Z-IoT: Passive Device-class Fingerprinting of ZigBee and Z-Wave IoT Devices

Conference Paper · June 2020

DOI: 10.1109/ICC40277.2020.9149285

6 authors, including:

Leonardo Babun
Florida International University
48 PUBLICATIONS **1,000** CITATIONS
SEE PROFILE

Hidayet Aksu
Florida International University
48 PUBLICATIONS **2,385** CITATIONS
SEE PROFILE

Kemal Akkaya
Southern Illinois University Carbondale
268 PUBLICATIONS **13,259** CITATIONS
SEE PROFILE

Selcuk Uluagac
Florida International University
181 PUBLICATIONS **5,526** CITATIONS
SEE PROFILE

Z-IoT: Passive Device-class Fingerprinting of ZigBee and Z-Wave IoT Devices

Leonardo Babun[1], Hidayet Aksu[1], Lucas Ryan[1], Kemal Akkaya[2], Elizabeth S. Bentley[3], and A. Selcuk Uluagac[1]

[1]Cyber-Physical Systems Security Lab, Florida International University, FL, USA
[2]Advanced Wireless and Security Lab, Florida International University, FL, USA
[3]Networking Technology, Air Force Research Lab, Rome, NY, USA
[1,2]{lbabu002, haksu, lryan005, kakkaya, suluagac}@fiu.edu
[3]elizabeth.bentley.3@us.af.mil

I. Abstract

***Abstract*—In addition to traditional networking devices (e.g., gateways, firewalls), current corporate and industrial networks integrate resource-limited Internet of Things (IoT) devices like smart outlets and smart sensors. In these settings, cyber attackers can bypass traditional security solutions and spoof legitimate IoT devices to gain illegal access to the systems. Thus, IoT device-class identification is crucial to protect critical networks from unauthorized access. In this paper, we propose Z-IoT, the first fingerprinting framework used to identify IoT device classes that utilize ZigBee and Z-Wave protocols. Z-IoT monitors idle network traffic among IoT devices to implement signature-based device-class fingerprinting mechanisms. Utilizing passive packet capturing techniques and optimal selection of filtering criteria and machine learning algorithms, Z-IoT identifies different types of IoT devices while guaranteeing the anonymity of the network data. To test Z-IoT's efficacy, we implemented several testbeds, including a total of 39 commodity IoT devices that communicate over ZigBee and Z-Wave protocols. Our experimental results showed an excellent performance in identifying different classes of IoT devices with average precision and recall of over 91%. Finally, the proposed framework yields no overhead to the IoT devices or the network traffic.**

***Index Terms*—Internet of Things, device-class fingerprinting, ZigBee, Z-Wave.**

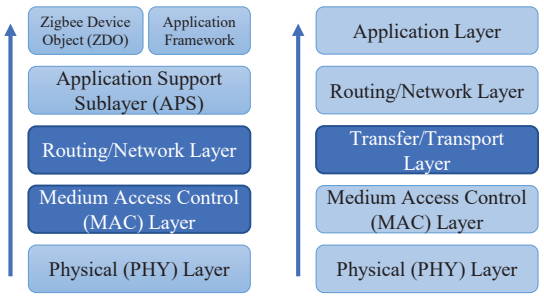
II. Introduction

A survey from the McKinsey Global Institute estimates investments in the Internet of Things (IoT) to be over $11 trillion by 2025 [1]. Indeed, the use of IoT devices in corporate and industrial environments is currently skyrocketing. In most cases, these IoT devices, which have limited computing resources and diverse communication capabilities [2], share access to sensitive information with other networking devices (e.g., servers and gateways) present in corporate networks and critical systems [3]–[8]. In these settings, hackers can impersonate legitimate IoT devices via spoofing attacks and gain unauthorized access to the networks. For instance, using a spoofed device, the attackers can steal sensitive information, inject illegitimate data to the system, or implement targeted attacks over other devices, while mimicking legitimate device operations [9]–[12]. The high diversity of devices and communication protocols (e.g., Internet Protocol (IP), ZigBee, Z-Wave) present in IoT devices makes defending against spoofing attacks extremely difficult.

Passive device-class fingerprinting techniques can be used to identify the type of resource-limited devices present in the network and detect unauthorized devices. Although there is a substantial amount of research in fingerprinting techniques for IP- and Bluetooth-enabled IoT devices, there exist no solutions to identify IoT devices that communicate via ZigBee or Z-Wave, which are very popular in current smart office and home settings [13], [14]. Since different communication protocols typically implement a unique protocol stack and network architecture, IP- and Bluetooth-based identification solutions would not effectively fingerprint ZigBee- or Z-Wave-enabled devices.

Contributions. To solve these limitations, in this paper, we propose Z-IoT, the first fingerprinting framework that performs device-class identification of IoT devices that specifically communicate via ZigBee or Z-Wave (i.e., Z-based devices) protocols. Utilizing passive packet capturing techniques and inter-arrival time (IAT) of network packets sent during idle IoT communications, Z-IoT successfully identifies different types of Z-based IoT devices. To achieve this, Z-IoT implements an optimal filtering and ML selection approach that guarantees the highest accuracy. We evaluated Z-IoT on a realistic testbed using a total of 39 popular ZigBee-, and Z-Wave IoT devices. Z-IoT exhibits a superior performance in identifying different types of ZigBee IoT devices with an average precision of 91.2% and a recall of 91.1%. Also, it identifies Z-Wave devices with an average precision of 93.6% and 93.3% recall. Finally, since Z-IoT performs passive fingerprinting, it does not impose an extra overhead to the IoT devices or the network traffic.

Organization. Section III provides background information on ZigBee, and Z-Wave communication protocols. Then, we review the related work in Section IV. Section V discusses the problem scope and the threat model. Then, in Section VI, we introduce the architecture and main components of Z-IoT. In Section VII, we evaluate the performance of Z-IoT. Finally, Section VIII concludes the paper.



(a) ZigBee Protocol Stack (b) Z-Wave Protocol Stack

Fig. 1: Stack representation of (a) ZigBee and (b) Z-Wave protocols.

III. Background

In this section, we discuss the Z-based (i.e., ZigBee and Z-Wave) communication protocols used in modern IoT devices and network architectures.

A. Z-based Network Protocols

We present details of ZigBee and Z-Wave network protocols that are used to further develop Z-IoT

1) ZigBee: Figure 1(a) details the stack representation of the ZigBee protocol. In this work, we focus on the specific characteristics of the ZigBee Network and MAC layers to build signatures and fingerprinting mechanisms capable of identifying different types of ZigBee IoT devices. Although ZigBee adopted the IEEE 802.15.4-2006 standard for Physical (PHY) and Medium Access Control (MAC) Layers for personal area networks (PAN), it implements a specific Network (NWK) Layer and provides an application framework for developing for the Application Layer.The MAC layer in IEEE 802.15.4-2006 allows for devices to either be a Full Function Devices (FFD) or a Reduced Function Devices (RFD).ZigBee builds upon the MAC layer with its network layer and defines three device types and network topologies: (1) end device, (2) ZigBee router, and (3) ZigBee coordinator.

2) Z-Wave: Figure 1(b) shows that Z-Wave's protocol stack consists of five different layers. Z-Wave implements a specific Transfer layer responsible for managing the communications between different network nodes. This layer implements four different frame types: Singlecast, ACK, Broadcast, and Multicast.Singlecast frames are for transmitting data to a specific node. Then, destination nodes use ACKs to notify the received the data. A Broadcast frame sends the data to every node on the network. Finally, in cases where a node sends data to multiple nodes, a Multicast frame is used. In this work, we focus on the specific characteristics of the Z-Wave Transport layer to build signatures and fingerprinting mechanisms capable of identifying different types of Z-Wave IoT devices.

IV. Related Work

In this section, we discuss the related work and highlight how our work is different from other works.

Network-based Fingerprinting. Earlier works propose techniques to remotely fingerprint devices using TCP protocol features such as microscopic differences in clock skews inferred from TCP packet timestamps. Also, the authors in [15] propose a similar approach to fingerprint network access points using 802.11 beacon frame timestamps. Furthermore, the work in [16] exploits Universal Serial Bus (USB) timing information of the device's responses to the fingerprinting host for identification purposes. On the other hand, technical surveys like [17] provide a comprehensive analysis of the physical layer in the network for the identification of wireless devices. Among other features, electromagnetic characteristics of devices are exploited for Radio Frequency (RF) emitter fingerprinting. Similarly, Bluetooth [18] and Wi-Fi [19] emitter fingerprinting utilize the variations of electromagnetic characteristics of radio frequency emitters; however, these techniques require expensive hardware to be effective. Other works propose passive methods to fingerprint the devices connected to a WLAN [20], [21]. Also, Kurtz et al. [22] study the feasibility of mobile device fingerprinting based on the user's personalized configurations, including Wi-Fi. Their technique requires the active involvement of users and does not apply to the devices that do not support user-customized configurations like Fitbit [23].

Fingerprinting IoT Devices. In [24], researchers present IoT Sentinel, one of the first device-type fingerprinting frameworks for IoT devices that are communicating over IP. IoT Sentinel monitors network traffic of IP connected devices to generate signatures based on IP header fields and the communication protocols. In addition, Miettinen et al. [25] propose a different device-type fingerprinting framework, Dïot. Building off from IoT Sentinel, Dïot continuously monitors network traffic to check if devices' network behavior still matches their corresponding signature. Also, the work in [26] proposes a Bluetooth-based fingerprinting framework to identify wearable device types. Some specific works focus on fingerprinting Z-based IoT devices. In [27], the authors propose the use of non-parametric RF signals to extract features to fingerprint ZigBee specific devices. Similarly, the authors in [28] fingerprint ZigBee hosts using multiple discriminant analysis (MDA). RF signals are also utilized in [29], [30] to authenticate ZigBee devices. Here, characteristics of the ZigBee physical layer are combined with statistical analysis and ML techniques to improve precious authentication mechanisms. Finally, the work proposed in [31] reduces the number of physical layer's features used to fingerprint ZigBee and Z-Wave devices while keeping good classification performance.

Differences from Existing Works. Although Z-IoT also monitors network traffic for fingerprinting purposes, the proposed framework does not analyze packet header or payload information, so the sensitive network information and the privacy of the users is preserved. Also, while other IoT fingerprinting frameworks run on the gateway devices, Z-IoT operates on an independent device outside the network, so no overhead is imposed on the critical network traffic. Furthermore, while other discussed frameworks focus on identifying IP/Bluetooth-

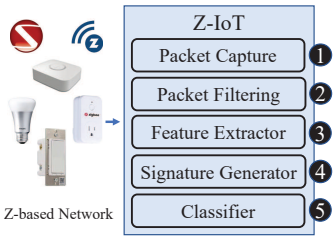


Fig. 2: Overview of Z-IoT architecture.

enabled devices, or specific Z-based hosts, Z-IoT is the first fingerprinting framework that passively uses network dynamics to fingerprint different types of ZigBee and Z-Wave based IoT devices.

V. Problem Definition and Threat Model

In this section, we define the problem as well as the type of threats Z-IoT attempts to solve.

Problem Definition. We consider an IoT network containing various types of devices (e.g., Smart Locks, Motion Sensors, Water Sensors). However, an attacker has gained physical unauthorized access to the network and has placed an unknown device to the network. The unauthorized device is capable of remaining hidden in the network by impersonating a legitimate device via spoofing while performing malicious activities. Z-IoT allows network administrators to detect unknown device-types hiding in the network by identifying the legitimate device-types using network-based fingerprinting mechanisms.

Threat Model. In this work, we consider an attacker that adds unauthorized IoT devices into a network. We only consider unauthorized devices that can mimic authorized devices' functionalities. We assume that unauthorized devices try to perform malicious activities while remaining undetected. To do so, the malicious devices spoof legitimate devices in a network. These spoofing devices may steal sensitive information, inject false data, attack other devices, or place back-doors in a system for future attacks [32]. Z-IoT does not consider devices that are clones or counterfeits of authorized devices, as well as authorized devices that have been compromised.

VI. Z-IoT Fingerprinting Framework

We provide an overview of the proposed IoT device-type identification framework and detail its main modules.

A. Overview

Figure 2 provides a general overview of the proposed IoT device-type identification framework. First, the *Packet Capture* module of Z-IoT performs passive IoT network traffic monitoring during IoT-to-IoT and IoT-to-Gateway idle communications of an unknown IoT device (❶). Following, the *Packet Filtering* module filters the captured data to remove packets that do not match the selected filter settings (e.g., wrong packet type/size) or packets with missing information (e.g., source, destination, packet type, length) (❷). Once the data is ready to be processed, the *Feature Extractor* module calculates the inter-arrival-time (IAT) elapsed between every consecutive packet of the same type received by the packet capture module (❸). The use of inter-arrival-time has proven to be a useful feature for the identification of different device-types via passive fingerprinting [26]. The *Signature Generator* module then uses the collection of IAT values to create a signature of the unknown analyzed device (❹). Finally, the unknown signature is fed to the *Classifier* (❺). This module uses a set of Z-IoT models generated from different known devices, filtering criteria, and machine learning (ML) algorithms during training. The classifier decides, based on a set of predefined known signatures, if the unknown device corresponds to an expected network-authorized IoT device type.

B. Z-IoT Modules

As shown in Figure 2, the main modules of Z-IoT are (1) Packet Capture, (2) Packet Filtering, (3) Feature Extractor, (4) Signature Generator, and (5) Classifier. We detail these modules below.

1) Packet Capture: Z-IoT performs passive device-type fingerprinting with minimal overhead to the IoT devices and the network traffic. To achieve these goals, the Packet Capture module monitors and captures the communication among IoT devices and IoT devices and IoT gateways during idle state only. With this, the proposed framework guarantees the identification of unknown device types with the use of the minimal amount of data and while protecting the sensitive information transmitted during the active state of the IoT devices (e.g., device states, sensor measurements, user information). Once the Packet Capture finishes the collection process, it sends the collected data to the Packet Filtering module for further processing.

2) Packet Filtering: The Packet Filter module of Z-IoT removes unnecessary data to improve performance and increase accuracy. We design this module to be configurable so that the filter settings can be changed anytime, depending on the type of data being processed. New filter settings added to the module may improve performance by removing network packets that do impact the final decision. Also, Z-IoT may add/remove filtering settings to improve the fingerprinting results. For instance, some specific type of network packet that adds randomness to the analysis and reduces the accuracy of the model can be removed from the data. The current implementation of Z-IoT considers *packet type* and *packet length* as filtering features. In Section VII, we evaluate the performance of the proposed fingerprinting framework based on these filter settings. Finally, since ZigBee and Z-Wave implement different protocol stacks, we select specific packet types to be included in the analysis. For instance, in the case of ZigBee, we include packets of type IEEE 802.15.4 and ZigBee's Network Layer packets. On the other hand, for Z-Wave, we include packets of type Singlecast, Broadcast, and Multicast.

3) Feature Extractor: The Feature Extractor module is responsible for extracting the inter-arrival-time (IAT) values from the captured packets. IAT measures the time delay between a device's successive network packets of a certain type (see

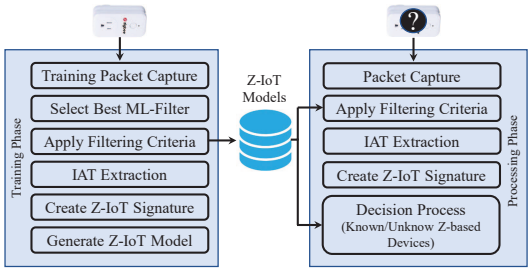


Fig. 3: Detailed representation of the Z-IoT device-type fingerprinting framework during both the training and processing phases.

Equation 1). The proposed fingerprinting framework assumes that different types of Z-based IoT devices incur in values of IAT different enough that can be used for classification purposes. It is possible, however, to find IAT variations between similar types of devices. These variations are generally due to small differences in circuit design and imperfections in circuit modules (e.g., clock). We assume these IAT differences between the similar type of devices to be small enough so our classifier may be able to still achieve high accuracy in classifying different device-types.

$$IAT = \Delta t_{arr} = t_{pk_i} - t_{pk_{i-1}} \quad (1)$$

The use of IAT provides specific advantages over other features used in the literature to fingerprint devices. First, as explained before, IAT values are specific to the different classes of devices and should not vary significantly among devices from the same type group, making the device-class fingerprinting possible. Second, the use of network dynamics such as IAT permits the generation of device-type signatures without the need to analyze network packet content, preserving the sensitive network information and the privacy of users. Finally, we measure the ITA to an external monitor Z-IoT device that is independent of the ZigBee or Z-Wave networks. Keeping Z-IoT outside of the Z-based network allows for a passive approach that does not impose additional traffic overhead to critical Z-based networks.

4) Signature Generator: The Signature Generator module utilizes a device's IAT values to generate a specific signature for the device type. First, it builds a density distribution from a device's IAT values, representing the probability that a network packet received at some specific time *t* was generated by some specific device. Next, the Signature Generator splits the density distribution into n bins and calculates the area under the curve for each bin. The collection of n area values a_i constitutes the final feature values included in every device's signature (see Equation 2).

$$s_{type} = \{a_0, a_1, ..., a_n\} \quad (2)$$

As we monitor the Z-based devices in idle state, we guarantee generating the signature from data collected during periods of low network traffic. Lower traffic allows for signature features that are less impacted by random network dynamics and for higher classification performance.

5) Classifier: We design Z-IoT, assuming multiple different classes of Z-based IoT devices. Moreover, we use a multi-class classification approach in our analysis. The Classifier module is responsible for classifying unknown devices by their type. It takes an unknown device's signature as an input and runs it against a collection of multi-class models generated during training. Z-IoT applies an efficient approach that allows using the best model for every specific communication protocol. For instance, for ZigBee-based devices, Z-IoT selects a model capable of handling ZigBee-generated network packets that guarantees the best filtering criteria and ML algorithm for the highest accuracy results. We trained the classifier via a supervised approach, using labeled signatures from pre-selected known devices. In addition to unique filtering settings for ZigBee and Z-Wave, a multi-class classifier was created for each of the communication protocols and trained only with their associated network traffic.

C. Best ML-Filter Settings Approach of Z-IoT

Figure 3 details the main steps followed by the Z-IoT fingerprinting framework during both training and processing phases. During training, the network packets as a result of idle communications from known Z-based IoT devices are captured and properly labeled. Then, for effective and efficient classification, Z-IoT implements a best ML-Filter settings approach that automatically selects the best classification algorithm and filter settings for every specific type of network packet. This information is then used to create specific Z-IoT models that guarantee the best classifier results. These models are then stored in the Z-IoT database for future use. Later, at processing-time, the right model (depending on the type of network packet analyzed) are selected from the database.

To find which combination of filter settings (e.g., packet type and length) and ML algorithm results in the best performance for Z-IoT, we propose the following logical steps. First, we group and label the set of collected network packets, and define the list of supported ML algorithm and filter setting combinations (the list of specific ML algorithm tested during Z-IoT evaluation is provided in Section VII). In general the process of selecting the best filter-ML combination has to be flexible so new filtering settings and ML algorithms can be included in the analysis anytime. Second, we calculate the accuracy of Z-IoT for every filter-ML combination for the specific network packet type analyzed. Third, the best filter-ML combination is selected using the top 15 percentile (i.e., *top15*) of accuracy values approach.

VII. Performance Evaluation

In this section, we evaluate-IoT based on the following research questions:

- **RQ1** What is the effectiveness of Z-IoT in correctly selecting the best (i.e., top15) filtering settings and ML algorithms for the analysis? (Section VII-C).
- **RQ2** What is the effectiveness of Z-IoT in fingerprinting different Z-based IoT device types? (Section VII-D).

TABLE I: List of Z-based IoT devices, their type, and how many were included on each testbed for evaluation.

Zigbee		
Device	**Device Type**	**Quantity**
SmartThings Arrival Sensor	Arrival Sensor	1
Peanut Outlet	Outlet	2
SmartThings Outlet	Outlet	2
SmartThings Button	Button	3
SmartThings Water Leak Sensor	Water Sensor	2
SmartThings Motion Sensor	Motion Sensor	2
SmartThings Multipurpose Sensor	Multipurpose Sensor	3
Schlage Smart Lock	Lock	2
Z-wave		
Device	**Device Type**	**Quantity**
Aeon Multisensor 6	Multipurpose Sensor	3
Fibaro Flood Sensor	Water Sensor	2
Fibaro Motion Sensor	Motion Sensor	3
Fibaro Door/Window Sensor	Door/Window Sensor	2
NEO Coolcam	Door/Window Sensor	2
GE Dimmer	Dimmer	2
GE Switch	Switch	2
Honeywell Switch	Switch	2
Aeotec Water Sensor 6	Water Sensor	1
Strips Water Drip Sensor	Water Sensor	2
August Smart Lock	Lock	1
Total Number of Devices		**39**

A. Implementation of Z-IoT

To test the efficacy of Z-IoT in fingerprint specific device-types by using network packets extracted from idle ZigBee and Z-Wave communications, we built two different testbeds, one for ZigBee devices and one for Z-Wave-enabled devices. In total, we included 39 Z-based devices from 15 different classes. In Table I, we summarize all the devices considered and specify their type label. In the following subsections, we provide details of both testbeds.

1) ZigBee-type Testbed: We implemented a testbed consisting of 17 ZigBee devices. We also included a Samsung SmartThing hub as the IoT Gateway to capture the idle IoT-to-Gateway idle network communications. We decided to use Samsung's SmartThings platform because it supports a wide range of devices, it is simple to configure, and extremely popular. Our ZigBee network included devices found in a smart office environment like smart locks, smart outlets, motion sensors, and water sensors (see Table I). We first added the SmartThing hub device to our local IP network so it could connect to the SmartThings cloud services. Next, we used the SmartThing Android app to add every device to the ZigBee-based network. Once finished, we started passively capturing network traffic between the 18 edge devices and the SmartThing's hub (IoT Gateway) in an idle state for seven consecutive hours using an AVR RS UZB USB sniffer [33], the killerbee ZigBee framework [34], and Wireshark.

2) Z-Wave-type Testbed: We also built a testbed with IoT devices that communicate via Z-Wave. Our Z-Wave testbed consisted of a total of 22 devices of different types (e.g., smart switch, smart lock, and water sensor). Table I also includes the full list of Z-Wave devices included in the testbed. Since the Samsung SmartThings hub possess both ZigBee and Z-Wave capabilities, we also use the IoT Gateway in our Z-Wave testbed. We then use the SmartThings Android app to add every Z-Wave device into the SmartThing's Z-Wave network. Using the USB Z-Wave 500 Zniffer [35], we captured network packets generated during idle communications between the 22 devices and the SmartThings hub for 24 hours.

TABLE II: Summary of best classifiers per network protocol used during Z-IoT evaluation.

Z-type Protocol	**No. Times in top15**	**Classifier**
Zigbee	4	Bayes Net
	1	Random Forest
	1	LibSVM
Z-wave	2	Random Forest
	1	Bayes Net
	1	Naive Bayes

B. Data Collection and Processing

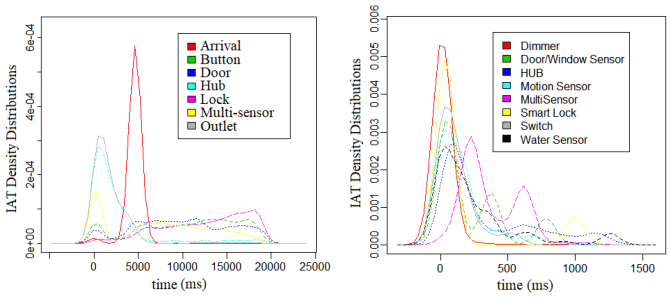
Before we train the Z-IoT models from the captured network packets, we cleaned and processed the data. Wireshark and Zniffer [36] allowed us to export captured packets to a multi-column CSV file. From there, we selected the packet features considered to build the models. In this specific implementation, we considered the following features: (1) capture time, (2) source address, (3) destination address, (4) protocol type, (5) packet length, and (5) the time elapsed from the previously captured frame. Next, we calculated the inter-arrival-times (IATs) from packets to generate signatures for device-types. IAT measures the time delay between a device's successive packets. Using the IATs, we created a density distribution representing the probability that a specific device generated a packet at the time instant t. Next, we divided the density distributions into 300 bins of equal time duration and calculated their geometric area. These 300 area values are used to build the signature vector of every device type. We then used Weka [37] to train and test the Z-IoT models. In addition to the classifier models provided by Weka, we also imported an external neural network implementation with a plugin for Weka [38]. The following ML algorithms were included and considered by Z-IoT's best filter-ML approach explained in Section VI.

- *Functions*: LibSVM, MultilayerPerceptron, NeuralNetwork, SMO, SimpleLogistic
- *Bayes*: BayesNet, NaiveBayes, NaiveBayesUpdateable, NaiveBayesMultinomialUpdateable
- *Rules*: DecisionTable, JRip, OneR, PART
- *Trees*: DecisionStump, HoeffdingTree, J48, LMT, REPTree, RandomForest, RandomTree

Finally, we applied a 10-fold validation approach to train the classifier models for both ZigBee and Z-Wave devices and evaluate their performance.

C. Selecting the Optimal Filtering and Classifier Options

In this section, we evaluate the effectiveness of Z-IoT framework in selecting the best combination of filtering settings and ML algorithm that guarantees the highest accuracy during classification.



(a) ZigBee-type devices (b) Z-Wave-type devices

Fig. 4: Inter-arrival time density distributions of the different (a) ZigBee and (b) Z-Wave enabled device types.

ZigBee. In the case of ZigBee idle communications, Z-IoT collects MAC frames (IEEE 802.15.4) and Network Layer (ZigBee) packets. We first evaluated Z-IoT by filtering MAC frames or Network packets separately. However, we obtained low classification scores for devices of types Lock, Water Sensor, and Outlet. We solved this issues after combining both MAC frames and Network Layer packets as part of the filter settings. Next, we attempted to filter packets based on their length; however, we ran into similar classification issues. With ZigBee devices transmitting small-size packets, roughly 75% of the captured packets were either 5 or 12 bytes long. Moreover, any attempt to filter based on packet length only also resulted in different types of devices not being correctly identified. Thus, for ZigBee IoT devices, Z-IoT applies filter settings that combine both packet type and the packet length. Figure 4(a) shows the IAT density distribution curves for each device-type included in the Zigbee testbed after applying the selected filtering settings. Further, Z-IoT builds a classifier specifically for identifying different types of IoT devices that transmit data over a ZigBee network. Table II shows the top three classifiers as a result of applying the best filter-ML algorithm approach (see Section VI) with the selected filter settings. As observed, Bayes Net classifier performed the best over the other classifiers. As a result, we used Bayes Net to evaluate the performance of Z-IoT in fingerprinting different IoT device classes using ZigBee packets.

TABLE III: Performance metrics of Z-IoT for different Z-based IoT device classes.

Z-type	TPR	FPR	Prec.	Recall	ROC	PRC	Device Class
Zigbee	1.000	0.000	1.000	1.000	1.000	1.000	Arrival
	0.860	0.033	0.811	0.860	0.978	0.798	Button
	0.880	0.007	0.957	0.880	0.990	0.961	Door/Window
	0.960	0.010	0.941	0.960	0.999	0.996	Hub
	0.740	0.030	0.804	0.740	0.958	0.847	Lock
	1.000	0.003	0.980	1.000	1.000	1.000	Multi-sensor
	0.940	0.020	0.887	0.940	0.990	0.891	Outlet
	0.911	**0.015**	**0.912**	**0.911**	**0.988**	**0.928**	**Average**
Z-wave	0.900	0.000	1.000	0.900	1.000	0.998	Dimmer
	0.900	0.037	0.776	0.900	0.963	0.934	Door/Window
	1.000	0.000	1.000	1.000	1.000	1.000	Hub
	0.940	0.003	0.979	0.940	0.997	0.986	Motion
	1.000	0.000	1.000	1.000	1.000	1.000	Multi-sensor
	1.000	0.000	1.000	1.000	1.000	1.000	Lock
	1.000	0.020	0.877	1.000	0.998	0.986	Switch
	0.720	0.017	0.857	0.720	0.983	0.827	Water sensor
	0.933	**0.010**	**0.936**	**0.933**	**0.993**	**0.966**	**Average**

(a) ZigBee-type devices (b) Z-Wave-type devices

Fig. 5: Confusion matrices obtained as a result of the classification of different types of (a) ZigBee and (b) Z-Wave enabled device.

Z-Wave. As for Z-Wave enabled devices, we designed Z-IoT to capture and process Singlecast, Multicast, and Broadcast type packets. Because no device included in our Z-Wave testbed generated Multicast packets, we removed this specific filter setting from the analysis. Next, we evaluated the performance of Z-IoT after filtering Z-Wave packets based on packet length only. With about 95% of Z-Wave captured packets between 10 and 20 bytes and nearly no variations in packet size between different types of devices, we found that packet length only cannot be effectively used as filtering criteria. Finally, as in the ZigBee case, we obtained the best performance results after combining both type and packet length as filtering criteria. The IAT density distributions curves for each type of device in the Z-Wave testbed is shown in Figure 4(b). Next, we evaluated which classifier performs the best for the Z-Wave case. Results from Table II list Random Forest as the best classifier for identifying Z-Wave enabled devices.

D. Classification Performance

We also evaluated the effectiveness of Z-IoT in fingerprinting different Z-based IoT device types. Table III details Z-IoT's performance for the models combining the best filtering criteria and ML algorithms.

In the case of ZigBee IoT devices, Z-IoT identifies different types of devices with an average accuracy, precision, and recall of of 91.0%, 91.2% and 91.1%, respectively. During the evaluation analysis, we also noticed that out of all different device-types considered, the smart locks and the smart buttons were the ones more susceptible to errors of type *false positives*. That is, a higher amount of devices were misclassified as locks and buttons compared to the other types. Figure 5(a) depicts the confusion matrix after applying the selected filtering criteria and the Bayes Net algorithm to classify the different types of IoT devices included in the ZigBee testbed.

Finally, for the case of Z-Wave devices, Z-IoT achieved an average accuracy, precision, and recall of 93.25, 93.6%, and 93.3%, respectively. This time, the device types that prompted the highest false-positive rate were the one of type smart windows/door sensor and smart switch. Figure 5(b) depicts the confusion matrix after applying the selected filtering criteria and the Random Forrest algorithm to classify the different types of IoT devices included in the Z-Wave testbed.

VIII. Conclusion

Modern corporate and industrial networks integrate a variety of IoT devices which can be spoofed by attackers.Although there exist mechanisms to identify unauthorized IoT device communicating via IP networks, there are none for ZigBee or Z-Wave networks. Therefore, in this paper, we introduced Z-IoT, the first device-class fingerprinting framework for IoT devices utilizing ZigBee and Z-Wave protocols. Leveraging passive packet capturing tools, optimal filtering, and machine learning algorithms, Z-IoT passively monitors idle network traffic and uses packets' inter-arrival-times to classify different IoT device types. We tested the efficacy and robustness of Z-IoT with a total of 39 ZigBee and Z-Wave IoT devices. Our results demonstrate that Z-IoT exhibits a superior performance, identifying different types of ZigBee and Z-Wave IoT devices with an average precision and recall of over 91%. Finally, Z-IoT yielded no overhead to the IoT devices and the network.

IX. Acknowledgement

The authors acknowledge U.S. Air Force Research Lab's (AFRL-FA8750-13-2-0116) and U.S. National Science Foundation's (NSF-CAREER-CNS-1453647, NSF-1663051) support in this work. The views expressed belong to the authors only.

References

[1] By 2025, Internet of things applications could have $11 trillion impact, https://www.mckinsey.com/mgi/overview/in-the-news/by-2025-internet-of-things-applications-could-have-11-trillion-impact, 2019, [Online; accessed 11-September-2019].

[2] H. Aksu, L. Babun, M. Conti, G. Tolomei, and A. S. Uluagac, "Advertising in the IoT Era: Vision and Challenges," *IEEE Communications Magazine*, 2018.

[3] L. Babun, H. Aksu, and A. S. Uluagac, "Identifying Counterfeit Smart Grid Devices: A Lightweight System Level Framework," in *2017 ICC*, May 2017.

[4] L. Babun, H. Aksu, and A. S. Uluagac, "A System-level Behavioral Detection Framework for Compromised CPS Devices: Smart-Grid Case," *IEEE Transactions on Cyber-Physical Systems*, October 2019.

[5] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive Information Tracking in Commodity IoT," in *27th USENIX*.

[6] Z. B. Celik, P. McDaniel, G. Tan, L. Babun, and A. S. Uluagac, "Verifying Internet of Things Safety and Security in Physical Spaces," *IEEE Security Privacy*.

[7] L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Real-time Analysis of Privacy-(un)aware IoT Applications," 2019. [Online]. Available: https://arxiv.org/pdf/1911.10461.pdf

[8] C. Kaygusuz, L. Babun, H. Aksu, and A. S. Uluagac, "Detection of Compromised Smart Grid Devices with Machine Learning and Convolution Techniques," in *2018 ICC*.

[9] J. D. Fuller and B. W. Ramsey, "Rogue Z-Wave Controllers: A Persistent Attack Channel," in *2015 LCN Workshops*, 2015.

[10] Babun, Leonardo, Aksu, Hidayet, Uluagac, S. A., "Detection of Counterfeit and Compromised Devices Using System and Function Call Tracing Techniques," Patent 10 027 697.

[11] ——, "Method of Resource-limited Device and Device Class Identification Using System and Function Call Tracing Techniques, Performance, and Statistical Analysis," Patent 10 242 193.

[12] K. Denney, E. Erdin, L. Babun, M. Vai, and S. Uluagac, "USB-Watch: A Dynamic Hardware-Assisted USB Threat Detection Framework," in *Security and Privacy in Communication Networks*, 2019.

[13] L. Babun, A. K. Sikder, A. Acar, and A. S. Uluagac, "IoTDots: A Digital Forensics Framework for Smart Environments," 2018. [Online]. Available: https://arxiv.org/pdf/1809.00745.pdf

[14] A. K. Sikder, L. Babun, H. Aksu, and A. S. Uluagac, "Aegis: A Context-Aware Security Framework for Smart Home Systems," ser. ACSAC 2019.

[15] S. Jana and S. K. Kasera, "On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews," in *MobiCom '08*. ACM, 2008.

[16] L. Letaw, J. Pletcher, and K. Butler, "Host Identification via USB Fingerprinting," in *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, May 2011, pp. 1–9.

[17] B. Danev, D. Zanetti, and S. Capkun, "On Physical-layer Identification of Wireless Devices," *ACM Comput. Surv.*, Dec. 2012.

[18] J. Hall, M. Barbeau, and E. Kranakis, "Detecting Rogue Devices in Bluetooth Networks Using Radio Frequency Fingerprinting," in *In IASTED International Conference on Communications and Computer Networks*, 2006.

[19] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless Device Identification with Radiometric Signatures," in *MobiCom '08*. ACM, 2008.

[20] A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah, "A Passive Technique for Fingerprinting Wireless Devices with Wired-side Observations," in *2013 CNS)*, Oct.

[21] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A Technique for Physical Device and Device Type Fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, 2015.

[22] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting Mobile Devices Using Personalized Configurations," *PETS*, 2016.

[23] I. Fitbit.

[24] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT," in *2017 ICDCS*, June.

[25] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A.-R. Sadeghi, "DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices," *ArXiv*, vol. abs/1804.07474, 2018.

[26] H. Aksu, A. S. Uluagac, and E. Bentley, "Identification of Wearable Devices with Bluetooth," *IEEE Transactions on Sustainable Computing*, 2018.

[27] H. Patel, "Non-parametric feature generation for rf-fingerprinting on zigbee devices," in *2015 CISDA*, May 2015, pp. 1–5.

[28] T. J. Bihl, K. W. Bauer, and M. A. Temple, "Feature selection for rf fingerprinting with multiple discriminant analysis and using zigbee device emissions," *IEEE Transactions on Information Forensics and Security*, 2016.

[29] H. J. Patel, M. A. Temple, and R. O. Baldwin, "Improving zigbee device network authentication using ensemble decision tree classifiers with radio frequency distinct native attribute fingerprinting," *IEEE Transactions on Reliability*, 2015.

[30] C. Dubendorfer, B. Ramsey, and M. Temple, "Zigbee device verification for securing industrial control and building automation systems," in *Critical Infrastructure Protection VII*, 2013.

[31] T. J. Bihl, K. W. Bauer, M. A. Temple, and B. Ramsey, "Dimensional reduction analysis for physical layer device fingerprints with application to zigbee and z-wave devices," in *MILCOM 2015*.

[32] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "HDMI-Walk: Attacking HDMI Distribution Networks via Consumer Electronic Control Protocol," ser. ACSAC 2019.

[33] AVR RZ USB Stick Module, https://www.element14.com/community/docs/DOC-67532/l/avr-rz-usb-stick-module, 2019, [Online; accessed 11-September-2019].

[34] killerbee, https://github.com/riverloopsec/killerbee, 2019, [Online; accessed 11-September-2019].

[35] USB Z-Wave 500 Bridge Controller Reference Design - Silicon Labs, https://www.silabs.com/products/development-tools/wireless/mesh-networking/z-wave/usb-z-wave-bridge-controller-reference-design, 2019, [Online; accessed 11-September-2019].

[36] Z-Wave Zniffer User Guide - Silicon Labs, https://www.silabs.com/documents/login/user-guides/INS10249-Z-Wave-Zniffer-User-Guide.pdf, 2019, [Online; accessed 11-September-2019].

[37] E. Frank, M. Hall, and I. Witten, "Appendix," *Data Mining Practical Machine Learning Tools and Techniques*, 2016.

[38] Java (Convolutional or Fully-connected) Neural Network Implementation with Plugin for Weka, https://github.com/amten/NeuralNetwork, 2019, [Online; accessed 11-September-2019].