2018 IEEE 15th International Conference on Mobile Ad-hoc and Sensor Systems

BF-IoT: Securing the IoT Networks via Fingerprinting-based Device Authentication

Tianbo Gu, Prasant Mohapatra
Department of Computer Science, University of California Davis, USA
Emails: {tbgu, pmohapatra}@ucdavis.edu

***Abstract*—Bluetooth low energy (BLE) based devices are already deployed in massive quantity as Internet-of-things (IoT) becomes prominent in the last two decades. In order to lower the energy consumption, BLE devices have to compromise with security and privacy problems. Existing research work shows that BLE devices can be easily spoofed and leveraged to gain access to a networking system. In this paper, we propose *BF-IoT*, the first IoT secure communication framework for BLE-based networks that guards against device spoofing via monitoring the work-life cycles of devices. We dig into the BLE protocol stack and extract the unique network-flow features from the link layer and ATT/GATT service layer so as to generate the fingerprints for device authentication. *BF-IoT* provides two-phase defense against malicious entities: continuously authenticating device identity before the connection setup and during session establishment. We build a customized system to validate the effectiveness of our mechanism. We extensively evaluate *BF-IoT* with a dozen of different off-the-shelf commodity IoT devices which shows that the devices can be accurately authenticated via only sniffing the transmission characteristics.**

***Index Terms*—Internet of Things (IoT), Bluetooth Lower Energy (BLE), Device authentication, Fingerprinting, Network security**

I. Introduction

IoT has become a household name during the last few years due to the rapid growth of smart devices. We are surrounded by many IoT devices throughout the day which is changing our lifestyle. International Data Corporation forecasts that in the year 2018, the worldwide spending of IoT devices will be around 772 billion [1]. By the year 2020, 24 billion devices will be connected worldwide, and by 2022, on average 500 devices will be connected per household [2]. It is expected that a more significant portion of these devices will be used for monitoring the activity of daily life (ADL), such as sleeping, exercise, eating habits, etc. ADL monitoring will be critical in evaluating our health status, predicting long-term chronic disease and providing medical advice for elderly people.

With its rapid growth, IoT brings unprecedented security and privacy challenges due to its unique characteristics and widespread deployment [3], [4]. Due to its attachment to our life, the security of these devices become very crucial to our daily living. However, IoT devices are constrained by its power and compromised with its security. Most IoT devices are connected to the Internet through short-range wireless communication protocols. IEEE published a few standards specifically for low power device communications [5]. Bluetooth low energy (BLE) [6] dominates in the IoT world due to its low energy consumption which increases the lifetime of the service provision [7]. It is used in almost every smart device we can think of, ranging from smart light-bulb, health tracker to intelligent vehicle, smart city, etc. In the year 2017, 4.4 billion BLE enabled devices had been purchased over the world. BLE is becoming indispensable parts of the wireless communication protocol in IoT.

However, BLE differs from conventional wireless communications such as WiFi, Zigbee, Bluetooth, etc. The stringent low energy transmission pushes the BLE devices to compromise with security, especially, BLE is vulnerable to identity spoofing-based attacks [8], [9]. For instance, by forging the MAC address of a temperature sensor, a malicious entity can send counterfeit temperature report to the smart home controller. The smart home controller guides the air conditioner and may mistakenly adjust the room temperature. This poses an imminent threat to the assistive living. In another case, a malicious agent can forge the same beacon advertisement messages of a wireless headset, which enables the agent to eavesdrop on conversations and sniffs all the conversation with that victim smartphone. Thus, authenticating an intended device is of utmost importance in smart living.

Fingerprint-based identification and authentication mechanisms had been used in our society for a long time because no two entities can pose the same fingerprint. Similarly, in cybersecurity, the term fingerprinting represents a process by which a device or an active agent can be distinguished by observing some external characteristics. There exist some research works about fingerprinting in wireless communications. For example, packet inter-arrival time [10] and the frames' duration field in 802.11 [11] can be utilized for identifying WiFi devices as the observed characteristics that are dependent on the particular device driver. However, these approaches targeting the 802.11 cannot be applied to BLE due to the distinct protocol characteristics.

Hardware-based characteristics, such as clock skew [12] or radio frequency signature [13], are also used to uniquely identify a network device. However, these works require sophisticated signal processing with additional hardware and also usually take an extended period to fingerprint devices, which are not applicable in the real-time BLE communication. [14] uses the inter-arrival time of packets sent by a device for a specific type of traffic (e.g., SCP, ICMP, etc.) to implement device-type identification. But the particular type of traffic cannot be obtained in BLE communication thereby

2155-6814/18/$31.00 ©2018 IEEE
DOI 10.1109/MASS.2018.00047

impeding usage of such an approach. The gap in research on fingerprinting-based authentication in BLE motivates us to design a secure communication framework that utilizes the unique protocol features in BLE to authenticate the devices irrespective of the location or environment.

In this paper, we propose *BF-IoT*, the first secure communication framework in BLE-based IoT networks via fingerprinting technique. In summary, our contributions break down into the following aspects:

- We dig into the BLE protocol stack and explore the unique features set from the link layer and ATT/GATT service layer in BLE. Based on the cross-layer transmission characteristics, *BF-IoT* is able to fingerprint the IoT devices and implement the device authentication.
- *BF-IoT* provides two-phase defense mechanisms across devices' work lifecycle: before connection establishment and during connection establishment, which enhances the accuracy of authentication and further secure the IoT networks. Compared to the existing approaches, *BF-IoT* can efficiently fingerprint and authenticate devices in a relatively short time.
- We utilize a machine learning based model to differentiate and authenticate IoT devices in a real-time way. We build a customized system and work around with a dozen of typical BLE-based IoT commodity devices. The extensive evaluation shows that our fingerprinting technique can reach approximately 100% accuracy thereby validating the effectiveness of our mechanism.

II. Adversary model

In this work, the system in consideration is deployed in typical BLE-based IoT network (such as a smart home, smart city, industrial plant, forest fire monitory service, etc.) where the BLE devices are connected to the outside world through a common IoT gateway. The open and broadcast nature of wireless communication enables an adversary to observe the data transfer of other devices in the connected network. The weak common security mechanism used in these gateways exposes the network to different malicious attacks [15], [16]. These devices may leak the common security credential such as PIN code during connection establishment [17], [18]. We assume that IoT devices may be benign or malicious when they initially connect to the system. An adversary can attack specific service at a precise point, or they can be corrupted by malicious software to target common vulnerabilities in the connected network devices. Some existing work can be found that demonstrates how to spoof a BLE device [19]–[21].

Device spoofing can have serious implications for network security. Also, by getting control of one device, an adversary can exploit the weak security requirements of the network behind the firewall and attack other devices. The malicious device may have intentions such as sniffing data or security credentials from other devices, compromising other devices in the network by probing them with well-known vulnerabilities or injecting false information into the system. A malicious device can spoof the identity of another trusted device inside a network to get access to secured data connection that other untrusted devices cannot have.

Our goal is to design a wireless communication framework to restrict the unauthenticated devices from connecting to the network via IoT gateway so that an adversary is unable to connect to a secured application or communicate with other devices by exploiting a spoofed ID of a trusted device. Continuously monitoring cross-layer data transmission is to be performed in order to detect malicious activities before and during connection establishment. In the next section, we present how devices can be identified based on their transmission signature.

III. System Architecture

The *BF-IoT* works by continuously sniffing transmission characteristics of devices from multiple protocol layers in BLE. The *BF-IoT* employs two-phase authentication mechanisms: before connection establishment and during connection establishment. Before discussing the actual implementation, this section provides the overview of the *BF-IoT* system components and its characteristics.

A. System components

1) IoT gateway: IoT gateway is the entity that both connects to the cloud and the peripheral devices. The major function of IoT gateway is to establish links with peripheral devices for collecting data and then send to the cloud for the usage of applications. IoT gateway can be a smartphone, smart router or a hub like the Samsung SmartThings at a smart-home.

2) Gatekeeper: Gatekeeper is a software running and deployed inside IoT gateway. It monitors all the packets from multiple protocol layers in BLE and sends the required and compressed data to the cloud in a specific format after preprocessing. Gatekeeper also receives the authenticate decision from the cloud and blacklist or whitelist devices and grants network access permissions to devices.

3) Fingerprinting engine: Fingerprinting engine is the key component in *BF-IoT* and deployed in the cloud. It analyzes the received transmission fingerprints to determine the authenticity of the source device. The decision process is based on the recorded fingerprints in the whitelist and blacklist databases. The ultimate decision is then delivered to the corresponding IoT gateway to execute.

4) Whitelist and blacklist database: In *BF-IoT*, we create a database to store the fingerprints of whitelisted and blacklisted devices which are generated by the fingerprinting engine. In our framework, whitelisted devices are the comparably small list of trusted and secured devices that a network frequently connects. A device is enlisted as a blacklisted device when it does not show any fingerprint matching with the whitelisted devices. Note that, blacklisted devices are not known, and we build their fingerprints database online/runtime. We use this fingerprints database of blacklisted devices for avoiding later spoofing attack by the known compromised devices and refrain any connection with those devices.

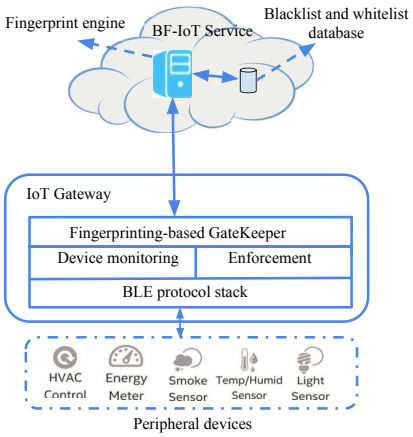


Fig. 1: *BF-IoT* system architecture

B. System design

The architecture of *BF-IoT* is illustrated in Figure 1. The fingerprinting procedures are described as following:

(1) ***Preliminary fingerprints generated***: For each newly installed device, the *BF-IoT* trusts the device that has been granted access by the network operator. *BF-IoT* keeps sniffing the link layer and service layers packets and extracts the corresponding features to generate the fingerprints for the trusted device. Then the device is added to the whitelist and its fingerprints are also stored in the cloud. The fingerprints are subsequently leveraged to authenticate the device for the future connection.

(2) ***First phase authentication***: *BF-IoT* provides two-phase procedure to enhance the authentication mechanism. The first phase is executed before link establishment. The Gatekeeper at the gateway keeps monitoring the advertisement packets from the link layer and obtains the first phase fingerprints of the device. Fingerprinting engine checks the fingerprints with the whitelist database. If it finds a match with an entry in the whitelist, the Gatekeeper allows the device to advance to the second phase for further authentication. Otherwise, the Gatekeeper forestalls all the connections with that device and adds the device's fingerprints to the blacklist if it is not on the blacklist yet.

(3) ***Second phase authentication***: In the second phase, the Gatekeeper attempts to establish a connection with the peripheral device. It monitors all the service layer packets received during the procedure. Fingerprinting engine computes the service layer fingerprints and compares with the whitelist database for further authentication. If the fingerprints of the device match with an entry in the whitelist, the device is authenticated and allowed to establish a connection with the IoT gateway for data transmission. Otherwise, the device is blocked and refrained from any further connection with IoT gateway. Meanwhile, the device is added to the blacklist. The fingerprints of devices in the blacklist are utilized for identifying the untrustworthy devices that frequently attempt to establish the link with the IoT gateway in the future. The abnormal behavior of blacklisted devices could be reported to the network operator.

(4) ***Spoofing attacks***: If a malicious device imitates the behaviors of trusted devices and attempts to connect to the IoT gateway and makes malicious activities to the IoT networks, it has to bypass dual authentication procedures. The extracted features set from multiple layer packets in BLE are unique and hard for an attacker to mimic all the behaviors that the trusted devices perform before. If any malicious behavior is detected, an appropriate alert is sent to the IoT gateway and cloud. IoT gateway will prevent the device from connecting the networks, and the connections with the corresponding applications will be terminated off immediately. The device's fingerprints will be recorded in the blacklist for future authentication.

C. System characteristics

In summary, *BF-IoT* poses the following characteristics:

i) It guarantees the authenticity of a device in a two-phase mechanism: before link establishment and during the connection establishment.

ii) It uses cross-layer traffic characteristics in BLE to enhance the accuracy of fingerprinting devices, and then detect insecure or compromised devices in the network.

iii) The authentication mechanism can be easily integrated into any existing IoT framework and allows vendors to blacklist and whitelist devices from particular actions or network permissions.

iv) The network-flow based approach can authenticate devices in a fast and real-time manner with lower overhead.

IV. BLE DEVICE AUTHENTICATION

In this section, we study the BLE protocol stack and investigate the unique cross-layer features set in BLE. A machine learning based classifier is used for fingerprinting and therefore authenticating the devices. The proposed approach can distinguish between different devices (e.g., a smart bulb and a Fitbit) and between same devices from two different manufacturers, but it is somewhat difficult to distinguish two devices of the same type from the same manufacturer.

Actually, the adversary model we are considering is that attacker attempts to use the customized hardware to imitate the behaviors of trusted devices and bypass the two-phase authenticate mechanism to gain trust from the IoT networks. The proposed approach may not be able to detect the attacker who purchases the same type of device from the same manufacturer and use exactly the same device hardware to execute spoofing attack. In our scenario, the attacker can learn and imitate the behavior of trusted devices. But it is challenging for the attacker to guess and infer what the exact device hardware it is among billions of IoT devices in the world. Our goal is to find the unique fingerprinting features that are hard to mimic for the attacker, which is more practical in the real world. In the future, we will incorporate the application layer data into

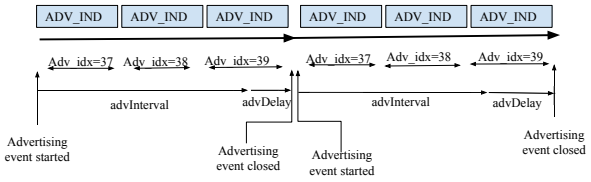


Fig. 2: Advertising event with only advertising PDUs

our framework to distinguish two devices of the same type from the same manufacturer.

A. BLE overview

BLE is a wireless technology specially designed for low power devices that operate in the 2.4GHz ISM band [5]. In this band, BLE has 40 channels, where each channel is 2 MHz wide, and they are numbered from 0 to 39. BLE uses 3 channels (37, 38 and 39) for advertisement, where BLE peripheral devices transmit advertisement packets to announce their presence and establish connections with the central devices. The rest of the channels are used for data transmission between peripheral and central devices. Note that, BLE uses frequency hopping mechanism to transmit data packets at different channels by using a pseudo-random sequence that is known to both peripheral and central devices. BLE is a connection-oriented peer-to-peer communication technology where peripheral and central devices communicate in a peer-to-peer fashion. BLE protocol stack is designed to support such peer-to-peer communication. In the following subsections, we present transmission feature extraction from multiple layers in BLE protocol stack.

B. Link layer feature

Peripheral devices may stay in sleep mode in the most time so as to save energy. But they must periodically broadcast advertisement packets for being able to discover once data transmission is required. IoT devices send the advertising packets with distinct patterns. Upon discovering a peripheral device, a central device decides whether to initiate a connection for data transmission.

The advertisement event is the smallest unit that is used for transmitting advertising packets. In Figure 2, three PDUs are sent on three advertising channels within one advertisement event. The advertising event could be terminated after the last advertising PDU is sent or the advertiser may close an event earlier to accommodate other functionality. The time (T_advEvent) between the start of two consecutive advertising events is computed as follows for each advertising event:

$$\text{T\_advEvent} = \text{adv\_Interval} + \text{adv\_Delay}$$

The adv_Interval is an integer multiple of 0.625 ms in the range of 20 ms to 10.24 s. The adv_Delay is a pseudo-random value with a range of 0 ms to 10 ms generated by the link layer for each advertising event.

Feature extraction: The advertising patterns from IoT devices are distinct and accommodate to hardware and software-based characteristics. 1) *Advertising event interval*: devices are required to transmit data in different frequency based on their hardware and functionalities characteristics. Small advertising event interval means the central device can fast discover them once the data transmission is requested. Some devices that synchronize data with central device few times a day may need relatively large advertising event interval so as to save energy. 2) *Advertising channel sequence*: Advertising PDUs are sent via three advertising channels within each advertising event. There are total six possible combinations of channel sequence: (37, 38, 39), (37, 39, 38), etc. IoT devices select one of the combinations to transmit advertising packets in round-robin fashion which decreases the probability of packet transmission collision from multiple devices in the meantime. 3) *Advertising delay distribution*: In the typical IoT environment, there are many IoT devices that periodically send advertising packets in the meantime. In order to minimize the possibilities of transmission collision in the same channel, the advertising events are perturbed in time using the adv_Delay. The random adv_Delay can be generated by using different probability distributions that IoT devices select.

The adverting event interval, channel sequence, and random delay distribution constitute the unique advertising pattern for different IoT devices. Such pattern can be utilized as the fingerprints of the first phase device authentication. The first phase authentication is not a one-off but a continuous procedure. *BF-IoT* keeps monitoring the link layer packets of ambient devices and compares their advertising patterns with the fingerprints recorded in the database. If the device passes the preliminary first phase authentication, it is allowed to enter into the second phase authentication procedures in order to establish a trusted data transmission link with central device.

C. ATT/GATT layer feature

In BLE protocol stack, the ATT layer allows a device to expose certain pieces of data or attributes to another peer device or central device. In typical IoT networks, peripheral devices usually sense data and send to the central device to provide associated services and functionalities. The state or data is exposed as one or more values called attributes. All attributes have handles, which are used to address an individual attribute. The attributes also have a type described by a universally unique identifier (UUID). The UUID determines what the attribute value means. The length of attributes varies from 0 to 512 bytes.

The ATT layer protocol manages discovering, reading, and writing attributes on a peer device. The ATT defines the communication protocol between two devices playing the roles of server and client respectively on top of a dedicated L2CAP channel. The server, usually a peripheral device, maintains the data as one or more attributes and exposes them to a client. The client, usually a central device, fetches the attributes using the handle from one or more servers. The client can access the server's attributes by sending requests, which trigger response messages from the server.

The GATT defines a framework that uses the ATT for the discovery of services, and the exchange of characteristics

TABLE I: Attribute Protocol PDUs

Function Category	Related Operation	Attribute Opcode
Find Information:	Find Information Request/Response	0x04/0x05
	Find By Type Value Request/Response	0x06/0x07
Read Attributes:	Read By Type Request/Response	0x08/0x09
	Read Request/Response	0x0A/0x0B
	Read Multiple Request/Response	0x0E/0x0F
	Read by Group Type Request/Response	0x10/0x11
Write Attributes:	Write Request/Response	0x12/0x13
	Write command	0x52
Server Initiated:	Handle Value Notification	0x1B
	Handle Value Indication/Confirmation	0x1D/0x13
Error Handling:	Error Response	0x01

between peripheral and central devices. A characteristic is a set of data which includes a value and properties. Characteristics are used to expose and exchange data between paired devices. Control information can also be formatted as a characteristic passing from central device to peripheral device.

IoT devices may provide different services based on their functionalities. During link establishment, the central device sends requests to probe the services list of device. We use $\{S_1(C_1, ..., C_i), S_2(C_1, ..., C_j), ..., S_m(C_1, ..., C_k)\}$ to denote the services list with corresponding characteristics. For instance, a smart watch can provide multiple services, such as temperature service and time service.

Feature extraction: After investigating the ATT/GATT layer protocol, we determine the unique features set for the fingerprinting. During connection establishment, the peripheral device has to communicate with the central device to exchange the device state and link parameters. The transmitted ATT/GATT packets also contain distinctive attributes and services list depending on the functionalities of peripheral devices. Assume the exchanged packets sequence Υ is denoted by $\{P_\mu^1, P_\nu^2, P_\mu^3, P_\nu^4, ..., P_\mu^i, P_\nu^{i+1}\}$, where P_μ denotes the packet sent by the central device and P_ν represents the packet sent by the peripheral device. P_μ and P_ν can be a request packet or response packet, and both of them are exactly timestamped. Table I provides the PDU type pairs used in ATT/GATT layer protocol. Our goal is to extract the features that are related to peripheral devices and can be used for fingerprinting.

Due to the differences in hardware, driver, and application software, the transmitted packets sequences for link establishment are different among IoT devices. The *packet size, packet type, total packets transmitted, number of packets transmitted for different type of function, burst rate, burst time point, etc.* in exchanged packets sequence Υ pose unique session-level pattern for different IoT devices. Moreover, we filter the packets that is requested by peripheral device and responded by central device and build the response time sequence $\{R_{t1}^1, R_{t1}^2, R_{t2}^3, R_{t2}^4, ..., R_{t_j}^i, R_{t_j}^{i+1}\}$, where $R_{t_j}^i$ represents the response time of i^{th} packet pair (P_μ, P_ν) requested by central device and responded by peripheral device, and t_j represent one of the request/response type pair in Table I. The response time varies with request type, device processing power, and other device-specific characteristics. Therefore, IoT device has the unique pattern for response time sequence which could be utilized to fingerprint the devices.

During link establishment, some peripheral devices periodically send packets back to central devices in a proactive way without the request of central devices. According to the BLE specification, a peripheral device can send two types of unsolicited messages that contain attributes: notifications, which are unconfirmed; and indications, which require the client to send a confirmation. The unsolicited packets sequence can be denoted as $(P_\psi^1, P_\psi^2, ..., P_\psi^i)$. Some peripheral devices may send one unsolicited packet at every interval. The interval of sending such packets also differs with devices. Other peripheral devices may periodically send consecutive packets in a burst way. These characteristics also constitute the unique pattern of unsolicited packets sequence which can also be used for fingerprinting devices.

D. Classifier for device fingerprinting

After exploring the features, we design a machine learning based model to implement our fingerprinting mechanism. We use $\{f_1, f_2, .., f_n\}$ to denote the features we extract from BLE-based IoT devices. $\{E_1, E_2, ..., E_m\}$ represents a set of authorized IoT devices in the whitelist database. We have two data sets: training data set D_{train}: $\{d_1, d_2, ..., d_p\}$ and test data set D_{test}: $\{d'_1, d'_2, ..., d'_q\}$. Each date subset d_i or d'_i is generated only by one device that contains all the n features. In training data set D_{train}, we know that each data subset is generated from which device. Our machine learning task is to map any data subset d'_i in D_{test} to a specific device E_j that most probably generates it. We treat this as a classic multi-class classification problem in machine learning.

We use *Random Forest* [22], [23] supervised machine learning algorithm for model training. *Random Forest* is a classical classification algorithm that combines decision tree induction with ensemble learning. We use the dataset D_{train} to train a classifier C, which can capture the characteristics of every authorized device type. Then we use the classifier C to classify the new unlabeled data in D_{test}. Classifier C can output a vector of posterior probabilities $P_{d'_i} = \{p_{E_1}, p_{E_2}, ..., p_{E_m}\}$. Each probability p_{E_i} denotes the likelihood that the data is produced by device E_i. If there exits any p_{E_i} is larger than the predefined classification threshold θ, then we think the data is produced from authorized device E_i. Otherwise, the data set is considered as an 'unknown' device, which may be added to the blacklist. The classifier C can be continuously applied to new unlabeled data for device fingerprinting.

E. Two-phase device authentication

In the first-phase, ***BF-IoT*** continuously monitors the sniffed packets sequence in the environment and computes their advertising patterns $\{PT_1, ..., PT_i\}$ for devices $\{D_1, ..., D_i\}$ respectively in the whitelist. By matching the patterns that are stored in the database, ***BF-IoT*** can detect if the device is deviating from the right path, and then can determine if the device is in a healthy state or is being compromised. Assume the adversary sniffs the advertising packets sequence

from device D_i and learns the advertising pattern. Then the adversary forges the same MAC address of the device and attempts to bypass the first-phase authentication via mimicking the device's behavior and sending the advertising packets using the same pattern. Actually, it is arduous for the adversary to exactly mimic the advertising pattern even if it learns the pattern. Usually, the device has its own clock drift pattern due to the hardware difference of crystal oscillator. Even for the identical devices from the same manufacturer, there is still a subtle change in the manufacturing procedure. Therefore, the adversary may unwittingly stray from the primitive advertising pattern. On the other hand, *BF-IoT* monitors the devices' state in a real-time way, if two devices almost behave the same way, *BF-IoT* can effectively detect them and stop the attacks. Furthermore, the first-phase is pre-connection authentication. As an auxiliary authentication of the second-phase, it makes no difference if the adversary ultimately bypasses the first-phase authentication. Then the adversary has to think how to pass the second-phase authentication.

The adversary cannot do any malicious activities although it passes the first-phase authentication. It has to continue to pass the second-phase thereby establishing a real link with a central device. We cannot prevent the adversary from sniffing the ATT/GATT service layer packets, and the adversary can also learn the pattern that how the trusted peripheral communicates with the central device. Here the communication scheme is not a secret. But even if the adversary knows the secret, it is hard to imitate the time series based packets sequence. For the read/write requests, the peripheral devices have different response time pattern due to both the hardware and driver based characteristics. The request handle way and process power are diverse among different devices, which constitutes the unique pattern of time-stamped packets sequence. The adversary has to figure out the detailed hardware and driver information and also exactly clone the way that requests are processed if it wants to generate the same fingerprints. But it is challenging in the real world.

The only way that can bypass the second-phase authentication is that the adversary exactly uses the same device from the same manufacturer, which could generate the same pattern. Then the adversary transmits the identical packet sequence following the same time series but injects the malicious content inside the packet. However, there are millions of IoT devices deployed in the world. By sniffing the packets, the adversary cannot exactly infer the device information and purchase the same device in the market. Therefore, we do not seek to identify two devices from the same type and manufacturer in this paper. Ultimately, if the device passes the two-phase authentication, it is allowed to establish a trusted link with central device and begins to transmit the real data. Otherwise, it will be prevented from connection and added to the blacklist.

V. Evaluation

A. System implementation

In order to validate the effectiveness of our secure IoT communication framework, we build a customized system

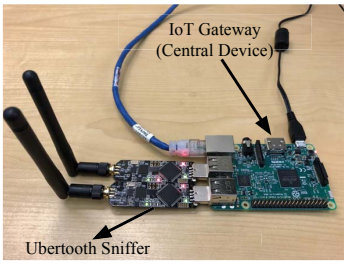


Fig. 3: *BF-IoT* system platform

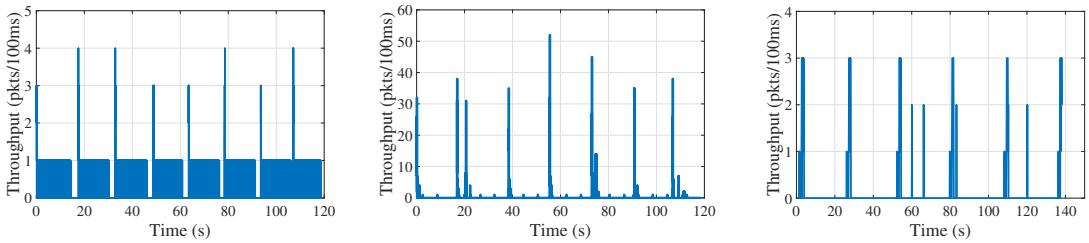
that consists of a Raspberry Pi 3 and two Ubertooths [24]. Raspberry Pi 3 is built on the latest Broadcom 2837 ARMv8 64 bit processor. Raspberry Pi 3 supports more powerful external USB devices and comes with built-in WiFi and BLE connectivity. Ubertooth is an open source wireless platform suitable for BLE development. Ubertooth is equipped with CC2400, a single-chip 2.4 GHz RF transceiver. Ubertooth has a capable BLE sniffer and can sniff data traffic from multiple layers in BLE.

The Raspberry Pi 3 works as an IoT gateway that communicates with the cloud and peripheral devices. Two Ubertooths are connected to the gateway via USB, which is displayed in Figure 3. The processor (LPC175x) in Ubertooth decodes the raw data received from the CC2400 RF transceiver and sends to the queue in USB buffer. Then Raspberry Pi 3 reads packets from the queue in the USB buffer. In order to exactly capture the time that Ubertooth receives packets, we modify the Ubertooth firmware code [25] to time-stamp the packet with 100 ns resolution once the preamble of the packet is detected.

We implement the functionalities of IoT gateway in the Raspberry Pi 3. The IoT gateway collects the data from peripheral devices and extracts and compresses the required data and delivers them to the cloud. IoT gateway also works as the Gatekeeper to control the network access permission of the devices. The IoT gateway is connected with a variety of commercial IoT devices. The devices are comprised of different types of devices, such as smart bulb, smart watch, heart rate monitor, etc, and also the same type of devices from different manufacturers, such as smart bulbs from Avea and iLink companies. We usually have two for each device type. This diversity of devices can help us thoroughly examine the performance of our fingerprinting mechanism.

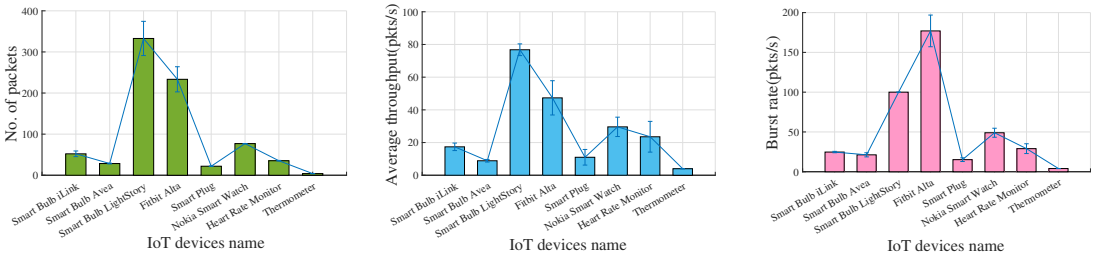
B. Performance analysis of cross-layers features

1) Link layer: We use Ubertooth to capture BLE advertising packets in the link layer. The Ubertooth is responsible for capturing BLE advertising packets from the link layer. The Ubertooth sniffs the advertising packets on channel 37, 38 and 39. The link layer packets are sniffed for the ambient IoT devices and used to constitute their advertising patterns. An advertising event can be one of the following types: connectable undirected event, connectable directed event, non-connectable undirected event and scannable undirected event. For each device, we run our system for five days and collect



(a) Heart Rate Monitor (b) Fitbit Alta (c) Smart bulb Avea

Fig. 4: Traffic pattern for duplicated session establishments.



(a) No. of packets sent (b) Average throughput (c) Burst rate

Fig. 5: Features from the packets sequence of session establishments in ATT/GATT layers.

TABLE II: Advertisement interval for different devices

Device name	Advertising event interval (s)	Standard deviation
Fitbit Alta 1:	1.29002800418	0.00104526798899
Fitbit Alta 2:	1.28996685678	0.00103673292851
Smart bulb Avea:	0.104050761475	0.00218257087699
Smart bulb iLink:	0.0814239081008	2.55710826474e-05
Smart bulb LightStory:	0.497569043441	6.36738129605e-05
Smart plug 1:	0.103930876001	0.00217028685053
Smart plug 2:	0.104025252545	0.00215880616127
Garmin:	1.28997122261	0.00294536429203
Ambient device 1:	0.183192135081	5.53702230595e-06
Ambient device 2:	0.0315230156198	0.00289863032782

a large number of experimental data about advertising packets. By observing the advertising packets, most IoT devices adopt connectable undirected advertising event type, and they transmit advertising packets ADV_IND PDU.

Table II exhibits the measured advertising event interval from different devices. According to the table, we can distinguish different types of devices via observing the adverting interval, such as sports bracelet (Fitbit Alta and Garmin) with smart bulb (Avea, iLink, and Lighstory). Concerning the same type of devices, it is still able to show the recognizable difference, which can be observed for three smart bulbs from different manufacturers Avea, iLink, and LightStory. Even for the two devices of the same model, there is still a subtle difference that we can leverage to distinguish them, which is exposed from two Fitbit Altas and Smart Plugs. Therefore, the feature in the advertising pattern can be regarded as part of fingerprints for the first phase authentication.

2) ATT/GATT layer: Based on the features extracted from the ATT/GATT service layer, we have to precisely collect all the transmitted data packets type and exactly record their timestamps. Applications are written and installed in the IoT gateway to communicate with the targeted BLE devices. We principally collect the session-level network traffic from link establishment stage to authenticate the IoT devices in the second phase. The reason is that all types of IoT devices may have different functionalities but must have the same session establishment activity. For instance, smart bulbs usually have the on-off action that we can operate, but smartwatches do not have such function. We seek to adopt the common activities across all IoT devices to extract the featured data to do fingerprinting. For each peripheral device, the central device(IoT gateway) establishes a connection with it via the corresponding installed the application, waits for approximately 20s and does not do any additional action, and disconnects the connection with the peripheral device. We repeat this process 50 times and collect all the packets transmitted. After collecting the data for all IoT devices, we use the extracted featured data and implement and validate our fingerprinting mechanism.

Figure 4 exposes the traffic patterns of Heart Rate Monitor, Fitbit Alta, and Smart Bulb Avea which are extracted from the exchanged packets sequence. As shown in the figure, the traffic patterns are obviously distinct from each other. The number of packets sent, average throughput and burst rate in Figure 5 demonstrate that different devices have unique transmitted packet sequence, which can be proved as a good feature of fingerprinting. After analyzing the response time sequence of requests, we find that the IoT devices have the

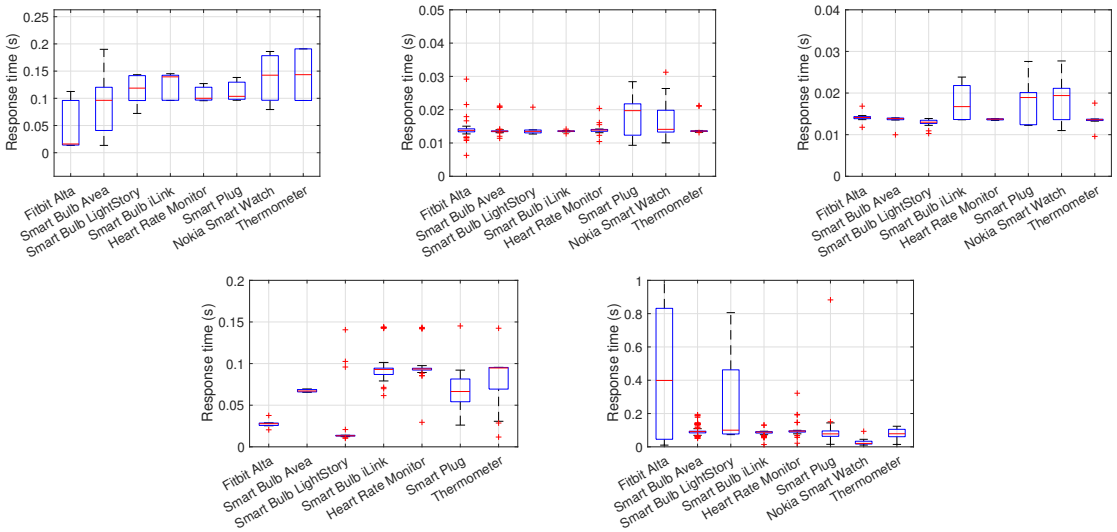
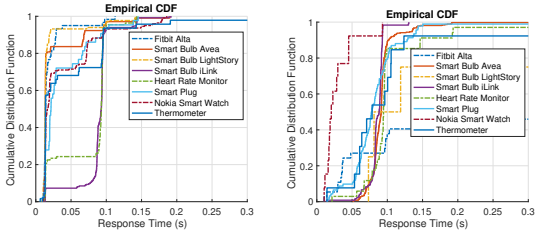


Fig. 6: Distribution of response time for read and write operations: (a) Read by group type request (Opcode: 0x10), (b) Read by type request (Opcode: 0x08), (c) Find information request (Opcode: 0x04), (d) Read request (Opcode: 0x0A), (e) Write request (Opcode: 0x12).



(a) CDF of response time for overall read requests.

(b) CDF of response time for overall write requests.

Fig. 7: Response time pattern for read and write requests.

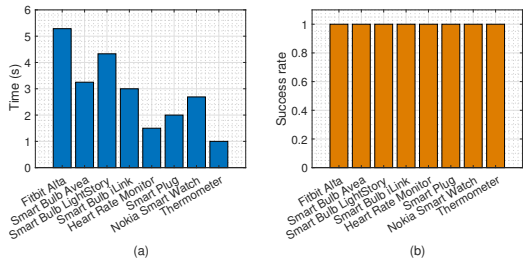


Fig. 8: (a) Time cost for the second phase authentication. (b) Success rate of device fingerprinting.

diverse response time of requests due to the different request handle way and device processing power, which is manifested in Figure 6. Even for the same type of requests from the same device, the response time is also showed to be different due to the varying content of response request. Compared to read requests, the response time of write requests can better reflect the device characteristics. Figure 7 shows that CDF of response time for overall read and write requests. The obvious different distributions of request response time can be observed in the figure, which can constitute the unique response time sequence of requests used for the fingerprinting of devices.

C. Fingerprinting performance

Conclusively, we combine all the mentioned features in the link and ATT/GATT service layers and use the machine learning model proposed in Section IV-D to fingerprint the IoT devices we have. The *BF-IoT* system was running for several days and collected enough data to train the model and generated the fingerprints for the trusted devices. Then we added 80% of the devices and their corresponding fingerprints to the whitelist and added the rest of 20% devices and their fingerprints to the blacklist. Whereafter, we deployed the IoT devices to the environment and let them attempt to establish data links with the IoT gateway. The IoT gateway sniffed the cross-layer traffic in a real-time way and tried to authenticate the devices. Figure 8(b) shows the accuracy that we successfully authenticated the devices in the whitelist and blocked the devices in the blacklist. *BF-IoT* can achieve approximately 100% fingerprinting accuracy for all current IoT devices, which demonstrates the efficiency of our fingerprinting mechanism. Figure 8(a) shows the time cost of our authentication mechanism, which is mostly controlled within 6 seconds. Furthermore, our authentication is a real network-flow based mechanism which does not request additional cost.

VI. Related Works

The security issues are major challenges in IoT networks due to its massive deployment. The IoT networks are much

vulnerable to spoofing attacks. A malicious device can forge the MAC or other network identities of a trusted device to either compromise IoT device or injects false or tampered information into the network. So the technique to identify IoT devices based on network identifiers such as MAC address is impractical. Some researchers investigated the transmission characteristics as a unique identification for wireless devices. Radio frequency identification (RFID) is one of the earliest use cases of using wireless transmission characteristics to fingerprint devices. [26] exploits the fact that an RFID tag presents a unique frequency response for different frequencies. Physical properties of these tags can be used to do fingerprinting with minimum power response. But it requires sophisticated hardware and massive digital signal processing which can not be done by commodity devices. [10] uses the link layer data transmission characteristic to fingerprint for WiFi devices. This approach statistically analyzes the usual packet transmission characteristics of different devices. But these WiFi-related approaches do not target on BLE transmission characteristics and may not be applicable for BLE device fingerprinting.

Hardware-based fingerprinting mechanisms is also investigated in IoT networks. Physical unclonable function [27] can be used to uniquely identify a device. The time skew induced by chip characteristics is another approach to identify different types of devices. BlueID [28] uses internal clock skew to classify devices Since the internal clock of each device is hard to forge, devices can not spoof this identity. Actually, these approaches mostly cost massive time to fingerprint the devices. This constraint prevents these techniques to be used in cases where fingerprinting is needed in a timely manner such as authenticating with a new device. In our work, we aim at using cross-layer feature based fingerprinting scheme that attains higher accuracy while requiring significantly smaller dataset obtained in a relatively small amount of time.

VII. Conclusion

This paper proposes *BF-IoT*, a novel defense framework against spoofing attacks in IoT networks. Unlike the conventional device ID based security framework, *BF-IoT* augments the authentication and device identification using unclonable transmission characteristics. It identifies IoT devices using their observable cross-layer transmission characteristics. The devices are screened in a twofold manner when connecting a secured application. Before establishing a connection, *BF-IoT* observes link layer transmission signatures and checks them with a global blacklist database for possible spoofing attacks. During data connection, *BF-IoT* monitors service layer data transmissions to find anomalies and intrusive behaviors. To evaluate the performance, we gathered data from some BLE based IoT devices in multiple network layers and trained the multi-class classifier. The experiments show that *BF-IoT* is able to fingerprint each device uniquely and detect identity spoofing. In the future, we plan to extend our framework to fingerprint devices in any of 6LoWPAN standards such as ZigBee, WiFi, and conventional Bluetooth.

VIII. Acknowledgement

This research was supported in part by the Army Research Office (ARO) through the grant W911NF-16-1-0224.

References

[1] "IDC Forecasts Worldwide Spending on the Internet of Things to Reach $772 Billion in 2018." https://www.idc.com/getdoc.jsp?containerId=prUS43295217, Dec 7, 2017.
[2] "Gartner Says a Typical Family Home Could Contain More Than 500 Smart Devices by 2022." https://www.gartner.com/newsroom/id/2839717, Sep 8, 2014.
[3] R. H. Weber, "Internet of things–new security and privacy challenges," *Computer law & security review*, vol. 26, no. 1, pp. 23–30, 2010.
[4] E. Fernandes, A. Rahmati, J. Jung, and A. Prakash, "Security implications of permission models in smart-home application frameworks," *IEEE Security & Privacy*, vol. 15, no. 2, pp. 24–30, 2017.
[5] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," tech. rep., 2006.
[6] "Bluetooth technology." http://www.bluetooth.com.
[7] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.
[8] S. Raza, A. Slabbert, T. Voigt, and K. Landernäs, "Security considerations for the WirelessHART protocol," in *Emerging Technologies & Factory Automation (ETFA)*, pp. 1–8, IEEE, 2009.
[9] M. Ryan, "Bluetooth: With low energy comes low security," in *USENIX Workshop on Offensive Technologies*, 2013.
[10] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting," in *USENIX Security Symposium*, pp. 16–89, 2006.
[11] J. Cache, "Fingerprinting 802.11 implementations via statistical analysis of the duration field," *Uninformed. org*, vol. 5, 2006.
[12] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Tran. on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.
[13] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *ACM MOBICOM*, 2008.
[14] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A technique for physical device and device type fingerprinting," *IEEE Tran. on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, 2015.
[15] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: the insecurity of 802.11," in *ACM MobiCom*, 2001.
[16] S. Xiao, W. Gong, and D. Towsley, "Secure wireless communication with dynamic secrets," in *IEEE INFOCOM*, 2010.
[17] J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, "Proximity based IoT device authentication," in *IEEE INFOCOM*, 2017.
[18] E. Ronen and A. Shamir, "Extended functionality attacks on iot devices: The case of smart lights," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 3–12, IEEE, 2016.
[19] "How to Hack Bluetooth ." https://null-byte.wonderhowto.com/how-to/hacks-mr-robot-hack-bluetooth-0163586/.
[20] "Spoofing a Bluetooth device." https://haxf4rall.com/2016/05/11/spoofing-a-bluetooth-device/.
[21] J. Dunning, "Breaking Bluetooth By Being Bored," in *DefCon*, 2010.
[22] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomForest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
[23] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis," in *ACM Symposium on Applied Computing*, 2017.
[24] "Ubertooth." http://ubertooth.sourceforge.net/.
[25] "Ubertooth source code." https://github.com/greatscottgadgets/ubertooth.
[26] S. C. G. Periaswamy, D. R. Thompson, and J. Di, "Fingerprinting RFID tags," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 938–943, 2011.
[27] S.-Y. Park, S. Lim, D. Jeong, J. Lee, J.-S. Yang, and H. Lee, "Pufsec: Device fingerprint-based security architecture for internet of things," in *INFOCOM*, pp. 1–9, IEEE, 2017.
[28] J. Huang, W. Albazrqaoe, and G. Xing, "BlueId: A practical system for bluetooth device identification," in *IEEE INFOCOM*, 2014.