VOL. 21, NO. 3, 2022, 23-27
https://elektrika.utm.my
ISSN 0128-4428

ELEKTRIKA
Journal of Electrical Engineering

Intrusion Detection System (IDS) Accuracy Testing for Software Defined Network Internet of Things (SDN-IOT) Testbed

Sharifah H. S. Ariffin[*], **Jia Le Chong, Nurul Mu'azzah Abdul Latif, Nik Noordini Nik Abd Malik, Rashidah@Siti Saedah Arsat, Muhammad Ariff Baharudin, Sharifah Kamilah Syed-Yusof** and **Kamaluddin M. Yusof**

Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.

[*]Corresponding author: shafizah@utm.my

Abstract: Intrusion detection system (IDS) are considered as one of the best solutions for network security as it can detect intrusion and alert the network administrator on possible intrusions. However there are possible false alert that could cause unnecessary trigger of the network to the administrator. This paper provides a proof of concept of the accuracy test of an intrusion detection system (IDS) using software defined network IoT platform. The testbed uses UNSW-NB15 dataset that feeds the testbed and the traffic are mirror in a Ryu Controller that is installed with Snort IDS to monitor any DDoS attacks. For proof of concept false positive and false negative tests are run to ensure that the IDS are well configured. The experiment shows that the SDN-IoT platform with Snort IDS is accurate in both false positive and false negative test.

Keywords: Intrusion Detection System, Software Defined Network (SDN), Internet of Things (IoT).

Article History: received 21 February 2022; accepted 3 December 2022; published 22 December 2022.

1. INTRODUCTION

With recent interest and progress in the development of internet and communication technologies over the last decade, network security has emerged as a vital research domain. Intrusion detection system (IDS) is deployed to ensure the security of the network and all its associated assets within cyberspace. There are also many studies in the security of Internet of Things (IoT) network [1,2,3] where enormous devices are easily connected and able to communicate to each other. When the large volumes of data generated by IoT are considered, it is obvious that the traditional wireless network does not satisfy the network users nor the network requirement. 5G and beyond networks rely on software defined network (SDN) and network function virtualization (NFV) for resource management it is the key enabler for future's ubiquitous IoT [4]. Therefore, it is particularly important to design an intrusion detection model that guarantees the security, integration and reliability of the IoT. The focus of this work is to provide the accuracy testing for IDS installed in SDN-IoT platform to ensure the testbed is able to capture DDoS attacks as it happens.

This paper is organized as follows: Section 2 present the types of intrusion detection system commonly used by network. Section 3 presents the accuracy test that include the false positive and false negative test and hypothesis. Section 4 provide the SDN-IoT testbed configuration and set up. Section 5 consists of the results from the testbed experimental. Section 6, we conclude on the testbed experimental and configuration.

2. INTRUSION DETECTION SYSTEM (IDS)

There are several type of IDS include host based IDS, network-based IDS, Signature IDS and Anomaly-based IDS.

Host based IDS (HIDS): runs in the host system and monitors the network activities. It keeps track of the incoming and outgoing packets, and alert the administrator in case of any miscellaneous activity held in the network. HIDS analyses not only traffic but also system calls, running processes, file system changes, communication between processes and application logs. Anonymous software allows machines to be updated automatically and change lines of control etc. is an example of a host-based intrusion detection system.

Network Based IDS (NIDS): is mainly deployed on network nodes, which is capable of listening to collecting data on the shared network segment in real time, so as to analyse suspicious phenomena. There are two types of NIDS: offline NIDS and online NIDS. Offline NIDS is a non-real time system that analyses audit events after the event and checks the intrusion activities. The Online NIDS is a real-time online detection system which includes real-time network packet analysis and real-time host audit analysis [5,6].

Signature based IDS (SIDS): utilize pattern matching technique to find a known attack. Matching methods are used to find a previous intrusion, where when an intrusion signature matches the signature of a previous intrusion that is already exist in the signature database, an alarm signal is triggered. The hosts logs are inspected to find sequences of commands or actions which have previously been identified. SIDS is also known as Knowledge-based Detection or Misuse detection [7]. Studies methods of SIDS are created as a state machine, formal language string pattern and semantic conditions [8-10].

Anomaly-based IDS (AIDS): is created using machine learning, statistical based or knowledge based methods. Any significant deviation between the observed behaviour and the model is regarded as an anomaly. This technique works on fact that malicious behaviour is different from typical user behaviour. There are two phases in the development of AIDS: the training phase and the testing phase. AIDs trigger a danger signal when the examined behaviour deviates from normal behaviour [11-13].

2.1 Snort Intrusion Detection System

In this paper Snort IDS is used as signature based IDS and it has the abilities to let users set their local rules. In the testbed, the constructed attack will be DDoS SYN flooding attack while the local rule will be configured by collecting the metrics and parameters from the observation of the IoT network traffic. For the experiment, the testbed uses an open source dataset which is UNSW-NB15 dataset [14]. In the SDN-IoT platform, the controller has been chosen as the target of the attacker. By flooding huge traffic towards the controller, it will attack the network and cause system malfunction. IDS should be able to detect the attack before the controller is malfunction due to the attack. In this project, the DDoS SYN flooding attack will be carried out repetitively every 2 hours in a day in order to collect the time taken for the controller malfunction as listed in Table 1. The time taken for the controller malfunction will be the metrics for the IDS to set their detection time for the attack. The detection time will be set less than the time taken for controller malfunction in order to ensure the IDS can protect the network by detecting the attack right before the network being down. From the Table 1, the minimum period the IDS should react to the attack before the controller malfunction is less than 6.19 seconds. Thus the time taken for the IDS to react to the attack is set as 5 seconds, which is 1 second earlier before controller malfunction. Figure 1 shows the flow of the testbed configuration for the accuracy test to be conducted. DDoS detection time is determine using Table 1, types of data packets is observe in the Wireshark display and the accuracy test is run to ensure the IDS is working well. Snort local rule uses data from the local network traffic to ensure workability of the IDS installed.

3. IDS ACCURACY TESTING

False positive (FP) test and false negative (FN) test have been carried out to study the accuracy of the IDS based on the local rules' configuration. The IDS is allowed to run and observe the traffic for several days. The activities of the IDS during the test will be logged into log file for analysis. Throughout the test, the IDS can achieve highest accuracy by giving null percentage of false positive rate and false negative rate.

Table 1. Average time taken for the controller malfunction

Starting Time	**Time Taken for the controller malfunction, N (s)**			**Average (s)**
	1	**2**	**3**	
0:00	6.57	7.58	5.98	6.71
2:00	6.34	6.89	7.02	6.75
4:00	6.02	6.58	6.49	6.36
6:00	6.85	7.32	6.48	6.88
8:00	5.73	6.45	6.38	6.19
10:00	43.45	40.87	38.56	40.96
12:00	6.59	6.47	6.98	6.68
14:00	9.47	9.86	10.68	10
16:00	50.78	55.43	59.64	55.28
18:00	>60	>60	>60	>60
20:00	30.84	28.79	31.64	30.42
22:00	6.75	6.42	6.21	6.46

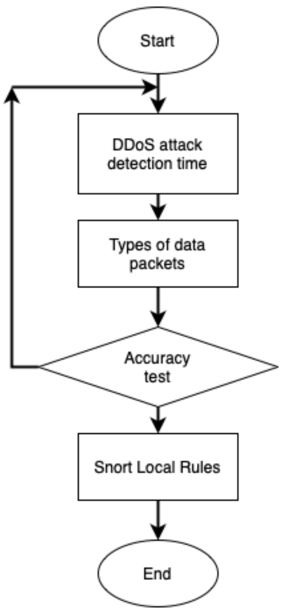


Figure 1. Flow of Snort local rule configuration

False positive test is carried out to make sure that the IDS will not provide false alert. For example, to understand False Positive (FP), the IDS is supervising a normal IoT traffic and there is no attack occurring. However, IDS mistakenly detected an abnormal traffic behaviour and report it as an attack.

Before carrying out this test, some hypothesis is set:

1. If the IDS is not configured, IDS will not be able to detect any malicious traffic occurred, hence no report will be generated.

2. If IDS is over configured, IDS supervise the normal traffic and make report on traffic although there is no malicious traffic occurred.

3. If IDS is well configured, IDS will supervise the traffic, but no report will be generated since it is normal traffic (i.e., no attack was launched).

Hence, to carry out this test, SDN-IoT testbed need to be set up. The UNSW NB15 IoT traffic dataset is used as the real time normal IoT traffic environment. IDS will supervise the traffic and its activities will be logged into log file for further analysis. Firstly, the IDS is not configured with those local settings and allowed to run for several days. The activities of the IDS are logged into log file, and we find out that there is no report generated. Hence, it proves our first hypothesis. Secondly, we over configure the IDS with the parameter (eg: count =30 in 5 seconds) and repeat the same steps as above. From the log file, we find out that the IDS keep reporting the malicious traffic although it is normal traffic. Hence, it proves the second hypothesis. Thirdly the IDS is well configured and repeat the steps. From the result, there is no report generated and proves the third hypothesis. Equation (1) shows the calculation of the false positive rate

$$False\ positive\ rate = \frac{FP}{FP+TN} \quad (1)$$

where,
FP = false positive event, TN = True negative event.

False negative test is carried out to ensure that the IDS will report the issue on time without missing any malicious traffic. For example, to understand False Negative (FN), the IDS is supervising IoT traffic. An attack has suddenly occurred, but IDS unable to detect the abnormal traffic behaviour and did not report the issue. The IDS will supervise the traffic and its activities will be logged into log file for further analysis. Firstly, the IDS is not configured with the local settings. Similar hypotheses are used for false negative test. IDS is allowed to run for a day and attack is launched at different hours (as shown in Table 1) and we find out that there is no report generated. Hence, it proves the first hypothesis since IDS does not recognize the attack yet. Secondly, we configure the IDS to be more sensitive with the parameter (e.g.: count =200 in 5 seconds) and repeat the same steps as above. From the log file, we found out that the IDS sends report of the malicious traffic. Hence, it proves the second hypothesis. Thirdly, the IDS is well configured and repeat the steps. From the results, the IDS report accurately once attack has occurred and proves our third hypothesis. Equation (2) shows the calculation of the false negative rate:

$$False\ negative\ rate = \frac{FN}{FN+TP} \quad (2)$$

where,
FN = false negative event, TP = True positive event

4. SDN-IOT TESTBED SET UP

For SDN-IoT testbed set up, we need to do network configuration of the centralized controller, connecting the SDN switches and setting up the SDN wireless interface by using embedded devices such as Raspberry Pi and Zodiac Fx. In this paper, we will focus on setting up an SDN-IoT testbed by using Raspberry Pi and Zodiac Fx. Figure 2 illustrates the hardware set-up for SDN-IoT testbed. Raspberry Pi A is connected to the Port 1 of the Zodiac Fx, it will act as the background IoT traffic generator. Raspberry Pi B is connected to the Port 2 of Zodiac Fx. It acts as the wireless access point to provide wireless connections for IoT applications. Raspberry Pi C which acts as the Ryu controller and Snort IDS is connected to Port 4 of Zodiac Fx and to a monitoring display. At the same time, Raspberry Pi C is also connected to Port 3 of Zodiac Fx by using mirror porting technique so IDS on Raspberry Pi C can observe and monitor the whole traffic. Raspberry Pi D is connected with a wireless dongle in order to connect to the wireless access point (Raspberry Pi B) wirelessly. It acts as the attacker bot to construct DDoS attack.

Zodiac Fx is an OpenFlow switch which is designed in smaller size and suitable used on a desk rather than in a data center. User can develop SDN applications using real traffic from the hardware by using Zodiac Fx switch. Ryu controller and Snort IDS have been implemented into Raspberry Pi C. Ryu controller will handle the network by managing the flow control to the switches via southbound APIs and the applications via northbound APIs. The Snort IDS will in charge of monitoring the event occurred in the network via mirror porting from the OpenFlow switch. It will alert the Ryu controller via UNIX socket6 in the same Raspberry pi once it detects DDoS event occurred.

Connection of Rasberry Pi (RP) to Zodiac FX (ZFX): Connect RP C to the Zodiac FX switch (as shown in Figure 2) The configuration is shown in Figure 3. Log into the RP then edit the file *"/etc/network/interfaces"* to add or modify the IP address on the interface connecting to ZFX. A nano editor will be opened and go to the end of the file to key in RPZ1 algorithm (shown in Figure 3 b). After that, restart the network service and then verify the connectivity to ZFX.

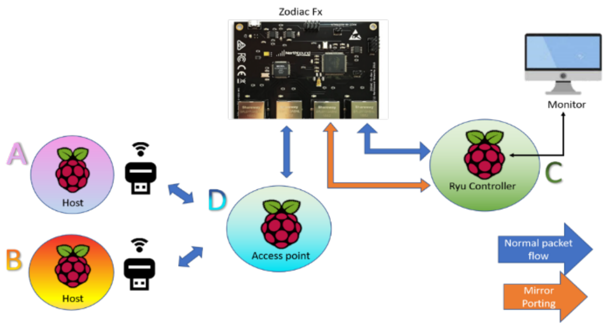
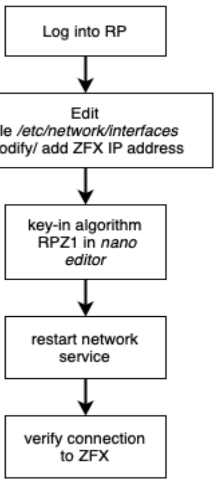


Figure 2. SDN-IoT Hardware testbed layout

SDN Controller configuration: After RP C is connected to ZFX, a configuration need to be done to ensure RP C is connected to the SDN controller. This configuration flow is shown in Figure 4 a). Verify that the Ryu controller is install properly and once Ryu controller is installed, start the controller. You will be able to observe the output via terminal windows or Wireshark display. In order to observe the whole traffic, mirror porting need to be set up. Then go back to "/etc/network/interfaces" and add RPZ2 algorithm (as shown in Figure 4 b).


Log into RP
Edit file /etc/network/interfaces Modify/ add ZFX IP address
key-in algorithm RPZ1 in nano editor
restart network service
verify connection to ZFX

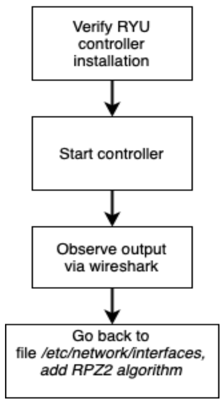
a) RP to ZFX configuration flow

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.0.1.8
netmask 255.255.255.0
gateway 10.0.1.99
```

b) RPZ1 Algorithm

Figure 3. Configuration and algorithm for RP to ZFX connection


Verify RYU controller installation
Start controller
Observe output via wireshark
Go back to file /etc/network/interfaces, add RPZ2 algorithm

a) SDN controller configuration flow

```
$ sudo nano /etc/network/interfaces

auto eth1
iface eth1 inet manual
up ifconfig eth1 promisc up
down ifconfig eth1 promisc down
```

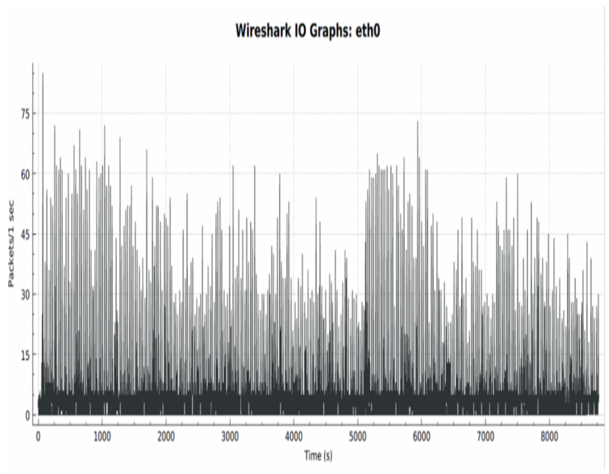
b) RPZ2 algorithm

Figure 4. SDN controller configuration flow and algorithm

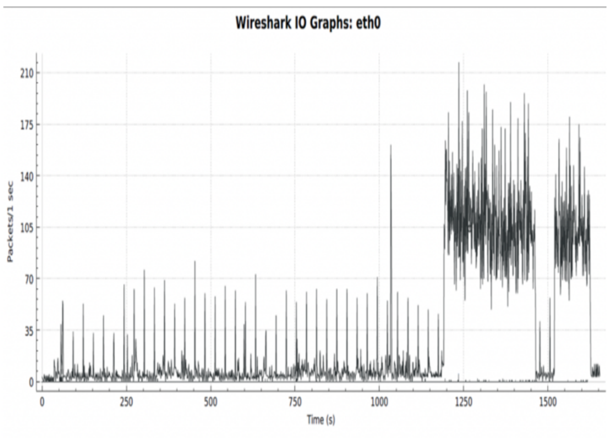
5. RESULTS AND DISCUSSION

The RP A in Figure 2 is used to generate the normal traffic in the network. Figure 5 a) shows the normal traffic in the SDN-IoT testbed of UNSW-NB15 IoT dataset. *tcpreplay* command is used to replay the IoT data while the controller will record the data using Wireshark packet sniffer. RP D is used to generate DDoS attack using hping3 command. And traffic with DDoS is shown in Figure 5 b) with average 100 packet per seconds (shown by the dotted line), where a traffic behaviour is obviously different from the pattern in a) with average 40 packet per seconds (shown by the dotted line).

This accuracy test has been done to ensure the proof of concept that the SDN-IoT IDS in the testbed is functioning using the hypothesis. The testing in this experiment was done by observe the false positive or false negative issue in the SDN-IoT testbed. From the experiment and calculation, both FP and FN show 0. Hence, throughout the test, the IDS able to achieve highest accuracy by giving null percentage of false positive rate and false negative rate. Figure 6 shows the activities of IDS recorded in the Log file.

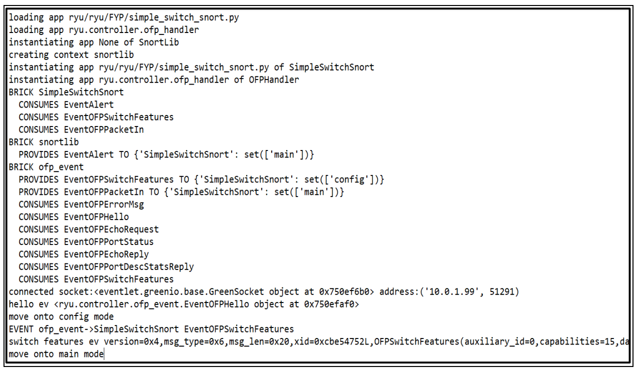


a) Normal Traffic



b) DDoS attacked traffic

Figure 5. Testbed normal and attacked traffic using UNSW IoT dataset via Wireshark display



```
loading app ryu/ryu/FYP/simple_switch_snort.py
loading app ryu.controller.ofp_handler
instantiating app None of SnortLib
creating context snortlib
instantiating app ryu/ryu/FYP/simple_switch_snort.py of SimpleSwitchSnort
instantiating app ryu.controller.ofp_handler of OFPHandler
BRICK SimpleSwitchSnort
  CONSUMES EventAlert
  CONSUMES EventOFPSwitchFeatures
  CONSUMES EventOFPPacketIn
BRICK snortlib
  PROVIDES EventAlert TO {'SimpleSwitchSnort': set(['main'])}
BRICK ofp_event
  PROVIDES EventOFPSwitchFeatures TO {'SimpleSwitchSnort': set(['config'])}
  PROVIDES EventOFPPacketIn TO {'SimpleSwitchSnort': set(['main'])}
  CONSUMES EventOFPErrorMsg
  CONSUMES EventOFPHello
  CONSUMES EventOFPEchoRequest
  CONSUMES EventOFPPortStatus
  CONSUMES EventOFPEchoReply
  CONSUMES EventOFPPortDescStatsReply
  CONSUMES EventOFPSwitchFeatures
connected socket:<eventlet.greenio.base.GreenSocket object at 0x750ef6b0> address:('10.0.1.99', 51291)
hello ev <ryu.controller.ofp_event.EventOFPHello object at 0x750efaf0>
move onto config mode
EVENT ofp_event->SimpleSwitchSnort EventOFPSwitchFeatures
switch features ev version=0x4,msg_type=0x6,msg_len=0x20,xid=0xcbe54752L,OFPSwitchFeatures(auxiliary_id=0,capabilities=15,da
move onto main mode
```

Figure 6. Activities of IDS recorded in Log file

6. CONCLUSION

This paper presents the set up configuration for SDN-IoT testbed to provide accuracy testing platform for the Snort Intrusion detection system (IDS). To ensure that the IDS in any platform work as it should be, network administrator need to ensure that IDS do not provide unnecessary trigger to the network. The accuracy testing has been conducted both false positive and false negative test taking account the hypotheses of the configuration. Both accuracy test had shown null percentage and this shows that the snort IDS is working well and the network are well configured.

ACKNOWLEDGMENT

The authors would like to thank all those had contributed to the paper. The author wishes to express gratitude to the Universiti Teknologi Malaysia under the Ministry of Higher Education Malaysia (MOHE), for providing the facilities for the experiment. The authors would like to thank the school of Electrical Engineering for financially supporting the work of this project.

REFERENCES

[1] E. Benkhelifa., T. Welsh., and W. Hamouda. “A Critical Review of Practices and Challenges in Intrusion Detection System for IoT: Toward Universal and Resilient Systems”, IEEE Comm Surveys & Tutorials, 2018, vol.20, no. 4, pp3496-3509.

[2] A. Munshi., N. Ayadh. and N. A. Almalki. “ DDoS Attack on IoT Devices”, Int. Conf. On Computer Applications & Information Security (ICCAIS), 2020.

[3] R. Arthi. and S. Krishnaveni. “Design and Development of IoT Testbed with DDoS Attack for Cyber Security Research”, Int. Conf on Signal Processing & Comm (ICPSC), May, 2021

[4] K. S. Alper and A. Pelin. “Explainable Security in SDN-Based IoT Network,” in Sensor 2020, 20, MDPI

[5] S. Pontarelli., G. Bianchi., and S. Teofili., “Traffic-aware Design of A High-Speed FPGA Network Intrusion Detection System, IEEE Trans. Comput., vol 62, no 11, pp2322-2334, 2013.

[6] A. Abraham., C. Grosan. and C. Martin-Vide. “Evolutionary Design of Intrusion Detection Programs”, Int. J. Netw. Secur., vol. 4, no. 3, pp328-339, Jan 2010.

[7] C. Modi., D. Patel., B. Borisaniya., H. Patel., A. Patel., M. Rajarajan., “A Survey of intrusion Detection Technique in Cloud”, Journal Network Computer Applied, 36, (1), pp 42-57, 2013.

[8] C. R. Meiners., J. Patel., E. Norige., E. Torng., and A. X. Liu. “Fast Regular Expression Matching Using Small TCAMs for network Intrusion Detection and Prevention Systems”, Proceedings of 19th USENIX conference on Security, Washington DC, 2010.

[9] C. Lin., and Y. D. Lin. “ A Hybrid Algorithm of Backward Hashing and Automation Racking for Virus Scanning”, IEEE Trans Computer 60, (4), pp594-601, 2011.

[10] Symantec, “Internet Security Treat Report”, vol 22., 2017

[11] A. L. Buczak., and E. Guven. “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection”, IEEE Communication Surveys Tutorial, 18, (1), pp1153-1176, 2016.

[12] A. Meshram., and C. Haas. “Anomaly Detection in Industrial Networks using Machine Learning: A roadmap”, Machine Learning for Cyber Physical Systems, 2016, pp65-72., 2017.

[13] O. Can., and O. K. Sahingoz. “ A survey of Intrusion Detection System in Wireless Sensor Network, International Conference on Modeling Simulation and Applied Optimization (ICMSOA), pp 1-6, 2015.

[14] Zoghi, Zainab and G. Serpen. “UNSW-NB15 Computer Security Dataset: Analysis Through Visualization”, arXiV preprint arXiV:2101.05067, 2021.