The Development of an Internet of Things (IoT) Network Traffic Dataset with Simulated Attack Data

Deris Stiawan[1], Dimas Wahyudi[1], Tri Wanda Septian[1], Mohd Yazid Idris[2], Rahmat Budiarto[3]*

[1] Faculty of Computer Science, Universitas Sriwijaya, Indonesia
[2] Faculty of Computing, Universiti Teknologi Malaysia, Malaysia
[3] College of Computer Science and Information Technology, Al Baha University, Saudi Arabia
deris@unsri.ac.id, mail.dimaswahyudi@gmail.com, twseptian@unsri.ac.id, yazid@utm.my, rahmat@bu.edu.sa

Abstract

Due to the complexity and multifaceted nature of Internet of Things (IoT) networks/systems, researchers in the field of IoT network security complain about the rareness of real life-based datasets and the limitation of heterogeneous of communication protocols used in the datasets. There are a number of datasets publicly available such as DARPA, Twente, ISCX2012, ADFA, CIC-IDS2017, CSE-CIC-IDS2018, CIC-DDOS2019, MQTT-IoT-IDS-2020, and UNSW-NB15 that have been used by researchers to evaluate performance of the Intrusion Detection Systems (IDSs), nevertheless, the datasets creation are not based on real-life scenarios and rely only on one communication protocol. This paper produces a dataset that is created using real-life scenarios. The data are captured from an IoT test-bed network consists of six sensors running IEEE 802.11 (WiFi) and IEEE 802.15.4 (ZigBee) communication protocols and considering normal as well as attacks traffics. Furthermore, the robustness of the dataset for recognizing the types of data traffics is evaluated using Intrusion Detection Engine (IDE) with Naïve String Matching. The experiments on dataset robustness show promising results, i.e.: Accuracy level of 99.92%, Precision of 100%, False Positive Rate (FPR) of 0, and False Negative Rate (FPR) of 0.0869.

Keywords: IDS Dataset, Internet of Things (IoT), IEEE 802.11 (WiFi), IEEE 802.15.4 (ZigBee)

1 Introduction

The Internet of Things (IoT) integrates several identifying devices, sensing, and communication technologies, such as Radio Frequency Identification (RFID) tags, sensors, actuators, cell phones, and others cable/wireless devices via unique addressing schemes with standard communication protocols [1-2]. A growing number of heteregeneous IoT devices and services and their more widespread distribution have made IoT security more complex and challenging [3].

A main challenge in an IoT system implementation is security issues, such as privacy, authorization, verification, access control, system configuration, storage and information management [4-5]. An Intrusion Detection System (IDS) plays an important role in the network security defense by warning the security administrator about malicious behaviors such as intrusions, attacks, and malware. The IDS domain has evolved over the years with better systems. However, many researchers yet have difficulty in finding comprehensive and valid datasets for testing and evaluating their proposed techniques [6-7]. A useful dataset must meet certain criteria. There are 11 characteristics for a comprehensive and valid IDS dataset, i.e.: diversity of attacks, anonymity, available protocols, complete catches, complete interactions, complete network configurations, complete traffic, feature sets, heterogeneity, labeling, and metadata [8].

Our contribution is the creation of a heterogenous IoT network/system dataset for various end devices (Arduino, Raspberry Pi, WeMos D1, etc.), sensors (DHT-11, DHT-22, FC-28, K-0135, MQ-2, etc.), communication protocols, and different types of denial of service (DoS) and distributed denial of service (DDoS) attacks.

2 Problem Formulation

The proposed dataset will be used to test and evaluate detection techniques against an intrusion. There are several datasets which have been used in IDS research, however some of these dataset do not support research domain in the IoT network security field [9-10]. Here, we have listed some of these datasets which were used in testing the simulation and the IDS validation, in order to show the actual needs on comprehensive datasets.

Defense Advanced Research Projects Agency (DARPA): This dataset, which was developed by MIT's Lincoln Laboratory, contains normal and attacks traffic data in a simulated experimental environment. Services considered in this dataset include: Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP), Telnet, Internet Relay Chat (IRC), and Simple Network Management Protocol (SNMP). The dataset contains attacks such as DoS, guess passwords, buffer overflow, remote FTP, "synchronize" (SYN) flood, Nmap, and rootkit. This dataset does not represent real-world network traffic. Simulation attacks are carried out over several networks. The attack process is grouped into 5 attack phases as follows. First, a scanning process is carried out on the network to find target host, after

*Corresponding Author: Deris Stiawan; E-mail: deris@unsri.ac.id
DOI: 10.53106/160792642023032402013

finding a loophole, a breach is made to entering the host by exploiting the Solaris sadmind vulnerability. Then, installing the mstream DDoS trojan software, and launching a DDoS attack on a server outside the network from the compromised host. In addition, the dataset is obsolete for an effective IDS evaluation of a modern network, both in terms of attack types and network infrastructure. Unfortunately, the dataset does not have real traffic history.

Twente (University of Twente, 2009): This dataset includes three services (Open Secure Socket Shell (OpenSSH), Apache web server and Proftp) using authentication/identification on port 113 and data are extracted from a honeypot network by *Netflow*. It includes some simultaneous network traffics such as authentication/identification, Internet Control Message Protocol (ICMP), and IRC traffic. Additionally, this dataset contains some unrecognized and non-correlated warning traffic. The simulation process is carried out by running a honeypot installed on a Citrix XenServer virtual machine. The selection of a honeypot as a virtual host is due to the flexibility to install, configure and restore virtual machines if compromised. In addition, compromised virtual machines can be saved for further analysis. Various scenarios are possible, in terms of number of virtual machines, operating systems and software. The experimental setup consists of a single virtual machine, namely Debian Etch 4.0. To keep the setup simple, controllable, and realistic, it does not rely on honeypot software, but configure the host manually [11]. The attack data volume is very limited.

ISCX2012 (University of New Brunswick, 2012): The attack simulation was carried out using a single layer 3 switch (Omniswitch 6850). A virtual Local Area Network is configured on the Switch to capture almost all communication between all machines. This dataset has two profiles. The alpha profile performs a variety of multi-stage attack scenarios. The beta profile, which is the normal traffic generator, creates realistic network traffic for the Hypertext Transfer Protocol (HTTP), SMTP, SSH, Internet Message Access Protocol (IMAP), Post Office Protocol 3 (POP3) and FTP with a full packet payload [12]. Nevertheless, attack simulation traffic data are not based on real situation.

ADFA (Universitas New South Wales, 2013): Simulation of attacks was carried out using a payload *meterpreter* encoded with *Metasploit*. then the *TikiWiki* Vulnerability was used to upload a copy of the Java Meterpreter payload, which initiates a reverse TCP connection to the attacker's computer during launching the attack. Once the shell is created, various actions are taken on the host system, including local privilege escalation, attempts to access the shadow password file, and persistent back door installation. This dataset includes training and validation data with 10 attacks per vector. Services that are considered for testing environment include: SSH, MySql, dan FTP. It contains FTP and SSH password brute force, Java-based Meterpreter, add new superuser, Linux Meterpreter payload, and C100 webshell attacks traffics. Furthermore, the dataset was created on Linux and Windows operating systems for evaluation purpose using system call-based IDS [13]. The dataset has only a few types of attacks, even some attacks have similar behavior with normal traffic.

CIC-DDOS2019: The dataset is created using abstracting behavior of 25 users that utilizing HTTP, FTP, HTTPS, SSH, and SMTP protocols. The dataset has different reflective modem DDoS attacks traffics such as: Lightweight Directory Access Protocol (LDAP), PortMap, Microsoft SQL (MSSQL), NetBIOS, Domain Name Service (DNS), User Datagram Protocol (UDP), Network Time Protocol (NTP), UDP-Lag, SYN, dan SNMP. These attacks traffics data were created by executing 12 DDoS attacks during the testing, including: SNMP, LDAP, SYN, NTP, MSSQL, NetBIOS, SSDP, WebDDoS, DNS, UDP, UDP-Lag and Trivial File Transfer Protocol (TFTP). While 7 DDoS attacks including: PortScan, MSSQL, NetBIOS, UDP, UDP-Lag LDAP and SYN are executed during the training phase. Simulation is done by implementing two networks, namely Attack-Network and Victim-Network. Victim-Network is a high security infrastructure with firewalls, routers, switches and commonly used operating systems required to provide normal traffic on each PC. Attack-Network is a completely separate infrastructure that executes different types of DDoS attacks [12]. Nonetheless, the data captured from simulated attacks are not based on real life statistics.

UNSW-NB15: The dataset was created by utilizing IXIA PerfectStorm tool at Cyber Range Lab, University of New South Wales (UNSW) Canberra. It has 9 types of attacks, namely: Backdoors, Exploits, Fuzzers, Reconnaissance, Analysis, Generic, DoS, Shellcode and Worms. HTTP, FTP, SSH and DNS are the services that supported by the dataset. The simulation is done by configuring *IXIA* traffic generated by three virtual servers. Server 1 and Server 3 are configured for normal traffic deployment. Server 2 creates abnormal/malicious activity traffic, establishes intercommunication between servers and acquires public and private network traffic. The servers are connected to the host via two routers. The routers are connected to a firewall device that is configured to pass all traffic whether normal or abnormal. The *tcpdump* tool is installed on router 1 to capture the *Pcap* files of the simulated uptime. The main purpose of this whole simulation process is to capture normal or abnormal traffic, originating from IXIA tools and dispersed among network nodes (for example, servers and clients) [14]. The entries in the dataset are also not based on real life statistics.

CIC-IDS2017: Attack simulation is made by implementing two networks, namely Attack-Network and Victim-Network. Victim-Network is a high security infrastructure with Firewalls, Routers, switches and running common operating systems (Windows, Linux and Macintosh), which are required to provide normal traffic on each PC. Attack-Network is a completely separate infrastructure designed by routers and switches and four PCs, which have Kali and Windows 8.1 operating systems. The Victim Network consists of three servers, one firewall, two switches, and ten PCs connected to each other by a domain controller (DC) and active directory. Also, one port on the Victim-Network main switch has been configured as a mirror port and fully captures all sending and receiving traffic to the network. The dataset contains Brute Force Attack, Heartbleed, Botnet, Dos, DDos, Web and Infiltration attacks, captured from a set of traffic behavior of 25 users running SSH, HTTP, FTP, HTTPS and SMTP service protocols. Packet size of the

protocol, number of packets per flow, certain pattern in the payload, payload size and time to request of the protocol are considered as features of the entry data in the dataset [9].

CSE-CIC-IDS2018: The attack simulation configuration is similar to the creation of CIC-IDS2017, except the Victim Network consists of three servers, one firewall, two switches, and ten PCs connected to each other by a domain controller (DC) and active directory. Also, one port on the Victim-Network main switch has been configured as a mirror port and fully captures all sending and receiving traffic to the network. In this dataset, packet size, number of packets per flow, certain pattern in the payload, payload size and distribution of requesting time for a protocol are considered as features. SSH, HTTPS, POP3, SMTP, IMAP, HTTP and FTP protocols are simulated in the testing environment. The available attack's type include: Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks and insider network infiltration [9].

MQTT-IoT-IDS-2020: This dataset is created using Message Queuing Telemetry Transport (MQTT) protocol. It has 4 types of attacks, i.e.: Aggressive Scan (Scan A), User Datagram Protocol (UDP) Scan (Scan U), Sparta SSH brute-force (Sparta) and MQTT Brute-Force attack (MQTT BF). The dataset is generated using a simulated MQTT network architecture. The network consists of 12 MQTT sensors, a broker, an engine to simulate camera feeds, and an attacker. During normal operation, the 12 sensors send random messages using the MQTT "Publish" command. Message lengths differ between sensors to simulate different usage scenarios. Message content is randomly generated. Camera feeds are simulated using the Visible Light Communication (VLC) media player which uses UDP streams. To further simulate a realistic scenario each network emulator drops packets by 0.2%, 1%, and 0.13%. During the recording of the four attack scenarios, normal background operations are left in action. The operating systems of different devices are as follows; Tiny Core Linux for sensors, Ubuntu for cameras & camera feed servers, and lastly, Kali Linux for launching attacks [15]. Some attacks have similar behavior with normal traffic.

3 Problem Solution

This research activity was conducted in the Computer Network and Information Security (COMNETS) laboratory in the Faculty of Computer Science, Sriwijaya University, Indonesia. The creation of the built-in system consisted of several steps which included topology design, identifying hardware requirements, identifying software requirements, system installation and configuration, as well as testing scenarios.

3.1 Topology

This research uses a star network topology. It consisted of six sensor nodes (air humidity, gas, fire, soil humidity, water level sensor and two units of heat sensor), two middleware devices, and one server. All the sensor nodes and middleware devices were connected in one network to the server via a wireless router by IP addresses using the Dynamic Host Configuration Protocol (DHCP).

Two communication protocols, IEEE 802.11 (*WiFi*) and the first version of IEEE 802.15.4 (ZigBee) are deployed. Data capturing was performed with three scenarios: (1) normal traffic data; (2) TCP FIN flood and zbassocflood attack traffic data; (3) mixed normal + TCP FIN flood and zbassocflood attack traffic data. The traffic flow during the data capturing is illustrated in Figure 1. Black line represents the normal traffic, light Blue represents traffics that connect direct to wireless router and will be captured by sniffing computer. Line with Red color represents the attacks traffic. Details of the traffic simulation process are as follow.

1. Node 1 consists of WeMos D1 based on ESP8266 as a WiFi module and a DHT 22 sensor as a temperature and humidity sensor. WeMoS D1 is connected to the DHT 22 sensor to form Node 1, which functions to sense temperature and humidity parameters. WeMos D1 collects and sends the sensing data to the server via WiFi.
2. Node 2 consists of WeMos D1 based on ESP8266 as a WiFi module and a soil moisture sensor. WeMoS D1 is connected to the soil moisture sensor to form Node 2 that functions to sense the parameters of the soil moisture level. WeMos D1 collects and sends the sensing data to the server via WiFi.
3. Node 3 consists of WeMos D1 based on ESP8266 as WiFi module and MQ2 sensor. WeMoS D1 is connected to the MQ2 sensor to form Node3, which functions to sense smoke and gas parameters. We-Mos D1 collects and sends the sensing data to the server via WiFi.
4. Node 4 consists of WeMos D1 based on ESP8266 as a WiFi module and a water level sensor. WeMos D1 collects and sends the sensing data to the server via WiFi. Furthermore, the data will be stored into the database server and displayed on a web in the forms of tables and figures.
5. Node 5 consists of Arduino UNO, I/O Expansion Shield V7, XBee series 1 radio module and DHT 11 sensor. Each device is connected to form Node 5. The DHT 11 sensor on the node serves to sense the temperature and humidity parameters. Arduino UNO collects and sends the sensing data to the middleware via the XBee series 1 radio module.
6. Node 6 has similar configuration with Node 5, the difference only on the sensor type, i.e.: temperature and humidity sensor. The sensing data on the middle-ware is then sent via WiFi and stored into the data-base server.
7. Middleware is in the form of a Raspberry Pi model 3 device which is integrated with the XBee Explorer Universal Serial Bus (USB) device. The middleware receives and accommodates sensor data from Node 5 and Node 6 via the XBee radio module, then will send the data to the database server via WiFi. In this case the middleware serves to connect devices with different delivery methods. The difference lies in the type of sensor used, where Node 5 uses an XBee series 1 radio module and a DHT 11 sensor, while Node 6 uses an XBee series 1 radio module and a

DHT22 sensor. The process starts from the sensor function by sensing the temperature and humidity parameters, sensor data is collected and sent to the middleware via the XBee radio module, then the middleware will collect and send sensor data to the database server via WiFi.

8. Server monitoring is used to display the data sent by each node in the form of a web display. Data from each node will be sent in real time to the database server and displayed in the form of a web view. In this case the server uses MySQL.

The created dataset in this study has benefit compared to the existing datasets, as it is the only dataset of IoT network that runs IEEE 802.1 protocol as well as the first version of IEEE 802.15.4 protocol. Table 1 summarizes the comparison among the datasets. Each node consisted of an active sensor that collected data in real time. The data delivery process was categorized according to the communication protocol used by the node and middleware device. The detailed system specifications can be seen in Table 2.

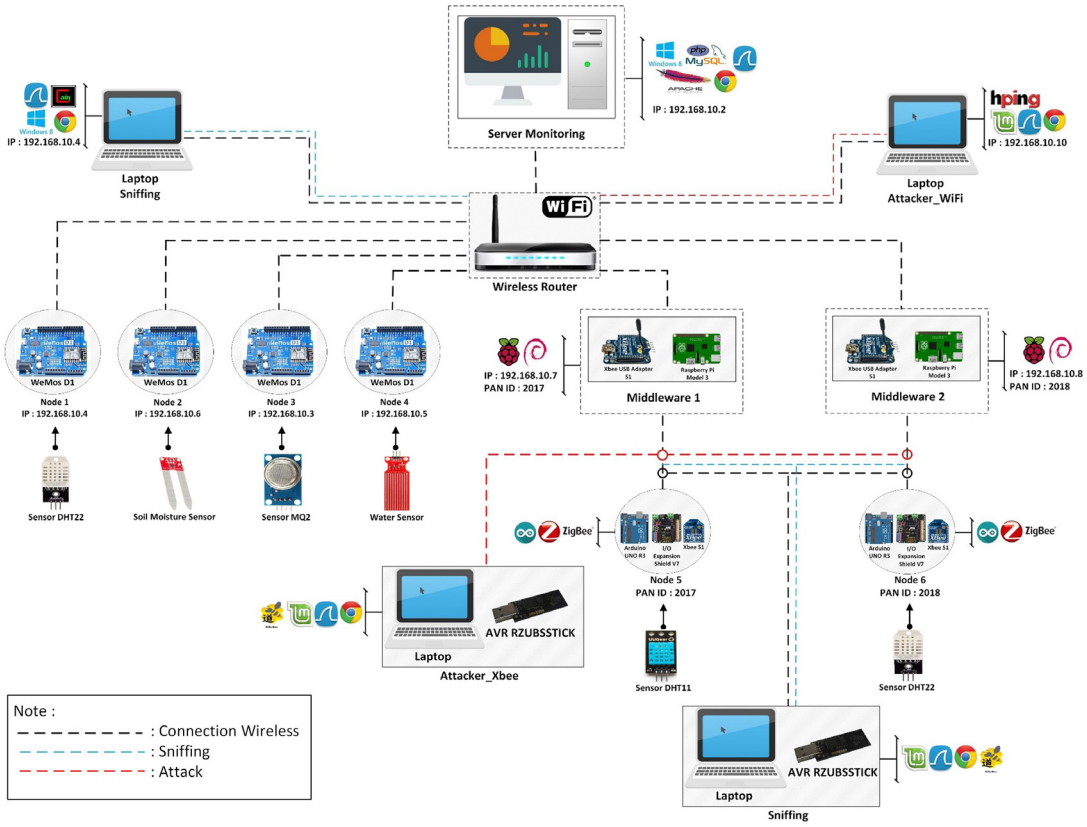

Figure 1. The network topology for creating the dataset

Table 1. Comparison protocols/Service dataset

No.	Dataset	Protocols/Service	Network
1	DARPA	SMTP, FTP, Telnet, IRC, and SNMP	Conventional
2	Twente	SSH, HTTP ,and PROFTP	Conventional
3	ISCX2012	HTTP, SMTP, SSH, IMAP, POP3 and FTP	Conventional
4	ADFA	FTP, SSH, MySql	Conventional
5	CIC-DDOS2019	HTTP, FTP, HTTPS, SSH, and SMTP	Conventional
6	UNSW-NB15	HTTP, FTP, SSH, DNS	Conventional
7	CIC-IDS2017	SSH, HTTP, FTP, HTTPS, and SMTP	Conventional
8	CSE-CIC-IDS2018	SSH, HTTPS, POP3, SMTP, IMAP, HTTP, dan FTP	Conventional
9	MQTT-IoT-IDS-2020	MQTT	IoT
10	**This Dataset**	**IEEE 802.11 (WiFi) and IEEE 802.15.4 (ZigBee)**	**IoT**

Table 2. System specifications

Device			Communication protocol
Node 1	-	WeMos D1 WiFi UNO ESP8266	IEEE 802.11 (WiFi)
	-	Temp./Humidity (DHT 22) Sensor	
Node 2	-	WeMos D1 WiFi UNO ESP8266	IEEE 802.11 (WiFi)
	-	Soil Moisture (FC-28) Sensor	
Node 3	-	WeMos D1 WiFi UNO ESP8266	IEEE 802.11 (WiFi)
	-	Gas (MQ2) Sensor	
Node 4	-	WeMos D1 WiFi UNO ESP8266	IEEE 802.11 (WiFi)
	-	Water Level (K-0135) Sensor	
Node 5	-	Arduino UNO Microcontroller	IEEE 802.15.4 (ZigBee)
	-	XBee Series 1 Wireless Module	
	-	Temp./Humidity (DHT 11) Sensor	
Node 6	-	Arduino UNO Microcontroller	IEEE 802.15.4 (ZigBee)
	-	XBee Series 1 Wireless Module	
	-	Temp./Humidity (DHT 22) Sensor	
Middleware 1-2	-	Raspberry Pi 3 Microcontroller	IEEE 802.11 (WiFi)
	-	XBee Series 1 Wireless Module	IEEE 802.15.4 (ZigBee)
Server	-	Personal Computer	IEEE 802.11 (WiFi)

3.2 Scenario

The dataset is created using twelve (12) scenarios consisting of normal data, attack data, and the combination of normal and attack data. For the experiment, each scenario has a duration of five minutes. The use of five minutes interval is based on research work by Maciá-Fernández et al. [16] that mentioned to capture DDoS attack traffic in lower level can be done during the interval of 3 to 10 minutes. This study takes the middle time interval, i.e.: 5 minutes.

In creating this dataset, we used three different DoS/DDoS attack profiles. Table 3 displays the scenarios for creating the attack traffic data.

UDP Flood: This attack uses an IP address spoof to perform multiple attacks simultaneously, sending large amounts of User Datagram Protocol (UDP) packets at specified time intervals directed at random ports of the target [17].

RST/FIN Flood: This attack sends a large number of false "reset" (RST) or "finish" (FIN) packets which are not included in any session in the server session tables. An RST or FIN DoS attack will drain or consume the victim's resources (memory, CPU, etc.) to locate and match the packets which are entered into an existing session. This results in a decrease in performance, and the service on the attacked server is not maximized [18-19].

Zbassocflood/association flooding: This attack sends a large number of association request packets on a device attach to the network. The attack results in high network traffic and large resource used by the victim. This is an insider attack, which occurs when an attacker joins into the network using a stored parameter that matches with the Personal Area Network Identifier (PAN ID) and channel on which the network operates [20-21].

Table 3. Scenario for creating attack traffic data

Scenario	Information	Duration (Minutes)	Attack
1	Send normal data packets in the form of sensor data to the server	5	*Node, Middleware*
2	Attack with TCP FIN flood and UDP flood on the server	5	*Server*
3	Scenarios 1 and 2 performed at the same time	5	*Node, Middleware, Server*
4	Nodes 1–4 send normal data packets in the form of sensor data to the server	5	*Node*
5	Attack with TCP FIN flood and UDP flood completely on nodes 1–4	5	*Node*
6	Scenarios 4 and 5 performed at the same time	5	*Node*
7	Middleware 1 and 2 sending normal packet data in the form of sensor data to the server	5	*Middleware*
8	TCP FIN flood and UDP flood completely to the middleware 1–2	5	*Middleware*
9	Scenario 7 and 8 performed at the same time	5	*Middleware*
10	Nodes 5–6 sending normal packet data in the form of sensor data to the middleware 1–2	5	*Middleware*
11	Zbassocflood attack completely on nodes 5–6	5	*Node*
12	Scenario 10 and 11 perform at the same time	5	*Node*

4 Dataset

The process of capturing the data took approximately 75 minutes. The 12 test scenarios were performed resulting in 15 pcap files/datasets of the normal, attack, and normal-attack combinations. Table 4 displays the results of the captured traffic data during the experiments.

The created datasets have different sizes. The different in the size is due to differences in the number of data packets and the different data packet delivery formats between WiFi and XBee. No specific technique was used for data labeling. As an experiment run certain scenario, communication protocol and on certain network segment, then the category of the captured traffic data are automatically recognized. For example, the traffic data in the first dataset/file in Table 4 are captured traffic in the server attached in the network that running WiFi communication protocol and using Scenario 1 (pumping normal traffic). Thus, all captured traffics are normal traffic, labeled as normal and then the file is named with normal-server.pcap, indicating that the dataset/file contains normal traffic. Same procedure of labeling was implemented to other created datasets. Therefore, a dataset with just "normal" in its filename contains data packets with normal traffic; a dataset with an "attack" in its filename describes the entire data packet traffic as FIN flood and UDP flood attack on WiFi communication or Zbassocflood on XBee communication; a dataset with "normalxattack" describes a combination of normal data packet traffic and attack data packet traffic.

Table 5 is the result of the packet frequencies based on protocol type which were used in the FIN flood attack dataset. There are six types listed: User Datagram Protocol (UDP), Transmission Control Protocol (TCP), Address Resolution Protocol (ARP), Hypertext Transfer Protocol (HTTP), Internet Control Message Protocol (ICMP), and a category for all others.

Table 4. Created dataset result

No	Dataset/Label	Communication protocol	Size	Description	Scenario
1	server-normal.pcap	IEEE 802.11 (WiFi)	5.8 MB	Server	1
2	server-attack.pcap	IEEE 802.11 (WiFi)	655.5 MB	Server	2
3	server-normalxattack.pcap	IEEE 802.11 (WiFi)	805.3 MB	Server	3
4	mid1-normal.pcap	IEEE 802.11 (WiFi)	263.4 KB	Middleware 1	7
5	mid1-attack.pcap	IEEE 802.11 (WiFi)	82.4 MB	Middleware 1	8
6	mid1-normalxattack.pcap	IEEE 802.11 (WiFi)	83.6 MB	Middleware 1	9
7	mid2-normal.pcap	IEEE 802.11 (WiFi)	284.4 KB	Middleware 2	7
8	mid2-attack.pcap	IEEE 802.11 (WiFi)	109.2 MB	Middleware 2	8
9	mid2-normalxattack.pcap	IEEE 802.11 (WiFi)	112.5 MB	Middleware 2	9
10	node-wifi-normal.pcap	IEEE 802.11 (WiFi)	1.1 MB	Nodes 1–4	4
11	node-wifi -attack.pcap	IEEE 802.11 (WiFi)	168.4 MB	Nodes 1–4	5
12	node-wifi-normalxattack.pcap	IEEE 802.11 (WiFi)	170.3 MB	Nodes 1–4	6
13	node-xbee-normal.pcap	IEEE 802.15.4 (ZigBee)	22.2 KB	Nodes 5–6	10
14	node-xbee-attack.pcap	IEEE 802.15.4 (ZigBee)	701.4 KB	Nodes 5–6	11
15	node-xbee-normalxattack.pcap	IEEE 802.15.4 (ZigBee)	808.9 KB	Nodes 5–6	12

Table 5. Normal and TCP attack packet frequencies

Dataset/Label	Protocol						Total
	UDP	TCP	ARP	HTTP	ICMP	Other	
server-normal.pcap	106	8856	457	2872	500	1	12792
server-attack.pcap	114	3132331	626	2322	0	0	3135393
server-normalxattack.pcap	114	3706556	639	2368	0	4	3709681
mid1-normal.pcap	99	1112	264	264	0	0	1739
mid1-attack.pcap	100	1174242	609	108	0	0	1175059
mid1- normalxattack.pcap	112	1190448	642	118	0	0	1191320
mid2-normal.pcap	119	1082	641	260	0	0	2102
mid2-attack.pcap	116	1554771	650	169	0	0	1555706
mid1- normalxattack.pcap	121	1602024	672	221	0	0	1603038
node-normal-wifi.pcap	124	5438	241	2003	0	0	7806
node-attack-wifi.pcap	232	2396917	1564	707	0	0	2399420
node-normalxattack.pcap	274	2423933	1640	752	0	0	2426599

The TCP packets were most prevalent, because the FIN flood attack packets use the TCP, and this is coupled with the sensors sending data and other normal data from the nodes to the server using the TCP simultaneously.

Table 6 shows the traffic in terms of frequencies of data packets on the normal XBee dataset, XBee attack dataset, and XBee normal-attack dataset. The highest traffic in terms of numbers of packets was those using the IEEE 802.15.4 protocol, followed by ZigBee, Internet Protocol version 6 (IPv6), LwMesh, and 6LowPAN. This shows an increase in the percentage of IEEE 802.15.4 packet traffic from the normal XBee dataset to the XBee attack dataset and from the normal-attack XBee dataset to the Zbassocflood attack scenario.

Table 7 and Table 8 is the result of the packet frequencies based on protocol type which were used in the UDP attack, i.e.: UDP, TCP, ARP, HTTP, ICMP and by a category for all other protocols.

The dataset retrieval process was performed on the server network interface side, so that in this UDP dataset, there would be sufficient TCP packets. The packets include: actual sensor data being sent to the server via TCP-HTTP.

Table 6. Normal and UDP attack packet frequencies

Dataset/Label	Protocol					Total
	ZigBee	IEEE 802.15.4	LwMesh	IPv6	6LowPAN	
node-xbee- normal.pcap	9	527	17	4	11	568
node-xbee-attack.pcap	11	19374	11	9	21	19426
node-xbee normalxattack.pcap	6	22355	33	1	46	22441

Table 7. Packet Frequencies during the UDP attack to the server

Dataset/Label	Protocol						Total
	UDP	TCP	ARP	HTTP	ICMP	Other	
server-normal.pcap	608	20329	1476	6981	0	6	29400
server-attack.pcap	9620943	17870	1584	5076	6	28	9645507
server- normalxattack.pcap	9772125	17632	1971	5102	6	19	9796855

Table 8. Old packet frequencies during the UDP attack

Dataset/Label	Protocol						Total
	UDP	TCP	ARP	HTTP	ICMP	Other	
pcserver- combine-1.pcap	3270682	5991	501	1723	2	8	3278907
pcserver- combine-2.pcap	3054444	5596	722	1608	2	7	3062379
pcserver- combine-3.pcap	3446999	6045	748	1771	2	4	3455569
pcserver-normal-1.pcap	207	7089	482	2445	0	2	10225
pcserver-normal-2.pcap	213	6613	540	2270	0	4	9640
pcserver-normal-3.pcap	188	6627	454	2266	0	0	9535
pcserver-attack-1.pcap	3122817	6190	460	1722	0	1	3131190
pcserver-attack-2.pcap	3417550	6092	582	1759	2	7	3425992
pcserver-attack-3.pcap	3080576	5588	542	1595	4	20	3088325

5 Analysis

An analysis was conducted in the form of a preprocessing testing stage by comparing the normal and attack datasets.

Feature extraction process was done offline using TShark tool, then convert the extracted files into Comma Separated Values (CSV) format. The extracted features are displayed in Table 9 for the Xbee dataset while Table 10 is for WiFi dataset. This step aims to facilitate in reading and recognizing normal packet patterns and attack packet patterns. Later on, for the dataset evaluation purpose, feature selection process was conducted using rule based method.

Next, experiments on dataset evaluation are performed. Intrusion Detection Engine (IDE) Naïve String matching approach is used for the evaluation process, where Naïve String Matching algorithm is implemented on the IDE. Besides, pattern recognition approaches are also carried out, i.e.: recognizing signatures (known patterns) via deep inspection analysis on raw data in pcap format for attack from these preprocessing results, not all of the attributes are used as features for the detection process on the IDS system, only the relevant attributes or features will be selected for the classification process. Traffic data packets and testing experiments using Snort as IDE. In addition, correlation among extracted features, Snort alert log and pcap raw are investigated. This investigation is considered as analysis engine for confirming the information regarding the attack traffic patterns in the raw data (pcap) files with the attack that defined as rules. The patterns of TCP FIN flood and zbassocflood attack traffic are defined as rules, as follows

finflood_rules = {ip.ttl: 64, ip.hdr_len:20, ip.len: 40, tcp. flags: F, window: 512, tcp.hdr_len: 20}.

zbassocflood_rules = {frame.len: 48, wpan.frame_type: 0x03, wpan.src_addr_mode: 0x03, wpan.src_pan: 0xffff, wpan. dst16: 0x00, wpan.cmd: 0x01, data.len: 1}

Table 9. Preprocessing result o the Xbee dataset

No	Feature	No	Feature	No	Feature
1	frame.encap_type	23	ppi.80211-common.chan.flags.ofdm	45	wpan.frame_type
2	frame.time	23	ppi.80211-common.chan.flags.2ghz	46	wpan.security
3	frame.offset_shift	25	ppi.80211-common.chan.flags.5ghz	47	wpan.pending
4	frame.time_epoch	26	ppi.80211-common.chan.flags.passive	48	wpan.ack_request
5	frame.time_delta	27	ppi.80211-common.chan.flags.dynamic	49	wpan.fcf
6	frame.time_delta_displayed	28	ppi.80211-common.chan.flags.gfsk	50	wpan.seqno_suppression
7	frame.time_relative	29	ppi.80211-common.fhss.hopset	51	wpan.ie_present
8	frame.number	30	ppi.80211-common.fhss.pattern	52	wpan.dst_addr_mode
9	frame.len	31	ppi.80211-common.dbm.antsignal	53	wpan.version
10	frame.cap_len	32	ppi.80211-common.dbm.antnoise	54	wpan.src_addr_mode
11	frame.marked	33	ppi.80211-common.tsft	55	wpan.seq_no
12	frame.ignored	34	ppi.80211-common.flags	56	wpan.dst_pan
13	frame.protocols	35	ppi.80211-common.flags.fcs	57	wpan.dst16
14	ppi.version	36	ppi.80211-common.flags.tsft	58	wpan.src16
15	ppi.flags	37	ppi.80211-common.flags.fcs-invalid	59	wpan.dst64
16	ppi.flags.alignment	38	ppi.80211-common.flags.phy-err	60	wpan.src64
17	ppi.flags.reserved	39	ppi.80211-common.chan.freq	61	wpan.fcs
18	ppi.length	40	ppi.80211-common.chan.flags	62	wpan.fcs_ok
19	ppi.dlt	41	ppi.80211-common.chan.flags.turbo	63	data.data
20	ppi.field_type	41	ppi.80211-common.chan.flags.cck	64	data.len
21	ppi.field_len	43	wpan.frame_length		
22	ppi.80211-common.rate	44	wpan.pan_id_compression		

Table 10. Preprocessing result o the Wifi dataset

No	Feature	No	Feature	No	Feature
1	frame.encap_type	33	ip.dsfield.ecn	65	tcp.flags.ns
2	frame.time	34	ip.len	66	tcp.flags.cwr
3	frame.offset_shift	35	ip.id	67	tcp.flags.ecn
4	frame.time_epoch	36	ip.flags	68	tcp.flags.urg
5	frame.time_delta	37	ip.flags.rb	69	tcp.flags.ack
6	frame.time_delta_displayed	38	ip.flags.df	70	tcp.flags.push
7	frame.time_relative	39	ip.flags.mf	71	tcp.flags.reset
8	frame.number	40	ip.frag_offset	72	tcp.flags.syn
9	frame.len	41	ip.ttl	73	_ws.expert
10	frame.cap_len	42	ip.proto	74	tcp.connection.sack,
11	frame.marked	43	ip.checksum	75	_ws.expert.message
12	frame.ignored	44	ip.checksum.status	76	_ws.expert.severity
13	frame.protocols	45	ip.src	77	_ws.expert.group
14	frame.coloring_rule.name	46	ip.addr	78	tcp.flags.fin
15	frame.coloring_rule.string	47	ip.src_host	79	tcp.flags.str
16	eth.dst	48	ip.host	80	tcp.window_size_value
17	eth.dst_resolved	49	ip.dst	81	tcp.window_size
18	eth.addr	50	ip.addr	82	tcp.checksum
19	eth.addr_resolved	51	ip.dst_host	83	tcp.checksum.status
20	eth.lg	52	ip.host	84	tcp.urgent_pointer
21	eth.ig	53	tcp.srcport	85	tcp.options
22	eth.src	54	tcp.dstport	86	tcp.options.mss
23	eth.src_resolved	55	tcp.port	87	tcp.option_kind
23	eth.addr	56	tcp.port	88	tcp.option_len
25	eth.addr_resolved	57	tcp.stream	89	tcp.options.mss_val
26	eth.lg	58	tcp.len	90	tcp.analysis
27	eth.ig	59	tcp.seq	91	tcp.analysis.acks_frame
28	eth.type	60	tcp.nxtseq	92	tcp.analysis.ack_rtt
29	ip.version	61	tcp.ack	93	tcp.analysis.initial_rtt
30	ip.hdr_len	62	tcp.hdr_len	94	tcp.time_relative
31	ip.dsfield	63	tcp.flags	95	tcp.time_delta
32	ip.dsfield.dscp	64	tcp.flags.res		

Table 11. Evaluation performance

Detection rate	Performance
Accuracy	99.92%
Precision	100%
FPR	0
FNR	0.0869

Attacks patterns which are defined as rules are needed as attacks knowledge base, pattern itself and as feature selection engine of the IDE. To validate the significance of the results of the carried out testing experiments on attacks detection, correlation between alerts produced during the experiments and the patterns information in the raw data should be investigated. The correlations are measured by inspecting the timestamp of the alert and the timestamp of the corresponding patterns in the raw pcap data, as shown in Figure 2 to Figure 6.

Performance evaluation on the IDE is carried out using True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) values as metrics, thus Accuracy, Precision, False Positive Rate (FPR), and False Negative Rate (FNR) are computed as presented in Table 11. It is shown that FPR and FNR values are low, 0% and 0.0869, respectively. As for precision level, the IDE achieves 100%, while the accuracy level achieves 99.92%.

$$Accuracy = (TP + TN)/(TP+TN+FP+FN). \quad (1)$$

$$Precision = TP/(TP+FP). \quad (2)$$

$$FPR = FP/(TN+FP). \quad (3)$$

$$FNR = FN/(FN+TP). \quad (4)$$

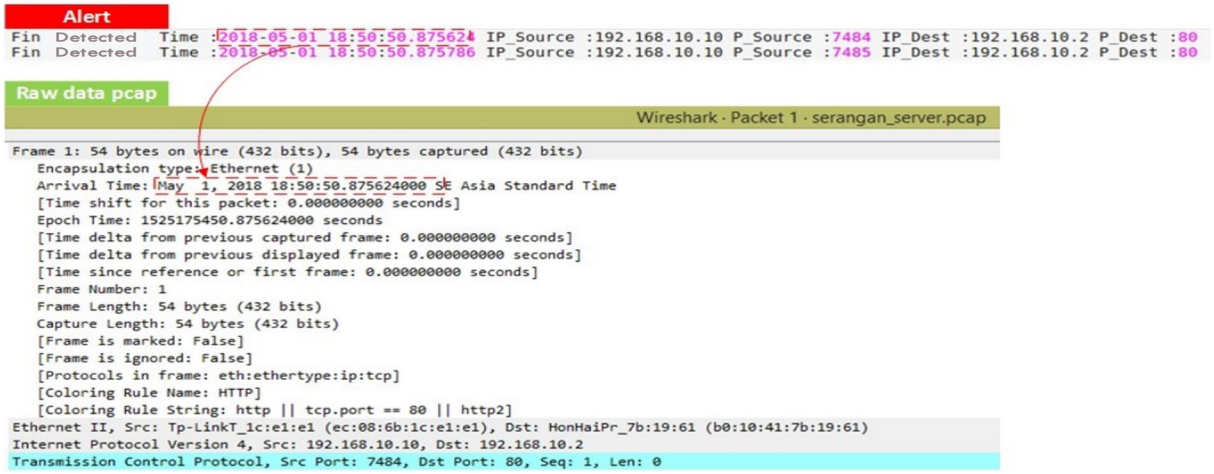

Figure 2. Server attack dataset test correlation

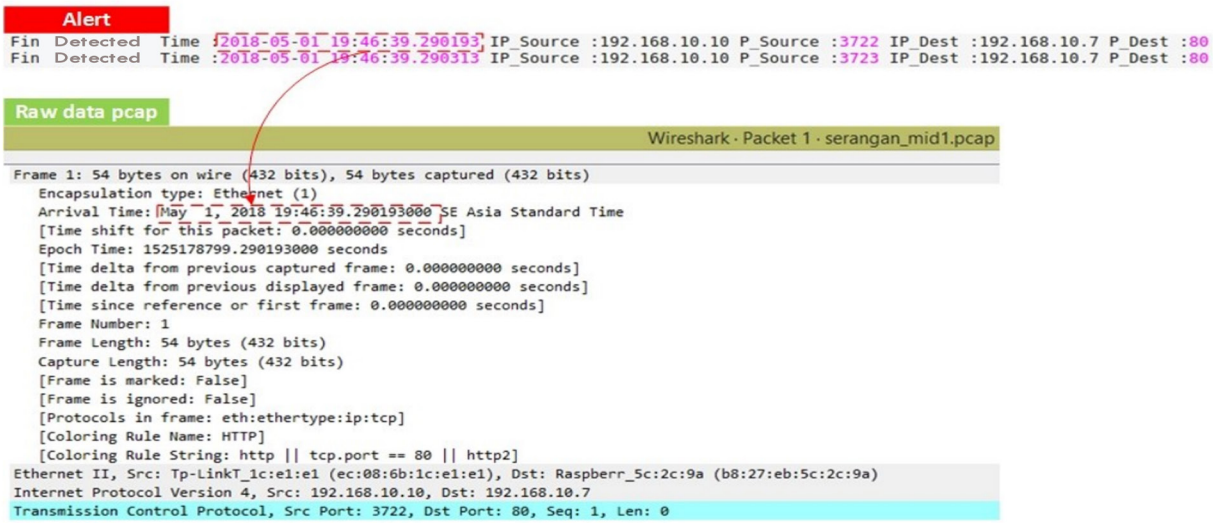

Figure 3. Middleware 1 attack dataset test correlation

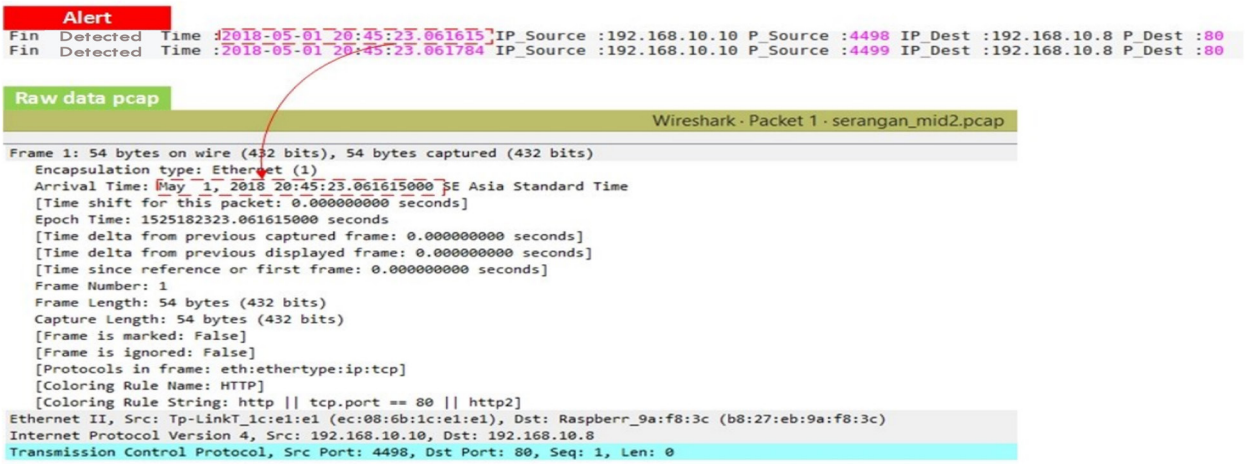

Figure 4. Middleware 2 attack dataset test correlation

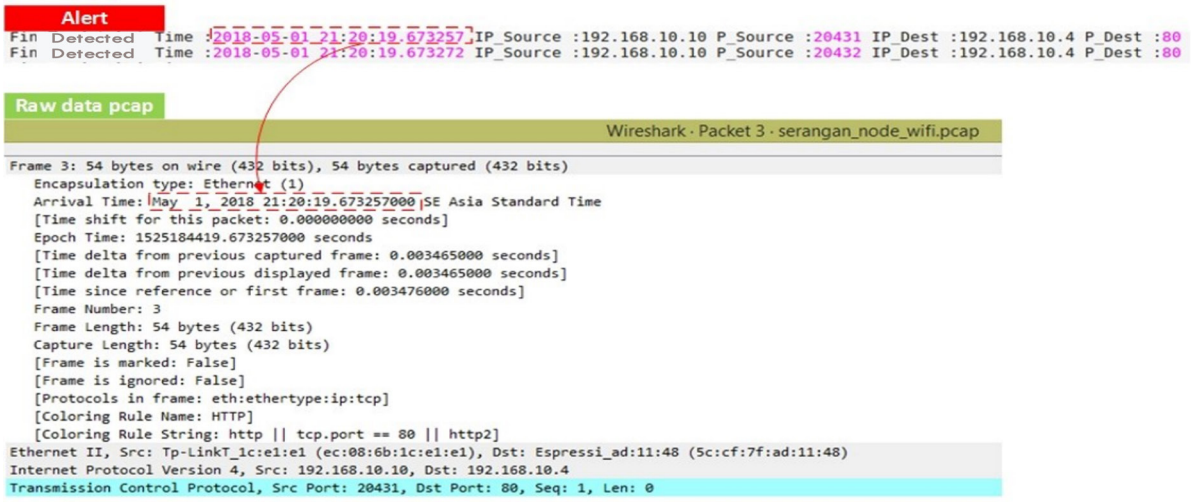

Figure 5. Node Wifi attack dataset test correlation

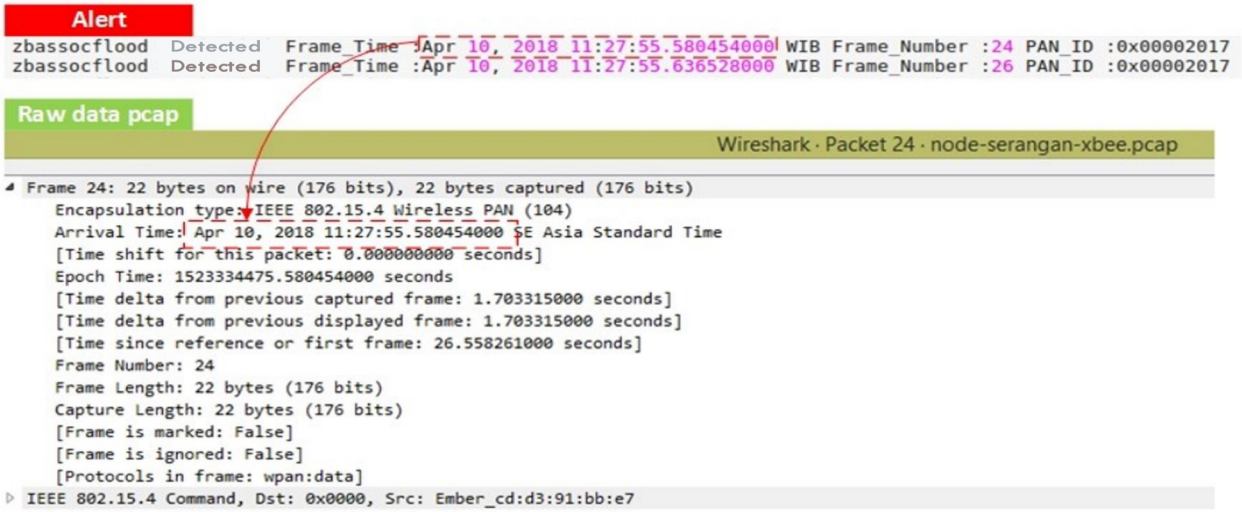

Figure 6. Node Xbee attack dataset test correlation

6 Summary

This study has created a dataset using a heterogenous IoT network with a variety of end devices (Arduino, Raspberry Pi, WeMos D1, etc.), sensor devices (DHT-11, DHT-22, FC-28, K-0135, MQ-2, etc.), Communication Protocol IEEE 802.11 (WiFi) and IEEE 802.15.4 (ZigBee) and subjected it to a variety of DoS/DdoS FIN flood, UDP flood, and Zbassocflood/association flood attacks. The resulting overall dataset is labeled with normal, attack, and combined normal-attack dataset categories. Processing the dataset has generated 95 attributes or features for the datasets using the IEEE 802.11 communication protocol (WiFi) and 64 attributes or features for the datasets using the IEEE 802.15.4 (ZigBee) communication protocol.

Supplementary Material

The online version of the datasets associated with this article can be found at DOI: https://dx.doi.org/10.21227/n109-ng79

References

[1] S. Li, L. Da Xu, S. Zhao, The internet of things: a survey, *Information Systems Frontiers*, Vol. 17, No. 2, pp. 243-259, April, 2015.

[2] L. L. Hung, Intelligent Sensing for Internet of Things Systems, *Journal of Internet Technology*, Vol. 23, No. 1, pp. 185-191, January, 2022.

[3] A. A. Diro, N. Chilamkurti, Distributed attack detection scheme using deep learning approach for Internet of Things, *Future Generation Computer Systems*, Vol. 82, pp. 761-768, May, 2018.

[4] F. A. Alaba, M. Othman, I. A. T. Hashem, F. Alotaibi, Internet of Things security: A survey, *Journal of Network and Computer Applications*, Vol. 88, pp. 10-28, June, 2017.

[5] S. Y. Moon, J. H. Park, J. H. Park, Authentications for Internet of Things Security: Threats, Challenges and Studies, *Journal of Internet Technology*, Vol. 19, No. 2, pp. 349-358, March, 2018.

[6] J. O. Nehinbe, A Simple Method for Improving Intrusion Detections in Corporate Networks, in: D. Weerasinghe (Eds.), *Information Security and Digital Forensics. ISDF 2009. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Vol. 41 LNICST, Springer, 2010, pp. 111-122.

[7] R. Koch, M. Golling, G. D. Rodosek, Towards Comparability of Intrusion Detection Systems: New Data Sets, *TERENA Networking Conference*, Dublin, Ireland, 2014.

[8] A. Gharib, I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, An Evaluation Framework for Intrusion Detection Dataset, *International Conference on Information Science and Security (ICISS)*, Pattaya, Thailand, 2016, pp. 1-6.

[9] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, *4th International Conference on Information System Security and Privacy*, Madeira, Portugal, 2018, pp. 108-116.

[10] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, A. Hotho, A survey of Network-based Intrusion Detection Data Sets, *Computer Security*, Vol. 86, pp. 147-167, September, 2019.

[11] A. Sperotto, R. Sadre, F. Van Vliet, A. Pras, A Labeled Data Set for Flow-Based Intrusion Detection, in: G. Nunzi, C. Scoglio, X. Li (Eds.), *IP Operations and Management. IPOM 2009. Lecture Notes in Computer Science*, Vol. 5843, Springer, 2009, pp. 39-50.

[12] A. Shiravi, H. Shiravi, M. Tavallaee, A. A. Ghorbani, Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection, *Computers & Security*, Vol. 31, No. 3, pp. 357-374, May, 2012.

[13] G. Creech, J. Hu, Generation of a New IDS Test Dataset: Time to Retire the KDD Collection, *IEEE Wireless Communications and Networking Conference (WCNC)*, Shanghai, China, 2013, pp. 4487-4492.

[14] N. Moustafa, J. Slay, UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set), *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015, pp. 1-6.

[15] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, X. Bellekens, Machine Learning Based IoT Intrusion Detection System: An MQTT Case Study (MQTT-IoT-IDS2020 Dataset), in: B. Ghita, S. Shiaeles (Eds.), Se*lected Papers from the 12th International Networking Conference. INC 2020. Lecture Notes in Networks Systems*, Vol. 180, Springer, 2021, pp. 73-84.

[16] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, R. Therón, UGR'16: A New Dataset for the Evaluation of Cyclostationarity-based Network IDSs, *Computers & Security*, Vol. 73, pp. 411-424, March, 2018.

[17] Kamaldeep, M. Malik, M. Dutta, Contiki-based Mitigation of UDP flooding Attacks in the Internet of Things, *IEEE International Conference on Computing, Communication and Automation (ICCCA 2017)*, Greater Noida, India, 2017, pp. 1296-1300.

[18] Allot Communications, *DDoS Attack Handbook*, Allot Communications Ltd, 2018, p. 20.

[19] D. Stiawan, D. Wahyudi, A. Heryanto, Samsuryadi, M. Y. Idris, F. Muchtar, M. A. Alzahrani, R. Budiarto, TCP FIN Flood Attack Pattern Recognition on Internet of Things with Rule Based Signature Analysis, *International Journal of Online and Biomedical Engineering (iJOE)*, Vol. 15, No. 7, pp. 124-139, April, 2019.

[20] B. Stelte, G. D. Rodosek, Thwarting attacks on ZigBee - Removal of the KillerBee Stinger, *9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, Switzerland, 2013, pp. 219-226.

[21] C. Azzi, *Vulnerability Analysis and Security Framework for Zigbee Communication in IOT*, Master Thesis, University of Nevada, Las Vegas, United States of America, 2016.

Biographies

Deris Stiawan received PhD degree in Computer Engineering from Universiti Teknologi Malaysia, Malaysia, in 2013. He is currently an Associate Professor at Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer network, Intrusion Detection/ Prevention System, and heterogeneous network.

Dimas Wahyudi received bachelor degree from Department of Computer Engineering, Universitas Sriwijaya, Indonesia, in 2018. His main interest includes both theoretical and practical aspect of Internet of Things and Computer Network under Computer Network & Information Security (COMNETS) Research Group at Faculty of Computer Science, Universitas Sriwijaya since 2017.

Tri Wanda Septian is currently a lecturer at the Computer Engineering Department, Universitas Sriwijaya. He holds Offensive Security Certified Professional (OSCP), EC-Council's Certified Ethical Hacker (CEH) practical and EC-Council's Certified Secure Computer User (CSCU). His research interests include heterogeneous networks, distributed systems, and network security.

Mohd Yazid Idris is an Associate Professor at School of Computing, Faculty of Computing, Universiti Teknologi Malaysia. In software engineering, he focuses on the research of designing and development of mobile and telecommunication software. His main research activity in IT security is in the area of Intrusion Prevention and Detection.

Rahmat Budiarto is a full Professor at the Department of Computer Engineering and Science, Al Baha University, Al Bahah, Saudi Arabia. His research interests include intelligent systems, brain modeling, IPv6, network security, Wireless sensor networks, and MANETs.