Future Generation Computer Systems 108 (2020) 414–423

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Real time dataset generation framework for intrusion detection systems in IoT

Yahya Al-Hadhrami [*], Farookh Khadeer Hussain

School of Computer Science and Centre for Artificial Intelligence, Faculty of Engineering and Information Technology, University of Technology, Sydney, Ultimo, NSW, 2007, Australia

ARTICLE INFO

Article history:
Received 15 September 2019
Received in revised form 3 February 2020
Accepted 19 February 2020
Available online 26 February 2020

Keywords:
IoT security
Trust
Intrusion detection system

ABSTRACT

The Internet of Things (IoT) has evolved in the last few years to become one of the hottest topics in the area of computer science research. This drastic increase in IoT applications across different disciplines, such as in health-care and smart industries, comes with a considerable security risk. This is not limited only to attacks on privacy; it can also extend to attacks on network availability and performance. Therefore, an intrusion detection system is essential to act as the first line of defense for the network. IDS systems and algorithms depend heavily on the quality of the dataset provided. Sadly, there has been a lack of work in evaluating and collecting intrusion detection system related datasets that are designed specifically for an IoT ecosystem. Most of the studies published focus on outdated and non-compatible datasets such as the KDD98 dataset. Therefore, in this paper, we aim to investigate the existing datasets and their applications for IoT environments. Then we introduce a real-time data collection framework for building a dataset for intrusion detection system evaluation and testing. The main advantages of the proposed dataset are that it contains features that are explicitly designed for the 6LoWPAN/RPL network, the most widely used protocol in the IoT environment.

© 2020 Published by Elsevier B.V.

1. Introduction

Intrusion detection systems (IDS) have been utilized in the field of computer science since 1970, especially for attack detection and network security monitoring [1]. An IDS in its basic form consists of a data acquisition unit which monitors the flow of data within the premises of a network and processes it to decide on the status of the network flow. This kind of anomaly detection requires a good quality dataset for training and testing all of the possible scenarios and making accurate and precise decisions for new network flows [2]. There are two main types of IDS. The first is a signature-based IDS that uses a predefined set of rules and patterns to detect attacks. The second is an anomaly-based IDS that uses a pre-collected dataset with either normal or malicious behavior data and uses it to train and test a detection model. Therefore, any network flow that deviates from the model threshold is marked as an anomalous entry and reported as an attacker. The key component for a good IDS system is a well-structured dataset that reflects a realistic real-world scenario. A popular dataset designed for an IDS in 1998 is the KDDCup 99 dataset, which is part of the DARPA intrusion detection evaluation program. The KDDcup99 was considered the golden standard for IDS evaluation [3], however many studies such as [4–6] criticized it, citing problems such as inconsistencies in the ratio of normal and attack data and a high number of duplicates. Therefore, several research efforts have attempted to construct a new dataset for IDS evaluation in traditional network, however to our knowledge, no attempt has been made to build a new dataset for the emerging Internet of Things (IoT). Although the KDD Cup 99 dataset is outdated and was designed for a different set of protocols and applications, many researchers have tried to apply the same dataset to an IoT-based architecture but most of these studies failed to report the applicability of such solutions in a real-world implementation. Furthermore, the lack of IoT datasets that have various kinds of IoT attack segments reflecting the different attack patterns is a critical issue that must be addressed to further enhance the quality of research in this field. We believe this paper can serve as the basis for future work that adddress different security issues in the Internet of things.

In this paper, we explore the limitations associated with the existing IDS datasets and propose new framework for generating an IoT dataset for IDS evaluation. The rest of the paper is organized as follows: Section 2 explores the related literature and existing datasets. In Section 3, the proposed framework is explained, including feature engineering and selection. The datasets are analyzed in Section 4. In Section 5, we compare the results of the produced datasets and compare them with several

[*] Corresponding author.
E-mail addresses: yahya.s.al-hadhrami@student.uts.edu.au (Y. Al-Hadhrami), farookh.hussain@uts.edu.au (F.K. Hussain).

https://doi.org/10.1016/j.future.2020.02.051
0167-739X/© 2020 Published by Elsevier B.V.

existing datasets. Finally, conclusions are drawn in Section 6 with suggestions for future work.

2. Related works

Over the years, many IDS dataset frameworks have been proposed to ease the process of generating up-to-date datasets. However, most of the studies have focused on traditional networks without exploring different network architectures, such as the IoT ecosystem. Therefore, to justify the need for a new dataset, this section answers the following questions:

1. In the context of IoT security, how is a dataset created?
2. What are the different features and protocol architectures used for dataset creation?
3. Does the dataset design adapt well to different IoT devices and networks?
4. How could one build a real-time dataset for IDS evaluation in IoT?

In previous research, most studies have focused on using the KDD98 dataset as the source for a machine learning intrusion detection system. The KDD98 dataset was created by [7] in 1998 by utilizing the DARPA98 test lab. This dataset has more than 2 million records that can be used for training anomaly detection algorithms. There have been several critics of the KDD98 dataset due to its outdated nature and several redundant entries, leading to incorrect biased results [4]. Therefore, NSL-KDD [8], which is a modified version of the KDD dataset, was proposed as redundant records have been removed from this dataset, it is less error prone. However, this means the new subset NSL-KDD is a subset of the KDD98 and inherits all the other issues of the KDD98 such as the outdated applications and protocols used to create the KDD98. Moreover, both of these datasets were designed for traditional TCP/IP wired networks not for IoT-based networks.

Another publicly available IDS dataset was released in 2006 by Kyoto University in Japan [9]. Distributed honeypots are used by this dataset to capture any abnormal behavior in the network. All traffic captured by the 348 honeypots is considered to be malicious attacks, and all traffic from mail and DNS servers are considered to be legitimate traffic. The process of capturing the dataset took almost three years (Nov 2006–Aug 2009) leading to 50,033,015 normal records and 425,719 considered to be attack sessions. However, this dataset suffers from a few drawbacks, one of which is the attack and normal traffic generation was created in two different environments leading to it being an uncorrelated and unrealistic dataset [10]. Moreover, in light of the rapid developments in technology over the past few years, it is considered to be an outdated dataset being created in 2006.

Sperotto 2008 is a dataset which was generated in 2008 by [11]. It is a flow-based dataset that provides a realistic representation of real network traffic. This dataset was collected in only six days with only one honeypot and a Netflow enabled router at the edge of the network. The author correlated the attack log from the honeypot to the captured flows with 98% accuracy. However, since it was captured in only six days, it does not allow for real traffic evaluation for an extended period of time. Furthermore, the labeling process is dependent on the logs generated by the honeypot. Therefore, any misclassification by the honeypot leads to a wrongly labeled flow.

The MAWILab dataset [12] was created based on the original dataset (measurement and analysis on the wide internet) [13], which is a daily archive recording the traffic between Japan and the United States every 15 min. In this dataset, traffic payload is omitted and all IP address are changed to make them anonymous. The author took this dataset and labeled it using a different set of anomaly classifiers; only data from 2007 was labeled. However, this dataset is not designed to be used for an IoT enabled network since it only focuses on specific protocols such as SSH and HTTP. Moreover, the short capture windows which are 15 min daily is one of the limitations since it does not provide enough insight into network activity throughout the day.

ADFA-LD12 2013 This dataset was created in 2013 by [14]. It is a Linux-based dataset that focuses on host intrusion detection solutions. Several services were selected to run on the test lab machines such as PHP, MySQL, SSH, and the vulnerable TikiWiki web portal. Several attack segments were launched against the Linux box. These attacks included Java-based attacks, FTP and SSH session brute-forcing, and a client side meterpreter to simulate social engineering. These attacks were launched against one server only to build the dataset. There are no details about how the author collected the data or how it correlates with the host logs. Moreover, we do not think this dataset will generalize well for other types of systems since it is not OS agnostic nor is it designed for an IoT system.

UNSW-NB15 This is a considerably recent dataset which was presented in 2015 by [15]. The author used three virtual servers configured with an IXIA automatic attack generation tool to launch nine types of attacks against several boxes in the network. This attack simulation was executed for 31 h, leading to more than 7,000,000 records with 49 features. One issue with this dataset is that it was generated in a simulation scenario and it might not reflect real-world implementation. Furthermore, it is not suitable for IoT networks since it was designed for a different set of protocols.

The ISCX was proposed by [16], and was generated to address the limitations of the previous work by building a dynamic real-time dataset. One of the main contributions of this dataset is it uses a different set of profiles for traffic generation, where profile A is used to represent the attack traffic and profile B is used to represent background traffic noise by simulating real-life computer use activities. One problem with this dataset is that it uses an old operating system (Windows XP) for the experiments, which is not relevant to today's networks. Furthermore, it was not designed for IoT networks.

Other accessible datasets are the CAIDA datasets provided by [17]. These datasets consist of TCPDump traffic traces of DDoS attacks, captured in 2007. Some of these are available publicly to the research community. All of the datasets are heavily anonymized, and in some instances, the protocol information and payload are removed from the dataset. Furthermore, it is targeted to a specific event, this being the DDoS attacks that occurred in 2007, making it restricted to DDoS attack.

2.1. Contribution

By considering the limitations associated with the previous work, this research proposes a real-time solution for data collection and dataset creation for IoT IDS evaluation and testing. Hence, the contributions of this paper can be summarized as follows:

- We present a framework for real-time dataset generation for the IoT network.
- We explore the generation of attack traffic including three known DoS attacks on IoT networks.
- We propose a queuing method for gathering network traffic from multiple sniffing nodes.
- We extract IoT network-related features from three different network layers.
- We generate a dataset from three attack scenarios with 12 features extracted from different level of IoT communication stack.

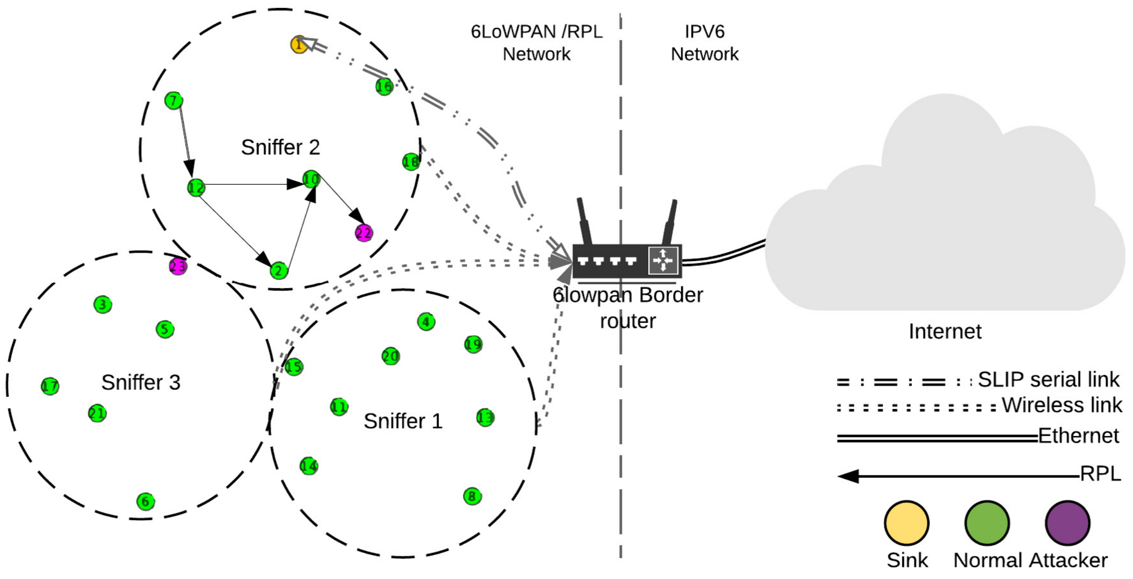

Fig. 1. Network architecture.

Table 1
Simulation parameters.

Parameter	Value
Number of nodes	29 nodes
Cluster head	1
Network area	100 m × 100 m
Size of packet	500
Transmission area	150 m
Routing protocol	RPL
MAC protocol	CSMA
Simulation time	24 h

3. Framework

Building a dataset in any context requires careful design of the collection framework to ensure precise and unbiased results. This involves a tremendous amount of effort in pre-planning and thinking out how every aspect of the network should be designed, which includes a good amount of trial and error to ensure optimal results. In this paper, we describe the process of designing the network and the different protocols involved. The network topology for this framework is shown in Fig. 1.

To generate the IoT-DDoS, we follow a specific procedure to ensure maximum data quality and realism. The ideal process is to have a real IoT network infrastructure in real-time operation. However, since this process is costly and can cause technical and ethical complications, we designed our network architecture to be based on simulation/emulation design, where the IoT nodes are emulated in the Cooja environment [18].

In Cooja, the IoT nodes are emulated to have real CPU and memory power from the host workstation. Hence, when designing the framework, we took into consideration the applicability and portability of such architecture in real-world scenarios. To emulate the noise level of an IoT network, we placed two distributor nodes that emit a noisy signal at specific interval. All of the network and simulation parameters are summarized in Table 1. Furthermore, the proposed system comprises four components: the capturing medium, data aggregation, queuing unit, and the feature extraction unit as can be seen from Fig. 2. In the following subsections, each part of the dataset generation process is explained.

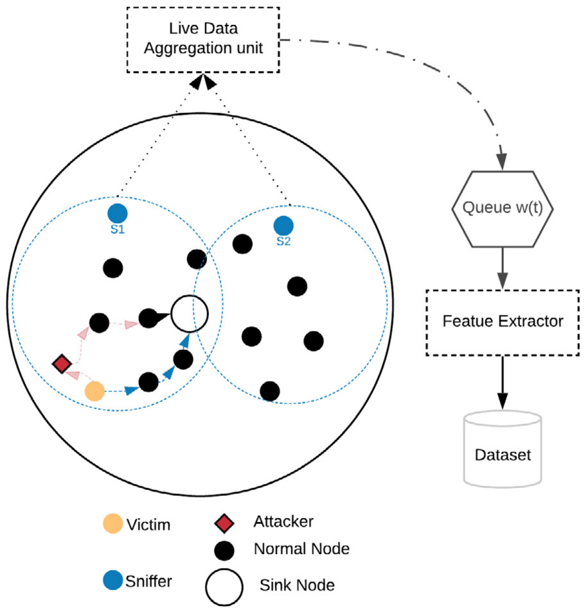

Fig. 2. Data collection model.

3.1. Network and attack scenarios

The network is designed to emulate a real-world IoT network with sensors that are commercially available for industrial IoT. One of the popular limited resource sensors is the Zolertia Z1 mote [19] which supports IEEE802.15.4 standard [20]. Table 2 summarize the type of devices used and the attached sensors. We simulate distributed weather stations that collect humidity and temperature data using the SHT21/SHT25 sensors [21] (see Fig. 3)

All of the sensors in the network use the same mote type (z1) including the sink, which means it is a homogeneous network since all of the nodes in the network use the same protocols and hardware. However, the data collection stage should not be different for a heterogeneous network that uses IEEE802.15.4 and 6LoWPAN protocols for networking [22]. All of the IoT nodes run on modified Contiki OS version 3.0 [23], including the sniffer

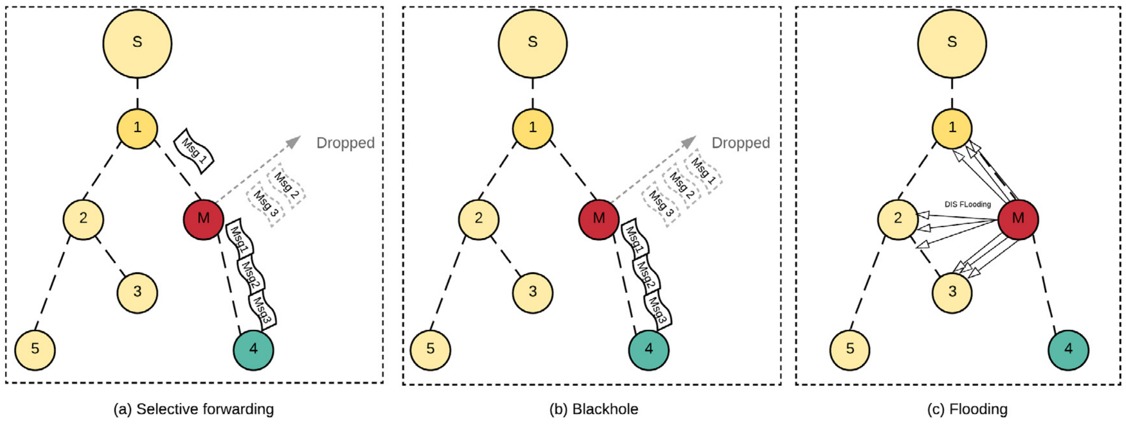

Fig. 3. Attacks .

Table 2
Sensor node parameters.

Name	Value
Mote	Zolertia z1
CPU	16-bit RISC CPU @16 MHz
RAM	8 kb
Memory	92 kB
Transceiver	CC2420
Wireless	IEEE802.15.4/2.4 GHz
Power	3.3 V/5 V
Sensors	SHT21/SHT25

nodes distributed across the network. As for routing within the 6LoWPAN architecture, we used the standard networking stack embedded within Contiki OS. However, to reproduce the three attacks, we had to restructure the RPL implementation [24] as explained in Section 3 of this paper. The 6BR router runs natively on an Ubuntu 18.4 box that handles all of the connections coming from the sniffers and the sink node. The connection between the sink node is a serial link using the slip protocol [25]. Here we assume the sink node is a z1 node. Therefore, we use the serial link to exchange data. The connection between the Internet and the 6BR router is an Ethernet link to ensure high network throughput, in case it is needed.

To emulate a real-world IoT environment, we designed four scenarios, each of which follows the same network topology. The first one is the normal behavior scenario without any attack introduced. The other three introduces some kind of attack on the network. The three variants of attacks are explained as follows:

Algorithm 1: DIS flooding attack

```
Result: Flooding attack
RplDisInterval = 0;
while i <10 do
    sendDIS();
    i++;
end
```

Flooding attack: In a flooding attack, the attacker node sends an unlimited number of DIS messages to the victim node over a short period of time, causing a denial of service attack and disabling the node services. This kind of attack can also lead to battery exhaustion as it consumes all the node's resources. All of the adjacent nodes to the attacker will be affected, since all these nodes connect wirelessly using the IEEE802.15.4 standard. To implement this attack in Contiki OS [23] we had to reverse engineer the networking architecture and add Algorithm 1 to the OS RPL module. The basic idea of this Algorithm is to set the RPL DIS message interval to 0, and for every segment of the DIS interval, 10 DIS messages are sent continuously.

Selective forwarding attacks: This is a commonly used routing attack that tries to selectively forward only particular packets to the next node by selectively dropping specific data of the packets. Such attacks can be extremely dangerous when combined with other attacks like the sinkhole attack which can lead to a Denial of Service (DoS). To implement the selective forwarding attack in Contiki OS, the networking module had to be modified using Algorithm 2, where the attacker, in its core networking module, checks if the packet is a UDP or not. If it is a UDP packet, then it is dropped and all of the other kinds of packets are sent normally. In this type of attack, the attacker selectively drops only the UDP packets (see Fig. 3).

Algorithm 2: Selective forwarding

```
Result: Drop UDP packet
packet = incoming packet;
while packets.size != 0 do
    if packet.payload == "udp" then
        drop.packet
    else
        continue;
    end
end
```

Blackhole attack: A blackhole attack is similar to a selective forwarding attack, but instead of forwarding specific packets, it drops all kinds of packets coming from other nodes. This attack can disturb the network topology by manipulating the ranking mechanism in the RPL protocol. To implement this attack and maximize its maliciousness, there are two steps involved. The first step is to ensure the attacker node has the best possible route to the sink node. Therefore, all of the neighbor's nodes will change their route to the sink to be through the attacker node. This is achieved by modifying the RPL ranking system in the Contiki OS source code, followed by the procedure of compiling and flashing the sensor node. The second step is similar to what has already been implemented in the previous selective forwarding attack, but instead of dropping only the UDP packets, it drops all of the packets going to the root node. This is explained in Algorithm 3.

3.2. Traffic generation

The sensor traffic is generated using the simulation environment Cooja [18], where the Z1 is used with the cc2420 transceiver as discussed earlier. All the nodes run a custom humidity and temperature sensor application which is implemented specifically for this data collection scenario. The length of the interval

Algorithm 3: Blackhole

```
Result: Drop all packets & and decrease rank
packets = incoming packets;
Dag Rank = root rank; while packets.size != 0 do
    if packet.dist == "root" then
        drop.packet
    else
        continue;
    end
end
```

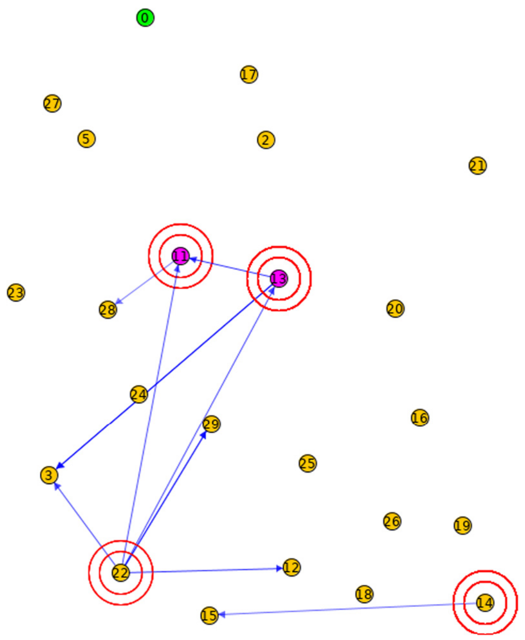

Fig. 4. Network topology.

in which the sensors send the data to the sink node is 30 s. Fig. 5 shows a comparison of the different UDP packet flows over time between the three attacks. From the flow, we can see the blackhole attack affects far less UDP packets of the three attacks, because the attacker drops all of the packets it receives. In the network as can be seen from Fig. 4, there are two types of nodes as follows:

1. Sink node: This is sometimes called the root node where all of the network packets are forwarded. The sink node has the highest rank across the network since it maintains the node hierarchy in RPL network. Moreover, it is a storing type, which means all of the network routes are stored within this node. At the application layer level, this node works as a UDP server where it handles all of the incoming UDP packets from the client's nodes. These UDP packets contain various kinds of information which was designed especially for the generation of this dataset. Furthermore, the sink node takes the responsibility of forwarding the data to the gateway node, where all of the data extraction and processing happens. The functionality of the sink node application is developed using the core functionality available within the Contiki OS.
2. Client node: This is a normal sensor node that senses the humidity and temperature and forwards the values to the sink node. Along with the humidity and temperature data, this node also sends other valuable information about its status such as the RSSI level, LQI, and the ETX values encapsulated as UDP payload.
   Both kinds of nodes implement the 6LoWPAN protocol on top of the IEEE802.15.4 standard for wireless communication. Although for this specific solution, we use the IEEE802.15.4 as a communication medium, the proposed framework can adapt to any protocol at the physical layer level.

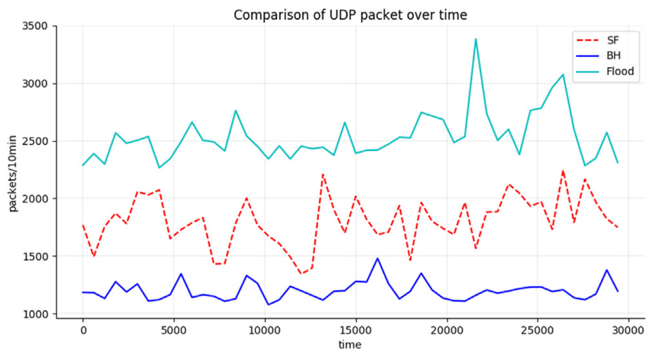

Fig. 5. UDP statistics.

3.3. Capturing data

In this section, we explain the process used to capture the raw data from the IoT network. Since the IoT network usually consists of limited resource devices, adding an extensive data collection and monitoring is not feasible. Therefore, we propose the use of props placed across the network and distributed equally in the network by the network administrators. In this study, we assume the props have more power than the other nodes in the network. These props continuously monitor and collect the data and send them to the data aggregation unit. To achieve this, we use the Sensniff model [26] to sniff and monitor the network continuously. At the hardware level, the probes are equipped with the popular CC2420 [27] transceiver which uses a large antenna to cover a wide range of the network. As the software level, Sensniff firmware is used. Sensniff is an open-source firmware used as network sniffing firmware.

On top of Sensniff, the probes are equipped with the Libpcap library [28] which is used to capture the link layer network traffic. Utilizing Libpcap, we develop a capturing algorithm defined as follows:

Algorithm 4: Capturing algorithm

```
Result: packet
packet = incoming packet;
while packets.size != 0 do
    tmp = packet[0].timestamp - packet.timestamp;
    if packet.size =< 40 then
        send packet.payload to queue;
    else
        continue;
    end
end
```

Algorithm 4 is used to define how the Libpcap library is used. Also, as can be seen, the algorithm captures the raw data at the MAC layer level to be sent later to the data aggregation unit, which is described in the next section. One critical point to mention here is that the data collected at this stage is raw IEEE802.15.4 data which has all of the related signal information.

3.4. Data aggregation

The main function of this module is to aggregate the data from multiple sources and ensure data integrity. The process is to ensure there is no data duplication in the collected data. We create a specific algorithm embedded with the queuing algorithm as follows:

1. Check how many sniffer nodes are attached to the network and store each of these sniffers in an array. This is crucial to aggregate all of the network traffic from all the nodes across the network. Although multiple props will be needed because of the nature of the wireless signal, there is a possibility of overlapping signals that can cause data redundancy and duplication. In addition, more props are needed to cover the large IoT network. Therefore, the need for the data aggregation unit is crucial.
2. Iterate between each node to check if there is any node ID duplication and double check it with the timestamp for each packet. If there is duplication as well, then run the duplication process check to compare all of the packet signatures with the sensor node. This step is crucial since it has multiple levels of the integrity check. If a packet has been duplicated, this can be identified by comparing the different attributes in each packet.

Each packet has a unique ID which is a combination of three values and they are the source IP, destination IP, and the timestamp. These values are used to check packet redundancy and eliminate any duplication.

3.5. Queue

The aggregated packet is sent to the queue system. The system takes all of the data sent from the data aggregator and checks the time window $_T$. All of the packets that are within the time window are then sent to the feature extraction unit. Alternatively, based on the number of observations, we create a variable time window which indicates how many packets a node has sent. To shed a light on this method let us assume $_T$=10, which means for every 10 observations of data for each node extract and calculate the relevant features. Although this method might be useful, a legitimate question can be asked which is what about the nodes that do not send anything in the network? The solution is to fill any missing data with zeros since there is no value in this block of observations. We design a queuing method to accept either the time or number of observation windows. For this study, a number of observations are used as a window. Algorithm 5 shows the queuing algorithm.

3.6. Feature extraction unit

The feature extraction unit is responsible for extracting the related information from the packets, as shown in Table 3. To ensure coverage of all three layers of the IoT communication stack, we collect the data from the three network layers since we are creating a diverse learning dataset with different kinds of feature vector. The three feature vector are explained below:

1. **Feature vector 1:** This vector includes data collected at the physical layer level such as signal strength and the transmission range.
2. **Feature vector 2:** This vector contains most of the features for this dataset where the data from the networking layer are extracted. This includes data extracted from the RPL and 6LoWPAN protocols.

Algorithm 5: Queuing algorithm

```
Sniffer: S = {$S_1$,...,$S_n$} of size $n$
output: A list of queued packets

node ← list of all nodes in each $S_n$ of size i;
while $i < n$ do
    Q = 0 ;
    for $k \leftarrow 0$ to $n$ do
        for $j \leftarrow 0$ to $i$ do
            if $Q > 0$ then
                if node [k].pkt = node [j].pkt then node is
                duplicated add one of them add node [k].pkt to
                Q ;
                else add node [k].pkt to Q;
            else Q < 0;
            featureExtraction(Q) ;
            continue
        foreach element e of the line i do FindCompress(p);
```

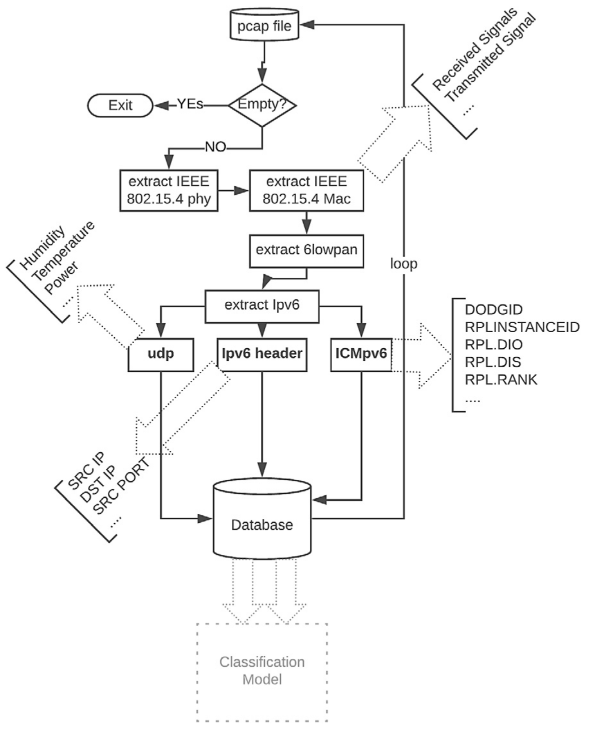

Fig. 6. Data collection model.

3. **Feature vector 3:** This vector includes the sensor reading data extracted at the application layer which for our example contain information about the humidity and temperature readings.

Moreover, most of the IoT networks run using some version of the 6LoWPAN protocol, so the related features are extracted based on the data collected as shown in Fig. 6. The process has a few stages as follows:

- First phase is extracting the features related to the mac layer, such as the RSSI value and the transmission strength. This will help the researcher to understand the correlation of the three different attack and the low level layer parameters such as the transmitted and the current signal strength.

Table 3
Data collection features.

Feature	Description
Physical layer features	
Received signal DBM	Mean value of the received signal at the MAC layer
Transmission signal DBM	Mean value of the transmitted signal at MAC layer
RSSI noise	Mean value of the noise recorded using RSSI
Beacon interval	Mean value of the beacon interval
Network layer features	
LQI	Link quality indicator
ETX	Mean value of the expected transmission count
Number of DIS messages	Number of DIS messages
Number of DIO messages	Number of the RPL DIO messages
RPL rank	Number of rank changes over time.
Number of neighbors	Number of neighbors
Application layer features	
Temperature	Mean value of the temperature
Humidity	Mean value of the humidity
Power level	Mean Value of the energy over time
Consumed power	Mean value of consumed node power
Remaining power	Mean value of the remaining node power
Node ID	Node ID

- The second phase is extracting the network level features like the RPL [29] related features as described in Table 3. Since the blackhole attack and the selective forwarding attack take place in the network layer, these features are the most relevant to our dataset.
- The third phase is extracting the application level features like the data payload in the UDP packet. In this layer, sensor related information is extracted, such as the battery level and the humidity/ temperature information.

To extract all of this information online, we have used the Scapy [30] library on Python that utilizes the powerful Linux library Libpcap for network packet capture and manipulation. The developed parser is capable of connecting to the 6BR router through a serial link port and extracts all of the relevant features as previously explained.

Our feature selection process is divided into three categories, each of which are explained as follows:

3.6.1. Physical layer features

This is where the DCM extracts the physical layer related features such as the received and transmitted signals at the MAC layer. This information is related to physical layer jamming attacks which interfere with the transmitted signals.

3.6.2. Network layer features

Collecting the network layer features is crucial for our IDS since the features extracted are closely associated with many well-known attacks, such as deceased rank attacks and blackhole attacks. In this category, features such as the number of DIS and DIO messages are extracted, with several other features related to the RPL protocol. Data transmitted through any network follow specific protocols. In the IoT, there are certain protocols available at every layer, and in this study, we focus on the RPL protocol on top of the 6LoWPAN protocol. This can be expanded to include extra features from other protocols.

3.6.3. Application layer features

At the application layer, our module collects application specific information such as the humidity, temperature, and node power level. The application related features can be extracted by programming the nodes to calculate the power consumption and other related features We chose to collect power consumption at the application layer because we can program the node to calculate the power level and send the result within the application-related information in the UDP payload, as shown in Fig. 7.

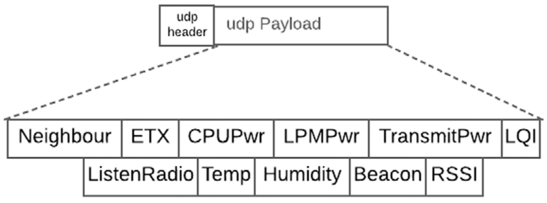

Fig. 7. UDP payload.

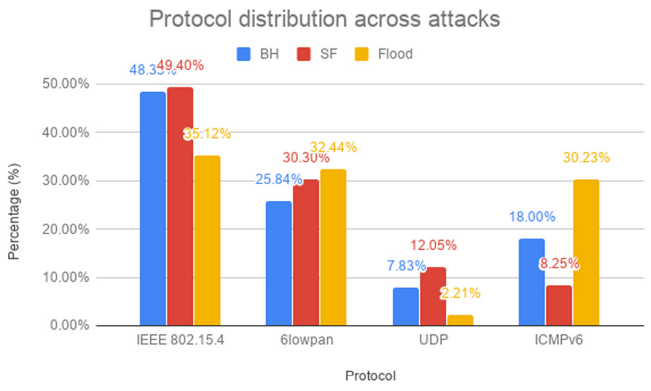

Fig. 8. Protocol distribution.

Including these values at the application layer allows the researcher to measure how different attacks affect the sensor reading and how this can be correlated for attack detection.

3.7. Data labeling

Since we are dealing with sensor network and 6LoWPAN traffic, the process of labeling the data based on the flow is not feasible due to the sheer amount of data transferred from each node. Furthermore, the network data in the IoT network are not transmitted as normal network flow, where the TCP for example uses the three handshake process that can be used to extract network flow for each node. Also, since there is a connection channel between the source and destination in a traditional network flow, this can be extracted easily. Therefore, to solve these issues, we

Table 4
Protocol distribution.

Protocol	Packets	Size (Bytes)	(%)
IEEE 802.15.4	1 391 817	289.5471874	33.17%
6LoWPAN	1 362 111	299.0477588	32.47%
UDP	553 529	15.78560291	13.19%
ICMPv6	888 080	93.6091151	21.17%

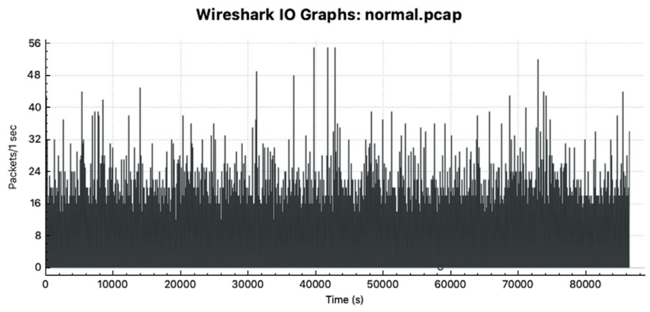

Fig. 9. Network flow.

label the data based on the simulation vector and the timing. Moreover, the segment where the attack takes place will be labeled to indicate at which run the attack occurred and the type of attack. This will help the classifier to narrow down any false positive alarms. The attacker node traffic segment is also labeled to indicate that this attacker is transmitting malicious traffic. Looking into the dataset, we can see the classification method would not have a problem with blackhole attacks and flooding attacks since there is an attack pattern either modified rank attack or a pattern of large number of DIS messages indicating a malicious activity (see Fig. 8).

3.8. Quantitative description of IoT-DDoS

The IoT-DDoS consists of 16 columns and each column represent the nodes for which data is collected. The number of nodes captured in the network is an accurate representation of our proposed capture method. Fig. 9 shows how frequently the packet was sent across the network in a normal scenario. Table 4 shows the contribution of each protocol in the dataset. This is the overall percentage for all of the four scenarios executed. However, this percentage changes when compared individually for each attack. For example, the number of UDP packets for the blackhole and flooding attack drops significantly compared to the ground truth normal behavior dataset. On the other hand, the flooding attack generates the highest number of DIS messages compared to all the other scenarios. Fig. 9 shows the protocol distribution from the attack perspective. Table 4 shows the percentage and size of each protocol in the whole dataset generated. One thing to note is that the results from all of these files are four network PCAP traces which are used to extract these results and generate the dataset.

4. Analysis of the dataset

The IoT-DDoS dataset contains various attack structures as previously discussed. To provide an insight into the complexity of this dataset, we analyze the 96 simulated hours of traffic using all of the acquired features in the different network layers. Fig. 9 shows the network flow of the ground truth normal behavior scenario. One critical aspect of the dataset we want to analyze is how the RPL protocol performs under such long intensive attacks. Fig. 10 shows the rank change over time of one of the parent

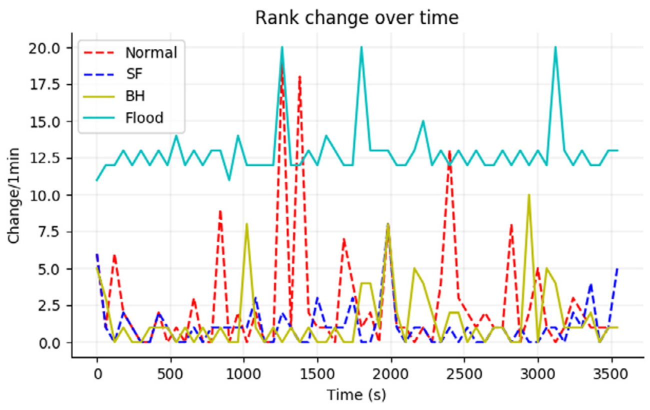

Fig. 10. Rank change over time.

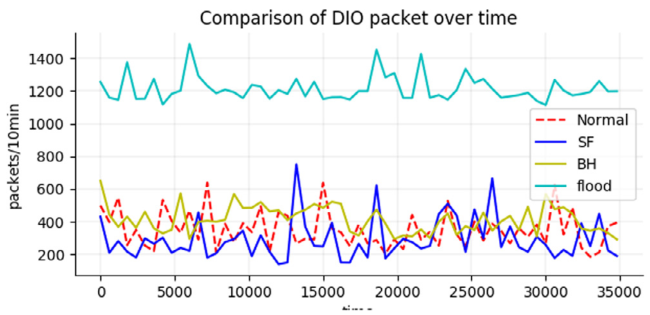

Fig. 11. DIO comparison.

nodes that sits at the end of the DoDAG tree. It can be seen that the average of the rank changes over time is extremely high in the case of flooding attacks in comparison to the ground truth normal behavior. This can be explained because of the type of flooding we implemented, that is, an RPL DIS flooding attack where the attacker node sends unlimited RPL DIS messages in a short interval. Interestingly, this attack affects the ranking behavior of the nodes across the network. Another observation is how the blackhole rank changes for this specific node. As can be seen, it is the most stable among all of the scenarios, The reason being that the attacker was set to have the best rank in the network during the whole simulation period.

On the other hand, the selective forwarding attack shows a similar network pattern to the normal behavior scenario in terms of the number of DIS and DIO messages. However, there is a small abnormal observation when we look at the rank changes over time. There is a frequent fluctuation on the node rank and we think this is due to the stealthiness of the selective forwarding attack and the way it can be misdiagnosed as normal behavior. One important aspect we have to look at when analyzing the RPL protocol is the number of DIO messages sent by the node over time. Fig. 11 shows the DIO messages from each scenario, and as can be seen, the flooding attack has the highest number of sent DIO messages, which we explained earlier due to the frequency of the attack. However, there is a considerable difference between the blackhole attack and the normal behavior scenario. This is because the rank value is actually encapsulated in the DIO message; therefore a higher number of packets is sent.

5. Comparison with other datasets

In this section, we compare some of the characteristics of the IoT-DDoS to other available datasets. Although some of these datasets were considered to be a golden standard for the research

Table 5
Dataset comparison.

Dataset	Realistic network	Labeled	Protocols				Attributes	IoT attacks	Type of flow	IoT environment
			TCP/IP/UDP	6lowpan	RPL	IEEE802.14.5				
KDDcup99	Yes	Yes	Yes	No	No	No	41	No	Dataflow	No
Darpa	Yes	Yes	Yes	No	No	No	41	No	Dataflow	No
NSL-KDD	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
Koyoto	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
CAIDA	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
UNBIS	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
TUIDS	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
Sperotto	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
MAWILab	Yes	Yes	Yes	No	No	No	–	No	Dataflow	No
IoT-DDoS	No	Yes	Yes	Yes	Yes	Yes	12	Yes	Time, Packet based	Yes

community in the area of IDS evaluation, to date, no dataset has been designed which can adapt to the emerging IoT for security evaluation. This research can be differentiated from the prior work by its capability to address the limitations associated with the previous datasets.

Table 4 summarizes all the explored datasets and compares them with the IoT-DDoS datasets, which is the result of this paper. As can be seen from the table, most of the datasets were designed for different sets of protocols that are not currently available in today's IoT networks. No datasets have been collected for network level IoT data, (e.g., 6LoWPAN and RPL traffic) which work as the base of many IoT communication technologies in the market today. Furthermore, none of the explored datasets was designed with IoT malicious activity in mind; all of the datasets evaluated risk activity in relation to a TCP/IP network. In the IoT-DDoS, we were able to implement three kinds of IoT-related attacks as explained in this paper. This can be used as the basis to implement a new solution to counter these attacks using artificial intelligence.

Table 5 details the different parameters involved in creating each dataset, showing that no dataset was explicitly designed for IoT network protocols, rather the focus was on a TCP/IP based traditional network. In the IoT-DDoS dataset, 16 features were extracted from the three level layers.

6. Conclusion and future work

One of the concerns of any researcher in the computer science field is the availability of datasets in their specific area. However, despite the effort provided by the research community in various areas of computer science, there are still limitations when it comes to datasets designed for IDS evaluation and testing. Hence, the need for dynamically updated datasets is crucial to ensure the maximum interoperability across different emerging technologies.

Also, we shed light on the issues associated with previous datasets when it comes to IoT security applications. Furthermore, we addressed some of these issues by designing a systematic approach for dataset creation in the IoT ecosystems. The result was the generation of the IoT-DDoS which includes the implementation of three different attacks related to IoT security. The applicability of this dataset can be extended to include more attacks and security issues. However, at this stage this dataset addresses the need for a comprehensive dataset for IoT security research with three popular attack scenarios. The significance of this dataset to the existing literature is two-fold as follows: (a) This dataset provides IoT researchers with a relevant DDoS dataset that they can use for validating the models developed to counter DDoS attacks; (b) This dataset can be regenerated by the users, since the developed framework can be used to collect data at any point in time. Hence, this dataset is self-sustaining in this regard. In addition to this, we explored all of the available datasets and compared them to our own dataset to ensure the feasibility of new dataset generated.

There are still research to be done for future work. Such as developing a systematic way of adjusting the number of observation windows to dynamically adapt to different network changes. This will allow more diverse network flows to be collected. It is also important to develop more IoT attacks in the datasets to help design a solution that generalizes well for various kinds of security threats.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the University of Technology Sydney, Australia and the Centre for Artificial Intelligence in UTS, Australia. The authors are thankful for the funding entities.

References

[1] H.-J. Liao, C.-H.R. Lin, Y.-C. Lin, K.-Y. Tung, Intrusion detection system: A comprehensive review, J. Netw. Comput. Appl. 36 (1) (2013) 16–24.
[2] J. McHugh, Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, ACM Trans. Inf. Syst. Secur. 3 (4) (2000) 262–294.
[3] W. Lee, S. Stolfo, Data mining approaches for intrusion detection, 1998.
[4] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, IEEE, 2009, pp. 1–6.
[5] S. Revathi, A. Malathi, A detailed analysis on nsl-kdd dataset using various machine learning techniques for intrusion detection, Int. J. Eng. Res. Technol. (IJERT) 2 (12) (2013) 1848–1853.
[6] M.K. Siddiqui, S. Naahid, Analysis of KDD cup 99 dataset using clustering based data mining, Int. J. Database Theory Appl. 6 (5) (2013) 23–34.
[7] I. University of California, KDD Cup (1999), University of California, Irvine (UCI), 2016, http://kdd.ics.uci.edu/databases/kddcup99/.
[8] NSL-KDD, Nsl-kdd data set for network-based intrusion detection systems, 2009, http://nsl.cs.unb.ca/NSL-KDD/.
[9] K. University, Koyto2006, 2006, http://www.takakura.com/Kyotodata.
[10] W. Haider, J. Hu, J. Slay, B.P. Turnbull, Y. Xie, Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling, J. Netw. Comput. Appl. 87 (2017) 185–192.
[11] A. Sperotto, R. Sadre, F. Van Vliet, A. Pras, A labeled data set for flow-based intrusion detection, in: International Workshop on IP Operations and Management, Springer, 2009, pp. 39–50.
[12] R. Fontugne, P. Borgnat, P. Abry, K. Fukuda, Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking, in: Proceedings of the 6th International COnference, ACM, 2010, p. 8.
[13] P. Casas, A. D'Alconzo, G. Settanni, P. Fiadino, F. Skopik, Poster:(semi)-supervised machine learning approaches for network security in high-dimensional network data, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2016, pp. 1805–1807.

[14] G. Creech, J. Hu, Generation of a new ids test dataset: Time to retire the kdd collection, in: 2013 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2013, pp. 4487–4492.
[15] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, 2015, pp. 1–6.
[16] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (3) (2012) 357–374.
[17] CAIDA, Caida (2007). caida ddos attack dataset, 2016, https://www.caida.org/data/passive/ddos-20070804dataset.xml.
[18] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, T. Voigt, Cross-level sensor network simulation with cooja, in: First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), 2006.
[19] M.-P. Uwase, N.T. Long, J. Tiberghien, K. Steenhaut, J.-M. Dricot, Outdoors range measurements with zolertia z1 motes and contiki, in: Real-World Wireless Sensor Networks, Springer, 2014, pp. 79–83.
[20] IEEE, Ieee standard for low-rate wireless networks - amendment 5: enabling/updating the use of regional sub-ghz bands, IEEE Std 802.15.4v-2017 (Amendment to IEEE Std 802.15.4-2015, as amended by IEEE Std 802.15.4n-2016, IEEE Std 802.15.4q-2016, IEEE Std 802.15.4u-2016, and IEEE Std 802.15.4t-2017), 2017, pp. 1–35, http://dx.doi.org/10.1109/IEEESTD.2017.7964803.
[21] S. Nabiha, K. Zaatouri, W. Fajraoui, T. Ezzeddine, New design of low power consumption mote in wireless sensor network, Power 20 (15) (2015) 3.
[22] G. Mulligan, The 6lowpan architecture, in: Proceedings of the 4th Workshop on Embedded Networked Sensors, ACM, 2007, pp. 78–82.
[23] A. Dunkels, O. Schmidt, N. Finne, J. Eriksson, F. Österlind, N.T.M. Durvy, The contiki os: The operating system for the internet of things, 2011, [Online], at http://www.contikios.org.
[24] N. Tsiftes, J. Eriksson, A. Dunkels, Low-power wireless ipv6 routing with contikirpl, in: The 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2010.
[25] J. Romkey, Nonstandard for transmission of ip datagrams over serial lines: Slip, 1988.
[26] G. Oikonomou, Sensniff: Live traffic capture and sniffer for ieee 802.15.4 networks, 2017, https://github.com/g-oikonomou/sensniff.
[27] C. CC2420, 2.4 GHz IEEE 802.15. 4/ZigBee-ready RF Transceiver, Chipcon Products from Texas Instruments, 2006.
[28] S. McCanne, V. Jacobson, The bsd packet filter: A new architecture for user-level packet capture, in: USENIX Winter, Vol. 46, 1993.
[29] T. Winter, Rpl: Ipv6 routing protocol for low-power and lossy networks, 2012.
[30] P. Biondi, et al., Scapy, 2011, https://github.com/secdev/scapy.

Yahya Al-Hadhrami: Received an M.Sc. in computer networks and data communication from the University of Essex in 2012. Currently, he is a Ph.D. student in the School of Computer Science at the University of Technology in Sydney. Previously, he has been working as a computer network and security lecturer in the college of applied science Oman. His research interests focus on the integration of artificial intelligence with IoT security applications.

Farookh Khadeer Hussain is an Associate Professor at the School of Software and the Centre for Artificial Intelligence (CAI) of the University of Technology Sydney (UTS). He is also the Head of Discipline (Software Engineering) in the School of Software. Within CAI, he formed and provides leadership to the Cloud Computing group. His key research interests are in Cloud-of-Things, Cloud Computing, Internet-of-Things, Cloud-driven Data Analytics, Fog Computing and Block chain. He has procured external funding worth more than $2.5 million for various projects. His research has been funded by the prestigious Australian Research Council (ARC) and by other Australian government agencies such as the Department of Industry, Innovation and Science etc. He has successfully co/supervised more than 10 Ph.D. students to completion. Currently, he is supervising around 9 Ph.D. students. He has published widely in top journals such as IEEE Transactions on Industrial Informatics, IEEE Transactions on Industrial Electronics, The Computer Journal, Journal of Computer and System Sciences, Journal of Network and Computer Applications, Future Generations of Computer Systems, Knowledge Based Systems etc... His H-index is 27, i-10 index is 83 and his research has received more than 3250 citations.