Data Leakage in Smart Homes

Steven M. Beyer
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
Email: steven.beyer@afit.edu

Barry E. Mullins
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
Email: barry.mullins@afit.edu

Scott R. Graham
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
Email: scott.graham@afit.edu

***Abstract*—As smart home technologies become more popular, the increased prevalence of Bluetooth Low Energy (BLE) and Wi-Fi devices in the home necessitates the need for consumers to be aware of the privacy information these devices inadvertently broadcast and take measures to defend against leakage. These protocols are increasingly used in a range of Internet of Things (IoT) devices such as security cameras, locks, and sensors. Two characteristics of these devices leave them vulnerable to privacy leakage: (i) these devices continuously broadcast unencrypted information, such as Media Access Control (MAC) addresses (Wi-Fi) or device name (BLE), which anyone with a properly-tuned receiver can observe, and (ii) the implementation of BLE security is left up to the developer and often results in essentially no link layer authentication or encryption.**

This work shows how vulnerabilities enable an eavesdropper to collect data from devices without being connected to the smart home environment. The information can then be used to identify devices, track user's movements, and deduce events such as when a door is opened or when a light is turned on.

To demonstrate these capabilities, we developed a voice activated digital assistant and IoT architecture by integrating a variety of off-the-shelf Wi-Fi and BLE devices with Apple's home automation application, HomeKit. Furthermore, we created a device classifier and pattern-of-life analysis tool, CITIoT (Classify, Identify, and Track Internet of Things), that was able to classify devices in the smart home architecture with a mean accuracy of 94%, identify 95% of all events over a 5 day period, and track when users are home or away with 100% accuracy. We also show how it is possible to piece together the sniffed information to track occupants of a house and crack a Bluetooth lock. Lastly, we present how a user can take advantage of the limitations to this approach and security recommendations to defend against these vulnerabilities and create a more secure smart home environment.

I. Introduction

In recent years, smart home devices have become one of the most popular categories in the Internet of Things (IoT) accounting for $3.5 billion of a $292 billion industry; over 29 million smart home devices are expected to ship in 2017, a 63 percent increase over 2016 [1]. Wi-Fi and Bluetooth Low Energy (BLE) are two of the primary protocols used in these devices and are commonly implemented in security cameras, locks, medical devices, sensors, and a myriad of other devices. With the increasing prevalence of BLE and Wi-Fi devices in the home, consumers must be aware of the information these devices inadvertently broadcast and what kind of privacy data an outside observer can infer.

This work contributes to the field of IoT security, specifically privacy within a smart home, by illustrating how devices leak data and demonstrating how users can prevent leakage. In doing so, we make four principal contributions:

Smart home architecture. To analyze IoT data leakage in the wild, we provide a realistic smart home architecture that integrates Wi-Fi and BLE commercial-off-the-shelf (COTS) devices with Apple's home automation application, HomeKit.

Vulnerability analysis. We explain how an eavesdropper can use device vulnerabilities, characteristic data exchanges, and packet sizes to create a classifier able to identify components of the smart home environment.

Classification, Identification, Tracking of IoT (CITIoT). We present a tool that demonstrates four capabilities enabled by data leakage: network mapping, device classification, event identification, and user tracking.

Synthesis. We stress the importance of smart home operational security by demonstrating how CITIoT can be used to gain access to a smart home when a user is away.

The rest of this paper is organized as follows: Section II discusses background and related research in the areas of Wi-Fi and BLE security and IoT device privacy. Section III presents the smart home architecture used in creating and evaluating CITIoT. Sections IV and V describe the creation of the device classifier and how it is used by CITIoT to classify devices, identify events, and track users. Section VI reports CITIoT's accuracy on the smart home architecture. Finally, the implications of device data leakage are discussed and methods to defend against these attacks are presented in Section VII before the paper is concluded in Section VIII

II. Background and Related Research

The 802.11 wireless specification defines the physical and link layers for communication in the 2.4 GHz radio band [2]. It also defines security procedures to encrypt data within a wireless network. Even with encryption, however, Wi-Fi still transmits a subset of information in the clear within the Media Access Control (MAC) Protocol Data Unit (MPDU) data frame. Values of interest in this frame include source, destination, and wireless router MAC addresses. Also, the time, size, and Received Signal Strength Indicator (RSSI) of each packet can be ascertained by the receiver wireless network card. While these values are necessary for communication at lower levels of the Wi-Fi protocol, researchers have taken advantage of values sent in the clear to eavesdrop privacy information about users. For example, researchers have used MAC addresses and RSSI values to create location tracking

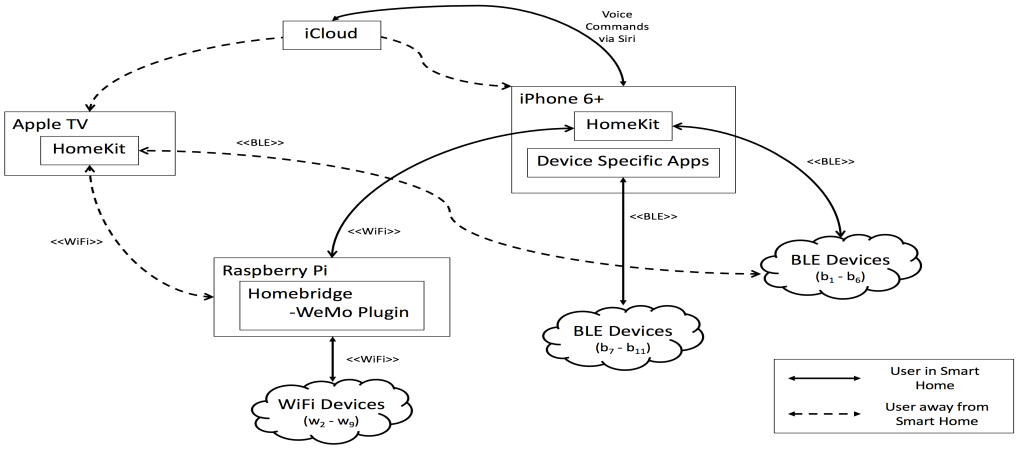

Fig. 1. Smart Home Architecture

systems on campuses [3], track crowds at mass events [4], and in Customer Relationship Management (CRM) allowing commercial businesses to track stores visited by customers [5]. Researchers in the United Kingdom were able to use frame size (FSize) and packet times to fingerprinting applications used on mobile phones [5].

The Bluetooth Special Interest Group (SIG) introduced BLE in Bluetooth Core Specification v4.0 [6]. BLE is designed to minimize power, cost, and data rate; these goals are accomplished by limiting overhead at every level of the architecture and using simple communication protocols. The BLE specification defines security procedures to encrypt the payload, generate private addresses, and provide authentication in its Security Manager (SM) [7]. However, implementation of security is left up to designers and each additional security measure contributes to increased energy consumption [8]. Poor SM implementation has left these devices vulnerable to privacy leakage and security vulnerabilities. For example, in a few recent studies focused on BLE wearable fitness trackers researchers were able to use messages sent from devices to determine activity level and gait of users [9], while another group tracked a user wearing a Fitbit up to 1,000 meters away [10]. Likewise, poorly implemented SMs enabled researchers to crack 12 BLE locks from up to a quarter mile away [11].

While research has been done in the realm of Wi-Fi and BLE privacy leakage, this is the first work providing a broad review of privacy leakage from smart home devices in the wild and methods to secure smart homes.

III. Smart Home Architecture

As shown in Figure 1, the smart home architecture includes three controller components and various connected devices. The controller components include (i) a Raspberry Pi running the Homebridge server that emulates the iOS HomeKit application programming interface (API) and exposes supported devices to Apple's HomeKit, (ii) an iPhone 6+ running Apple's HomeKit and device specific applications, and (iii) an Apple TV Generation 2 acting as a smart home hub to allow access to HomeKit supported devices while the user is away from the smart home. The communication between controllers and devices can be observed in Figure 1 and is described in the rest of this section.

The Raspberry Pi 3 Model B with Raspbian Jessie Lite version 4.9 operating system is connected to the smart home network via the on-board 802.11 b/g/n 2.4 GHz wireless chip [12]. The Raspberry Pi runs Homebridge version 0.4.14 as a systemd service, and each interaction between a controller and device is logged in the systemd journal [13]. A Homebridge module is utilized to expose Belkin devices to Apple's Homekit and is loaded into Homebridge [14]. The Apple devices communicate via the Raspberry Pi to interact with the Wi-Fi devices (devices w_2-w_9 in Table I).

The iPhone 6+ and Apple TV act as controllers in the smart home architecture and connect to devices via Wi-Fi and BLE. When the user is home, the iPhone connects to Wi-Fi devices via the Homebridge and directly to BLE devices. Some of the BLE devices are not supported by Apple's Homekit and can only be accessed through the manufacturer-provided iOS application on the iPhone (devices b_7-b_{11} in Table II). When the user is away from the smart home, the iPhone can communicate with Homekit-supported devices via the iCloud and Apple TV acting as a hub. For example, if the user is away from home and wants to access the temperature in a room, the iPhone communicates with the Apple TV via the iCloud and the Apple TV will communicate with the device in the home via Wi-Fi or BLE. This will only work with Homekit supported devices, therefore, BLE devices b_7-b_{11} cannot be accessed while the user is away from the home.

To facilitate Wi-Fi communication in the smart home architecture, a 2.4 GHz Wi-Fi access point (AP), "Prancing Pony", was setup with Wi-Fi Protected Access 2 - Pre-Shared Key (WPA2-PSK) security on channel 1. A list of devices connected to the AP can be found in Table I. The smart home devices include a camera, six outlets (four smart plugs, one mini plug, and one energy plug), and a motion sensor (w_2-w_9). These devices use the Homebridge to communicate with Apple's HomeKit on the iPhone.

For BLE communication to occur in the smart home ar-

TABLE I
WI-FI DEVICES.

ID	Manuf	Device Type	Device Name
w_1	Calix	Wireless Router	Prancing Pony
w_{2-5}	Belkin	Outlet	Switch1-4
w_6	WeMo	Outlet	Mini
w_7	Belkin	Outlet	Insight
w_8	Belkin	Motion Sensor	Motion
w_9	Belkin	Camera	NetCam
w_{10}	Raspberry Pi 3B	Computer	Pi
w_{11}	Apple	iPhone 6+	Steves-phone
w_{12}	Apple	TV 2	Apple-TV

TABLE II
BLE DEVICES.

ID	Manuf	Device Type	Device Name
b_1	Elgato	Indoor Temperature	Eve Room
b_2	Elgato	Outdoor Temperature	Eve Weather
b_3	Elgato	Motion Sensor	Eve Motion
b_4	Elgato	Outlet	Eve Energy
b_6	Elgato	Door Sensor	Eve Door
b_7	Instant Pot	Smartcooker	Instant Pot
b_8	MPow	Lightbulb	Playbulb
b_9	ZKTeco	Lock	BioLock
b_{10}	BitLock	Lock	Bike lock
b_{11}	SafeTech	Gunsafe	Gunsafe
b_{12}	Apple	iPhone 6+	Steves-phone
b_{13}	Apple	TV 2	Apple TV

chitecture, a BLE master must be present. In the smart home architecture, the iPhone and Apple TV act as masters while each of the BLE devices are slaves. A list of devices operating in the BLE can be found in Table II.

IV. BUILDING THE CLASSIFIER

This section describes the process used to build a classifier that can identify Wi-Fi IoT devices and events. To emulate the scenario in which the classifier operates, each device in Table I was activated over two 10-hour days within the smart home environment. Wireless packets were collected using an 802.11ac wireless adapter (Alfa Card AWUS036ACH) in monitor mode and an open source sniffing tool (Airodump-ng). The file captures were decrypted using the network passkey to allow analysis of traffic. These packets were then analyzed using a 4-tuple that includes packet arrival time, FSize, and the unencrypted information sent within the MPDU (source and destination MAC addresses). For device identification, these 4-tuples were sorted by destination address (i.e., when the iPhone sends an event request to a smart home device). By observing traffic over the 2-day period, clear criteria were created to classify devices into one of three categories: electrical outlet, sensor, or camera. Outlets are the only devices to receive packets with FSize greater than 600 bytes, sensors never receive a packet with FSize greater than 275 bytes, and cameras are the only devices to receive packets between 275 and 300 bytes. Therefore, the classifier measures each device by these criteria and categorizes them accordingly.

Event identification relies on successful device classification and must wait until the first step is complete. For electrical outlets, the classifier utilizes the same 4-tuple sorted by destination address to identify when a user turns on or off the outlet. From traffic analysis, outlets are controlled using a HTTP POST command with a packet size of 620 bytes when encrypted. Each outlet event is identified via that criteria. Unlike outlets, however, sensors and cameras transmit information (e.g., camera snapshots or motion notifications) to the controller in short bursts at the time of an event; a slightly different approach is used for these devices: each packet's 4-tuple is sorted by source address and the outgoing FSize are aggregated in 1-minute segments. Camera events are identified by a combined FSize greater than 100,000 bytes, while motion events occur between 10,000 and 100,000 bytes. Table III summarizes the criterion used for device classification and event identification.

V. CITIOT TOOL

To illustrate the operation of CITIoT, we provide the following scenario: a user is in a smart home connected to a Wi-Fi AP and interacts with various Wi-Fi and BLE IoT devices. In the morning, the user turns on lights, activates sensors while walking throughout the house, and eventually turns off the lights before leaving the house for work (the door is locked before leaving). While at work, the user checks on the temperature in the house or other devices (e.g., security cameras) remotely. After work, the user returns home, unlocks the front door, turns on lights, and activates sensors throughout the house. Before going to bed, the lights are turned off. During the day, an observer is outside the house sniffing wireless data packets attempting to infer information about the user, devices, and events within the house. The observation process is completely passive and, therefore, undetectable by the user. The eavesdropper has no access to network credentials and is not part of the smart home network.

CITIoT operates in five-steps: (i) reconnaissance and scanning, (ii) passive sniffing, (iii) device classification, (iv) event identification, and (v) user tracking. The remainder of this section describes the operation of CITIoT in regards to the scenario and smart home architecture described above.

A. Reconnaissance and Scanning

The first step is to identify the target walking out of the home using a directional antenna, an Alfa Card, and Airodump-ng. As the target is still connected to the smart home AP, the adversary is able to obtain the target's MAC address, associated router address, and service set identifier (SSID) of the smart home. Next, the adversary scans for devices connected to the smart home by using the sniffing tool while filtering on the target router MAC address. The list of device MAC addresses obtained from this scan are inputed into an Organizationally Unique Identifier (OUI) lookup tool to identify device manufacturers. Using this information, it can be inferred which are IoT devices (e.g., Belkin devices) and which are controllers (e.g., Raspberry Pi or Apple devices).

TABLE III
CLASSIFIER CRITERION.

Device Type	Device Classification	Event Identification (Relies on Device Classification)
Outlet	$FSize_{in} > 650\ bytes$	$1,000 > FSize_{in} > 600\ bytes$
Sensor	$device\ != outlet\ and\ all\ FSize_{in} < 275\ bytes$	$100,000 > \sum_{t}^{t+1} FSize_{out} > 10,000\ bytes$
Camera	$device\ != outlet\ and\ 275 < FSize_{in} < 300\ bytes$	$\sum_{t}^{t+1} FSize_{out} > 100,000\ bytes$

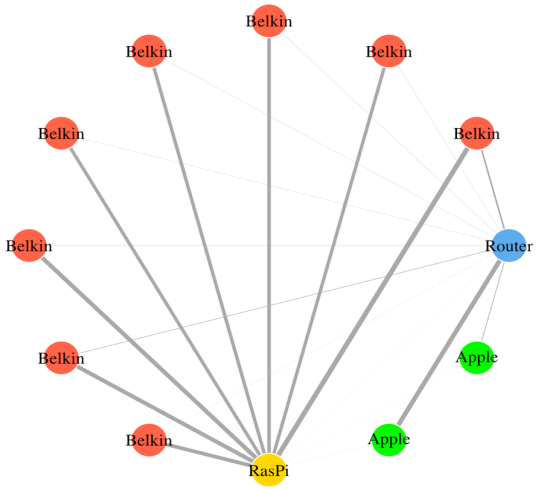

Fig. 2. Network mapping of smart home architecture; thicker lines mean stronger correlation between devices.

Similarly, a Bluetooth wireless adapter (Plugable USB 2.0) and sniffing tool (BlueZ [15]) are used to scan for BLE devices. The adversary then scans Wi-Fi traffic to develop a Wi-Fi network map using the network mapping tool in CITIoT. The network mapping tool parses the capture file for the source, destination, and FSize of each packet to build correlations between devices; Figure 2 shows the output mapping for the smart home architecture with thicker lines indicating that more data is transmitted between these devices.

B. Passive Sniffing

After reconnaissance and scanning, the eavesdropper collects wireless data from the smart home. The Alfa Card is used to collect Wi-Fi traffic, and the sniffing tool is set to filter on the target router MAC address and associated wireless channel. Three Ubertooth Ones (Firmware: 2017-03R2 [16]) are used to collect BLE traffic and are set to follow connections. All of these sniffers can be used on a Raspberry Pi with the proper tools installed, using a battery pack for power, and concealed outside of the target house. The sniffers are left on for the duration of the scenario. At the end of the day, the adversary transfers the capture files from the Raspberry Pi either remotely or by retrieving the Raspberry Pi. The Wi-Fi capture files are parsed using Python 2.7.10 and Pyshark 0.3.7.8, a Python wrapper for parsing packets using Wireshark dissectors [17]. The time, size, source, and destination for each packet are extracted and the resulting 4-tuples stored in a comma-separated values (CSV) file and sent to the classifier.

C. Device Classification

Device classification uses the classifier to categorize Wi-Fi devices into one of three groups: sensor, electrical outlet, or camera. A Python script iterates the CSV file of 4-tuples and passes each packet to the classifier until one meets the criteria and the device is identified. The output of this step is a CSV file with the MAC address and type for each device. The BLE capture files are parsed using Python and Pyshark to locate each advertisement and scan response packet and create a list of devices and names stored in a CSV file.

D. Event Identification

After device classification, events are identified. For Wi-Fi, the CSV file of 4-tuples is parsed and each packet is passed to the classifier. Using the device types and classifier criteria, events are identified. The resulting output is a list of Wi-Fi events with time, source, and destination for each day. BLE events are identified by parsing the capture for connection events. The resulting time, source, and destination of each event is stored in a CSV file.

E. User Tracking

User tracking is accomplished by noting the times of messages sent by devices within the Wi-Fi network. A device that does not send any messages for more than five minutes is considered to be away from the network. Timing begins at the start of the scenario and ends when the scenario is complete.

VI. RESULTS

CITIoT was evaluated while a user interacted with the smart home environment ten hours a day for five days. The user interacted with devices while home in the morning and evening, and while away in the afternoon. During the interactions, the user recorded the time, device, and type of each event. Each device was activated twice in the morning, twice in the evening, and a couple times throughout the day for a total of more than thirty activations per device over the week. As seen in Figure 3, the accuracy of CITIoT was analyzed by the comparison of device classification with actual device types, event identification with the event logs, and user tracking with actual user locations. Findings are summarized in Tables IV and V and discussed in the rest of this section.

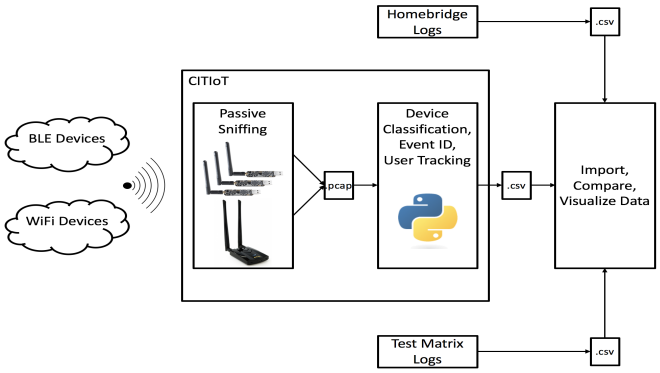

Fig. 3. Data Collection Framework

In step one, reconnaissance and scanning successfully identified the manufacturer of each Wi-Fi device in Table I, correctly mapped the Wi-Fi network, and identified 10 out of 13 BLE devices listed in Table II. The two Apple devices and bike lock do not advertise their device names and were not identified. In step two, sniffing was successful all but one day when the BLE sniffers failed and no data was collected. A total of 6.5 GB and 43.4 million packets were captured throughout the experiment. In step three, device classification was 94% successful with all 8 Wi-Fi devices and 9 out of 10 BLE devices categorized each day. On average, step four was 94% successful in identifying events (true positives). CITIoT failed to identify 6% of events (false negatives) and identified 3 events (false positives) per day that did not actually occur. False positives and negatives were observed in both Wi-Fi and BLE. For Wi-Fi, dropped packets accounted for all failures in event identification. Drops can be expected as the Alfa Card is sniffing passively and there is no method to recover from or identify a dropped packet. Each false positive was caused by the camera or the motion sensor. This was due to the method of aggregation of sent packets within a minute. Enough packets were collected in that minute to break the classifier threshold. These packets appeared to be control responses sent from the devices to the controller. For BLE, dropped or malformed packets accounted for most of the failures in event identification. If a connection request packet is dropped by the Ubertooth One, the sniffer cannot follow the connection and CITIoT will not identify an event. Two event identification failures were due to the Ubertooth only being able to follow one connection at a time; if the Ubertooth is following a different connection and an event occurs, that event will be ignored. BLE false positives occur if a device connects to the controller to pass status information (e.g., battery status). CITIoT assumes every connection is an event and will wrongly identify these connections as events. Step five was 100% successful in determining if the user was in the home or not.

VII. Synthesis

A. Putting It All Together

Using the results from CITIoT we were able to make a few observations about the user and smart home that have security implications: (i) the user was away from the home from 0800-1100 four days out of the week, (ii) the user used a BLE lock to secure their home, and (iii) the user has a Wi-Fi based security camera and motion sensor in their home. The event identifications point to times in the packet capture when the BLE lock was used, and by examining that traffic more closely we were able to observe that the communication between the user and the lock was not encrypted and passwords were sent in the clear. Using this information, we were able to change the user and administrator password or unlock the lock at will. With the knowledge of when the user is away from the home an adversary can predict a good time to attempt to gain access to the smart home. The adversary also knows to be aware of a Wi-Fi camera and sensor.

TABLE IV
Summary of Wi-Fi Results.

Date	True Pos	False Neg	False Pos	# Events
Day 1	31	0	0	31
Day 2	34	1	1	35
Day 3	34	3	1	37
Day 4	35	2	1	37
Day 5	28	4	1	33
Average	32.4	2	1	34.6

TABLE V
Summary of BLE Results.

Date	True Pos	False Neg	False Pos	# Events
Day 1	No Data	No Data	No Data	No Data
Day 2	37	5	5	42
Day 3	40	1	3	41
Day 4	36	0	9	36
Day 5	31	2	4	33
Average	36	2	5.25	38

B. Recommendations to Manufacturers and Users

Many of the vulnerabilities used in this work take advantage of information that is not encrypted at the lower levels of the Wi-Fi and BLE protocols, therefore, to create more secure smart home devices, developers must consider security from the physical layer on up. For Wi-Fi, this includes periodically changing MAC addresses, randomizing FSize, and encrypting lower-layer data packets. M. Gruteser and D. Grunwald provide a framework to change MAC addresses frequently while still maintaining wireless connectivity [18]. The technique of "chaffing and winnowing," as introduced by R. Rivest, can be adapted in smart home technologies to send secure packets intertwined with fake packets of random size to make events impossible to identify [19]. In BLE, devices need to make their advertisements private. K. Fawaz et al., designed an authentication system, BLE-Guardian, that restricts who can discover, scan, and connect to BLE devices that would greatly increase the security of smart home devices [20]. Also, common operational security methods will help prevent against smart home device attacks. For example, users should be aware that routine schedules leave them vulnerable to pattern-of-life modeling– a threat which is increased by smart

devices. Maintaining unpredictable schedules will help prevent attacks. Similarly, turning on or off lights while away from home can trick an observer into thinking someone is home. It is also important to have situational awareness of potential eavesdroppers or suspicious devices around when accessing smart locks or other devices.

C. Vulnerability Drivers

While the recommendations in the previous section can improve the security of smart home environments, none of these ideas are new. Why, then, have these fixes not been implemented to secure against privacy leakage? In response to rapid growth of the IoT market, efforts to limit power, develop devices quickly, and other design constraints drive developers toward poor security implementation, leaving devices vulnerable. Also, while the areas of network and computer security have seen more adversarial pressure, the smart home is relatively new. Until recently, outlets, locks, and light-bulbs were not connected to networks. This is the same evolution of vehicles as they become connected to the Internet and, therefore, open to attack. The privacy implications demonstrated in this work, however, require that developers of IoT technologies consider security in design and engage with the computer security community to create more secure smart homes.

D. Limitations and Future Work

Limitations exist that can enable consumers to mitigate some of the attacks presented in this work. For example, the BLE lock attack relies on poor SM implementation than can be mitigated with software updates. Similarly, if a consumer does thorough research into device security and makes purchases from reputable manufacturers, vulnerabilities are less likely to exist. Also, only a limited selection of Wi-Fi devices were used in training the classifier in CITIoT and only one smart home architecture was tested. If a smart home designer integrated different controllers and devices into the smart home it would further degrade CITIoT's capabilities as it is not trained for those environments or devices. For future work, an analysis of more devices and architectures should be carried out to examine if similar fingerprinting methods can be used and vulnerabilities exist in other devices. Also, other tools can be added to CITIoT, such as location tracking, to further stress the security implications of smart home devices. Additionally, implementation of the above security recommendations on open-source smart home devices can significantly further research in the security of smart home devices.

VIII. Conclusion

As the modern home gets smarter it also becomes more vulnerable to attacks that were reserved to computers and networks. IoT devices constantly communicate data that enable an eavesdropper to infer information about people and devices within a smart home. Users must be aware of what their devices are advertising and how this information can be used against them. This work informs the consumer by presenting a real-world smart home architecture and how an eavesdropper can use CITIoT to learn the types of IoT devices within the home, how they are networked, when events occur in the house, and when someone leaves. We demonstrated how this information can point an attacker to a vulnerable device such as a BLE lock, inform when the users will not be home, and identify what kind of security devices are in the home. After presenting the security implications of smart homes, we recommended ways manufactures and users can defend against these vulnerabilities and how future work can expand on these findings to create a more secure smart home.

References

[1] Consumer Technology Association. (2017). "Record Year Ahead: Consumer Enthusiasm for Connectivity to Propel Tech Industry to Record-Setting Revenues" [Online]. Available: www.cta.tech/News/Press-Releases/2016/January/Record-Year-Ahead-Consumer-Enthusiasm-for-Connect.aspx, [Last accessed Aug 27, 2017].

[2] *Wireless LAN Medium Access Control, MAC, and Physical Layer, PHY, Specification*, IEEE Standard 802.11, 2016. Available: standards.ieee.org/about/get/802/802.11.html, [Last accessed Aug 27, 2017]

[3] M. Zhou et al., "SCaNME: Location tracking system in large-scale campus Wi-Fi environment using unlabeled mobility map," in *Expert Systems with Applications*, vol. 41, no. 7, pp. 3429-3443, Oct. 2014.

[4] B. Bonne et al., "WiFiPi: Involuntary tracking of visitors at mass events," in *IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, Madrid, Spain, 2013, pp. 1-6.

[5] J.S. Atkinson et al., "Your WiFi is leaking: What do your mobile apps gossip about you?" *Future Generation Computer Systems*, May 2016.

[6] *Specification of the Bluetooth System*, Core Version 4.0, June 30, 2010. Available: www.bluetooth.com/specifications/bluetooth-core-specification, [Last accessed Aug 27, 2017].

[7] *Specification of the Bluetooth System*, Core Version 4.2, December 2, 2014. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/legacy-specifications, [Last accessed Aug 27, 2017],

[8] R. Heydon, *Bluetooth Low Energy: The Developer's Handbook*. Crawfordsville, IN: Pearson Education, Inc., 2013.

[9] A. Das et al., "Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers," in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, St. Augustine, FL, 2016, pp. 99-104.

[10] A. Rose et al., "Bluefinder: A range-finding tool for Bluetooth classic and low energy," in *Proceedings of the 12th International Conference on Cyber Warfare and Security*, Dayton, OH, 2017, pp. 303-311.

[11] A. Rose and B. Ramsey. (2016). "Picking Bluetooth Low Energy locks from a quarter mile away" [Online]. Available: media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/, [Last accessed on Aug 30, 2017].

[12] Raspberry Pi Foundation. (2016). "Raspberry Pi 3 Model B Specifications" [Online]. Available: raspberrypi.org/products/raspberry-pi-3-model-b/, [Last accessed on Aug 30, 2017].

[13] nfarina. (2015). "Homebridge" [Online]. Available: github.com/nfarina/homebridge/, [Last accessed on Aug 30, 2017].

[14] devbobo. (2017). "homebridge-platform-wemo" [Online]. Available: npmjs.com/package/homebridge-platform-wemo/, [Last accessed on Aug 30, 2017].

[15] BlueZ. (2016). *BlueZ, Official Linux Bluetooth Protocol stack* [Online]. Available: bluez.org, [Last accessed on Aug 27, 2017].

[16] Ubertooth. (2017). *greatscottgadgets* [Online]. Available: github.com/greatscottgadgets/ubertooth/, [Last accessed on Sep 5, 2017].

[17] Pyshark. (2017). *KimiNewt* [Online]. Available: github.com/KimiNewt/pyshark, [Last accessed on Aug 27, 2017].

[18] M. Gruteser and D. Grunwald, "Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative Analysis," in *Mobile Networks and Applications*, vol. 10, no. 3, pp. 315-325, Jun. 2005.

[19] R. Rivest. "Chaffing and winnowing: Confidentiality without encryption" in *CryptoBytes (RSA laboratories)*, vol. 4, no. 1, pp. 12-17, 1998.

[20] K. Fawaz et al., "Protecting privacy of BLE device users" in *USENIX Security Symposium*, Austin, TX, 2016, pp. 1205-1221.