROC formed a curve on the upper-left corner of the graph and this shows the optimal performance of the model at that point. Forming curve at the upper-left corner indicate the performance of model prediction is very good.

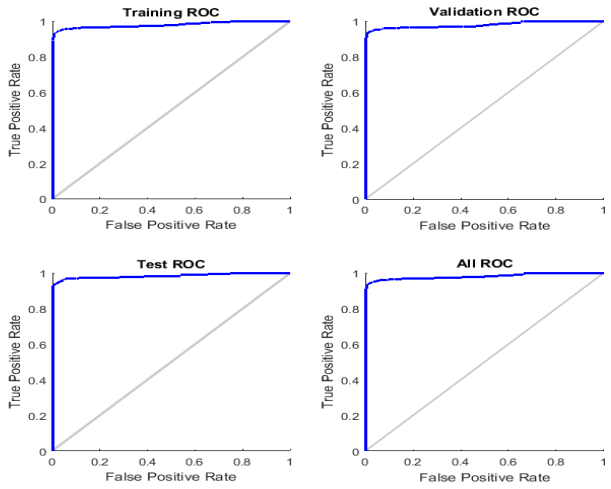

Fig. 8. ROC Curve

A. Evaluation

The performance of this model IDS is evaluated based on other neural network type such as Curve Fitting and Time Series. The results shown in **Table** indicate Pattern Recognition has better performance accuracy of detecting anomaly with 97.3% detection rate. Fitting Curve has 89.5% accuracy, it initially has less performance but with weight initialization and re-training the performance in detection accuracy is improved. Moreover, Time Series Neural Network method recorded the poorest result, it takes longer time in training, this also makes retraining very difficult. During the training, it takes at least have an hour to complete training, hence retrain in several times is difficult.

Table III. Comparison of Neural Network Performance Accuracy

Neural Network Type	**Detection Accuracy (%)**
Fitting Curve	89.5
Pattern Recognition	***97.3***
Time Series	33

V. Conclusion

Software Defined Networks as an emerging technology bring innovation into the networking, with decoupling of control plane and the data plane, removing proprietary in the network architecture to open and programmable network. Due to the numerous advantage of this architecture, many companies are shifting from the traditional network architecture to new SDN architecture. However, SDN as a new technology has arising issues that pose a challenge to the futuret of the technology. Security is one of the main issue that threatens the future of SDN technology.

The paper present machine learning (Neural Network) based intrusion detection for SDN. The model IDS are built on the existing signature-based IDS architecture as flow-based IDS to detect anomaly-based attacks in the SDN environment. The Pattern Recognition is used in this paper due to its performance accuracy rate as compared with the other type of neural network model.

References

[1] OpenNetworkingFroundation. (2017, 06/2017). Available: https://www.opennetworking.org/

[2] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* vol. 103, pp. 14-76, 2015.

[3] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A Survey of Security in Software Defined Networks," *Communications Surveys & Tutorials, IEEE,* vol. PP, pp. 1-1, 2015.

[4] S. Sezer, S. Scott-Hayward, P. K. Chouhan, B. Fraser, D. Lake, J. Finnegan, *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *Communications Magazine, IEEE,* vol. 51, pp. 36-43, 2013.

[5] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," presented at the Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, Hong Kong, China, 2013.

[6] B. Pranggono, K. McLaughlin, Y. Yang, and S. Sezer, "Intrusion Detection Systems for Critical Infrastructure," in *The State of the Art in Intrusion Prevention and Detection*, A.-S. K. Pathan, Ed., ed: CRC Press, 2014, pp. 115-138.

[7] B. Fakhim, A. Hassani, A. Rashidi, and P. Ghodousi, "Predicting the Impact of Multiwalled Carbon Nanotubes on the Cement Hydration Products and Durability of Cementitious Matrix Using Artificial Neural Network Modeling Technique," *The Scientific World Journal,* vol. 2013, p. 103713, 2013.

[8] S. Hettich and S. D. Bay, "The UCI KDD Archive [http://kdd.ics.uci.edu]," University of California, Irvine, C. A.

[9] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," presented at the Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications, Ottawa, Ontario, Canada, 2009.

[10] P. Manandhar, "A Practical Approach to Anomaly-based Intrusion Detection System by Outlier Mining in Network Traffic," *Masdar Institute of Science and Technology,* 2014.

[11] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258-263.

[12] F. Hendrik. (07/2017). *NSLKDD-Dataset*. Available: https://github.com/FransHBotes/NSLKDD-Dataset