0.1 Introduction

In recent years, smart home devices have become one of the most popular categories for 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 [2]. WiFi 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 WiFi 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 an observer can use information leaked to classify devices, identify events such as when a door is opened or when a light is turned on, and track occupants of the home. 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 WiFi and BLE commercial/-off/-the/-shelf (COTS) devices with Apple's home automation application, HomeKit. Examples of interactions with the smart home environment include turning on lights, opening doors, activating motion sensors, and unlocking locks. These interactions occur both in a controlled manner and freely and while the user is home or away.

Vulnerability analysis. We explain how an eavesdropper can use device vulnerabilities to extract information while outside the smart home environment via raw signals sniffed over the air. These, combined with characteristic data exchanges and packet sizes, can be used to fingerprint components of the smart home environment.

CITIoT. The tool presented in this work provides three capabilities against smart home environments: device classification, event identification, and user tracking. For

WiFi, a fingerprinting technique is applied to classify smart home devices into one of three groups: sensor, outlet, or camera. For BLE devices we similarly classify devices, but provide more descriptive information. The fingerprint technique is also applied to identify events within the smart home such as turning on a light or movement in the house. Lastly, the observed smart home traffic is used to predict when users will be in the smart home.

Synthesis. We present how an observer can use the information gathered from smart home devices to create pattern-of-life models, crack a Bluetooth lock, and gain access to the home when a user is predicted to be away. We observe that these vulnerabilities are not unique to the devices under study, but indicative of IoT design. We provide limitations to our approach and recommendations to prevent these vulnerabilities and create a more secure smart home environment.

The remainder of this paper is organized as follows: Section 0.2 discusses smart home technologies, a brief technical overview of the WiFi and BLE protocols, and a summary of tools leveraged in the design of CITIoT. Section 0.3 summarizes related research in the areas of WiFi and BLE security and IoT device privacy. Section 0.4 presents the smart home architecture used in creating and analyzing CITIoT. Section 0.5 describes the process used to create CITIoT starting with methods of reconnaissance and device traffic sniffing, to detailing the construction of the fingerprinting method used to classify, identify, and track devices. Section 0.6 reports CITIoT's accuracy on our smart home architecture. Finally, applications, recommendations, vulnerability drivers, and limitations are discussed in Section 0.7 before the paper is concluded in Section 0.8.

0.2 Background

Smart Home Technologies.

A list of smart home terms relevant to the work in this paper is provided:

- **Devices**: BLE or WiFi devices such as switches, smart outlets, cameras, or sensors. Can be connected to and controlled by controllers.
- **Controllers**: A master device such as an iPhone or Android that connects to device within the smart home to get status updates or change states.
- **Hub**: A system that sits on the home network, connects to different devices via the manufacturer application programming interface (API), and exposes control of the device via a centralized application on the `controller`. Hubs often provide access to the devices while a user is away from the smart home. Examples of hubs includes Apple's Homekit and the open-source server Homebridge.
- **Apple's HomeKit**: A hub that provides a controller with voice control and automation capabilities for devices.
- **Applications**: Many smart home devices require proprietary applications to interact with the device's full range of capabilities. A controller must use these applications to control the device.

Technical Overview.

This section provides a brief technical summary of the WiFi and BLE protocols necessary to understand the rest of this work.

WiFi.

The WiFi standard defines the Physical layer and Link layer for wireless communication in the 2.4 GHz radio band [1]. Because WiFi does not use a point-to-point medium, anyone with a properly-tuned receiver can observe traffic sent over the air. Consequently, for the sake of confidentiality, the protocol defines security procedures to encrypt data sent within a wireless network. Even with encryption, however, WiFi still transmits information in the clear within the MAC Protocol Data Unit (MPDU) data frame. Values of interest sent in the clear include frame type, source Media Access Control (MAC) address, destination MAC address, and router address. Also, the time, size, and Received Signal Strength Indicator (RSSI) of the packets can be ascertained from the receiver wireless network card. In this work we make no effort to break encryption, but instead demonstrate what can be inferred from packets sent within an encrypted wireless access point (AP).

Bluetooth Low Energy.

The Bluetooth Special Interest Group (SIG) introduced BLE (Bluetooth Smart) in Bluetooth Core Specification v4.0 [4]. BLE is designed to minimize power, cost, and data rate and these goals are accomplished by limiting overhead at every level of the architecture and using simple communication protocols. To this same end, BLE devices predominantly transmit state data, such as whether a light is on/off, in short, infrequent bursts. These characteristics make BLE ideal for IoT applications where battery life is a top priority. Similarly to WiFi, the elements of the BLE architecture relevant to this work are in the Physical and Link layers.

The physical and link layers control frequency hopping, finding devices, establishing connections, packet structure, and transmitting/receiving data. As shown in Figure 1, BLE operates in the 2.4GHz band which is divided into forty channels separated by 2-MHz. These frequencies are distributed into three advertising channels (37, 38, and 39) and thirty-seven data channels.

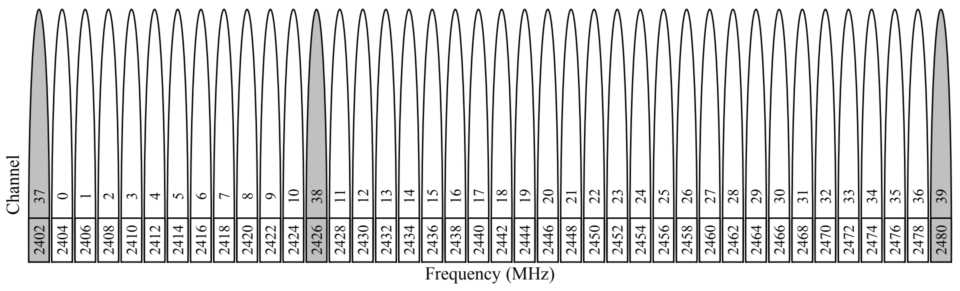


Figure 1. Bluetooth Low Energy channel mapping; darker channels represent advertisement channels

Connections use a master-slave model which begins with a slave announcing its presence by broadcasting advertising packets on the three advertisement channels. Each advertising packet includes device information such as connectability, scannability, services provided, and name of the device. Advertising packets also include a “TxAdd” bit that indicates if the advertiser is using a public or random address. A master actively or passively scans the advertisement channels detecting connectible devices. Active scanning is a key concept for fingerprinting in this work and the process is depicted in Figure 2. When actively scanning, a master observes an advertising packet and, if the device is scannable, sends a scan request to the device. The advertiser sends a scan response back with more information, typically expanding on the device name and possibly including broadcast data such as battery level. A master can only connect to a device that advertises its presence and is connectible.

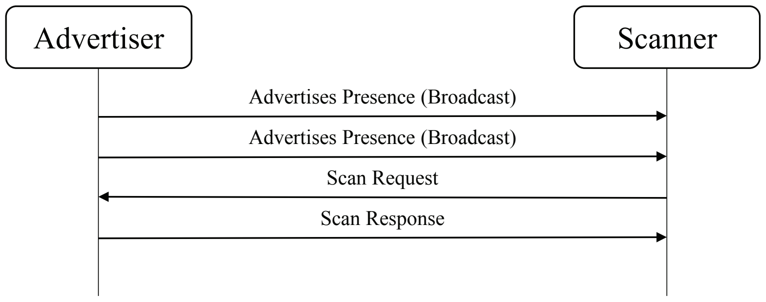

Advertiser
Scanner
Advertises Presence (Broadcast)
Advertises Presence (Broadcast)
Scan Request
Scan Response

Figure 2. Active Scanning Process

When in a connection, a master and slave communicate on one channel per Connection Events (CE). After each CE, both the master and slave hop to a new frequency per hopping parameters established by the master at the beginning of a connection or in a parameter update.

Tools.

This work employs many WiFi and BLE tools for sniffing, parsing traffic, visualizing data, and attack. As an aid ot the reader, a summary of tools is provided in Table. airodump-ng -scanning alfa card ubertooth one python pyshark

0.3 Related Work

Although Wifi and BLE smart home devices are becoming commonplace, the private data leaked and security vulnerabilities of these devices is largely unexplored. In a few recent studies focused on BLE wearable fitness trackers one group of researchers observed leaked privacy data to determine user activity level and gait [7], while another group tracked a user wearing a Fitbit Surge up to 1,000 meters away with greater than eighty percent accuracy [14]. Researchers have used Wifi MAC addresses sent in the clear and RSSI values to create location tracking systems on campuses, crowd tracking at mass events, and in Customer Relationship Management (CRM) allowing commercial businesses to track customer interactions and data [12][6][3]. The lack of Link Layer authentication or encryption for some smart home BLE devices enabled researchers to crack twelve BLE locks from up to a quarter mile away [15]. Closest to this work, researchers from the United Kingdom were able to use raw Wifi signals to create fingerprinting techniques able to identify applications used on a mobile phone [3]. From the the extent of our research, however, this is the first work providing a broad review of privacy leakage from COTS smart home devices in the wild and using this data to classify devices, identify events, and track users in a smart home.

0.4 Smart Home Architecture

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 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 3 and is described in the rest of this section.

Raspberry Pi.

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 [9]. 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 plugin is utilized to expose WeMo devices to the Apple Homekit and

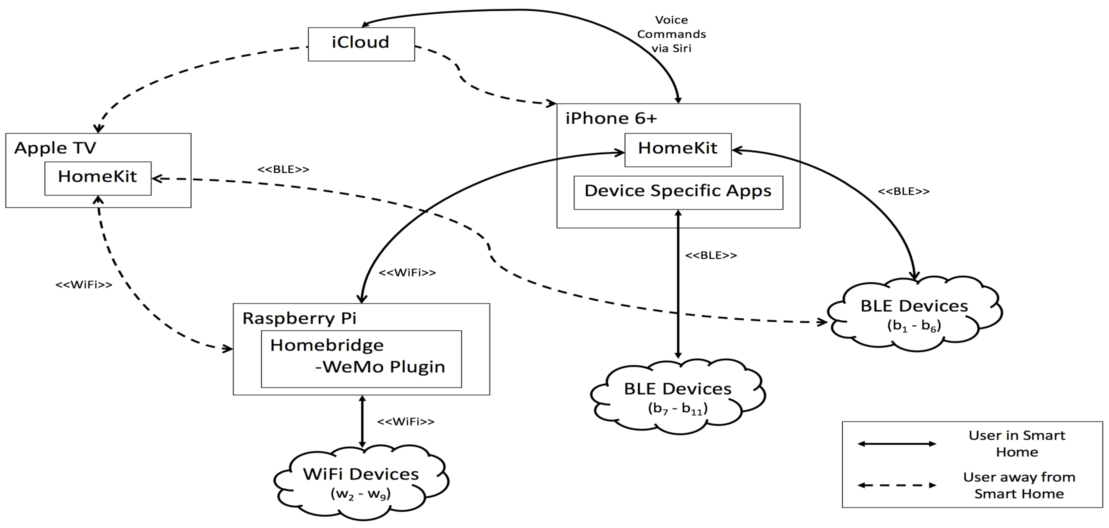


Figure 3. Smart Home Architecture

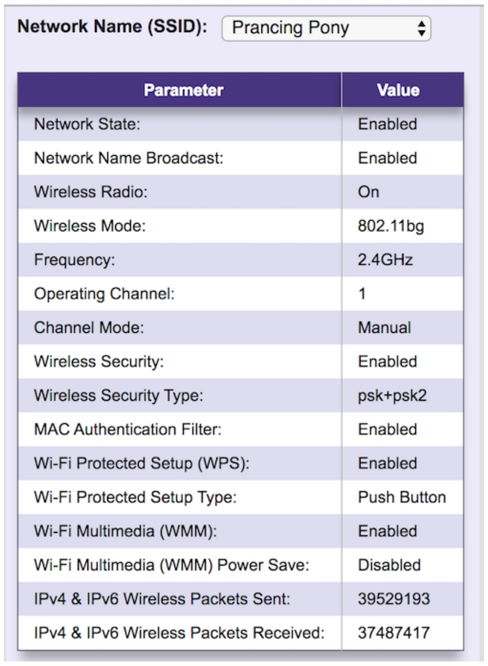
is loaded into Homebridge [8]. The Apple devices communicate with the Raspberry Pi to interact with the WiFi Devices w_2-w_9.

Apple Devices.

The iPhone 6+ and Apple TV act as controllers in the smart home architecture and connect to devices via WiFi and BLE. When the user is home, the iPhone connects to WiFi devices via the Homebridge and connects directly to the 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. 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 WiFi or BLE. This will only work with Homekit supported devices, therefore, BLE devices b_7-b_{12} (see 2) cannot be accessed while the user is away from the home. Devices will be accessed while the user is both home and away to observe differences in communication.

WiFi Devices.

To facilitate WiFi communication in the smart home architecture, a 2.4GHz WiFi AP, “Prancing Pony”, was setup with WPA2 security on channel 1 (see Figure 4 for a complete list of settings). A list of devices connected to the AP can be found in Table 1. 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.


Network Name (SSID): Prancing Pony

Parameter	Value
Network State:	Enabled
Network Name Broadcast:	Enabled
Wireless Radio:	On
Wireless Mode:	802.11bg
Frequency:	2.4GHz
Operating Channel:	1
Channel Mode:	Manual
Wireless Security:	Enabled
Wireless Security Type:	psk+psk2
MAC Authentication Filter:	Enabled
Wi-Fi Protected Setup (WPS):	Enabled
Wi-Fi Protected Setup Type:	Push Button
Wi-Fi Multimedia (WMM):	Enabled
Wi-Fi Multimedia (WMM) Power Save:	Disabled
IPv4 & IPv6 Wireless Packets Sent:	39529193
IPv4 & IPv6 Wireless Packets Received:	37487417

Figure 4. Prancing Pony Access Point Settings

Bluetooth Low Energy Devices.

For Bluetooth communication to occur in the smart home architecture a Bluetooth 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

Table 1. **Wi-Fi Devices.**

ID	Manuf	Device Type	Device Name	MAC	IP Address
w_1	Calix	Wireless Router	Prancing Pony	EC:4F:82:73:D1:1A	206.55.216.151
w_2	Belkin	Camera	NetCam	EC:1A:59:E4:FD:41	192.168.1.44
w_3	Belkin	Outlet	Switch1	B4:75:0E:0D:33:D5	192.168.1.40
w_4	Belkin	Outlet	Switch2	B4:75:0E:0D:94:65	192.168.1.41
w_5	Belkin	Outlet	Switch3	94:10:3E:2B:7A:55	192.168.1.42
w_6	Belkin	Outlet	Switch4	14:91:82:C8:6A:09	192.168.1.7
w_7	Belkin	Motion Sensor	Motion	EC:1A:59:F1:FB:21	192.168.1.43
w_8	Belkin	Outlet	Insight	14:91:82:24:DD:35	192.168.1.47
w_9	WeMo	Outlet	Mini	60:38:E0:EE:7C:E5	192.168.1.51
w_{10}	Raspberry Pi 3B	Computer	Pi	B8:27:EB:09:1A:81	12.168.1.50
w_{11}	Apple	iPhone 6+	Steves-phone	A0:18:28:33:34:F8	192.168.1.4
w_{12}	Apple	TV 2	Apple-TV	08:66:98:ED:1E:19	192.168.1.54

in the BLE can be found in Table 2. The MAC addresses for each device are not included because they are randomized per the BLE protocol [5].

0.5 Methodology

The methodology explained in this section is illustrated in Figure 5 and expanded on in the following scenario: a user is in her smart home interacting with devices throughout the day. In the morning, she turns on lights, activates sensors while walking throughout the house, and eventually turns off lights before leaving the house for work (she locks her door on her way out). During the day, while at work, she may check on the temperature in the house or other devices remotely. After work, the user returns home, unlocks her front door, turns on lights, and activates sensors throughout the house. Before going to bed she turns off the lights. During the normal usage of the smart home environment, an adversary is outside sniffing 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 adversary has no access to the network's credentials and is not part of the

Table 2. 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_5	Elgato	Switch	Eve Light
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

smart home network. The rest of this section will discuss the process used in CITIoT that allows the observer to infer things about the smart home and is summarized in five primary steps: (i) reconnaissance and scanning, (ii) passive sniffing, (iii) device classification, (iv) event identification, and (v) user tracking.

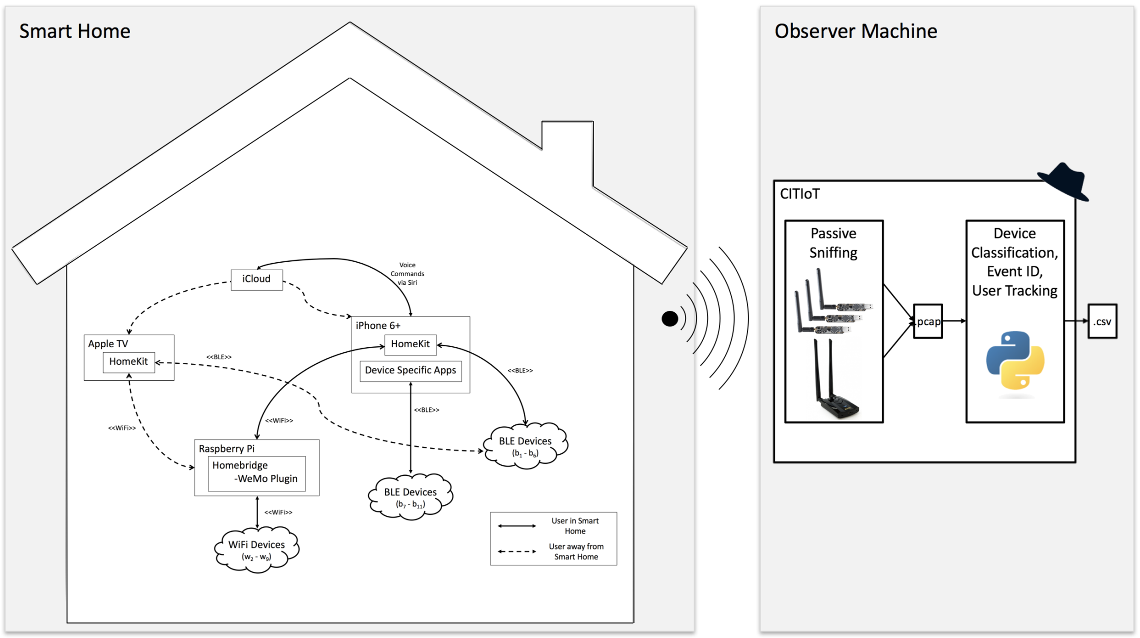
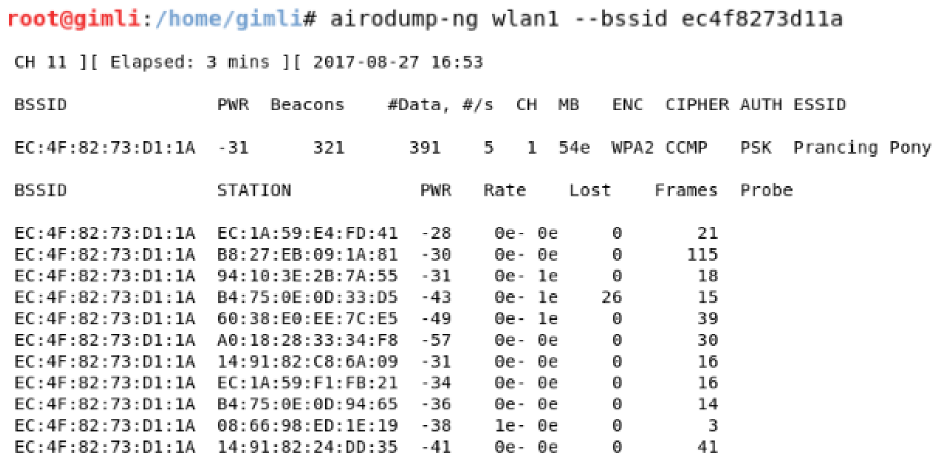


Figure 5. Overall scenario setup.

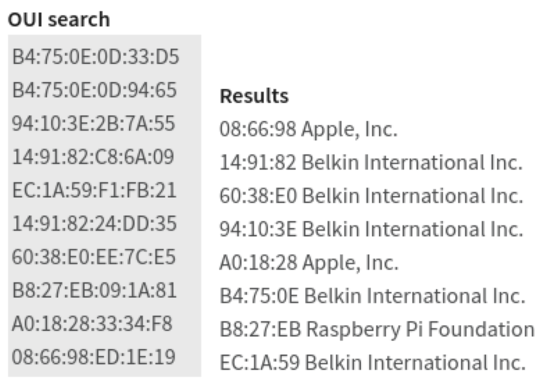
Reconnaissance and Scanning.

Outside of the primary capabilities of CITIoT, reconnaissance and scanning is a key component to the overall operation of the tool. Reconnaissance includes selecting the target and observing that target's phone's WiFi MAC address. This is a trivial step as the MAC address is sent in the clear in every packet from the device and can be sniffed using an Alfa Card with a directional antenna. The next step is to observe which access point in the smart home the phone connects to. Again, this is trivial, as devices send the service set identifier (SSID) of the AP they are connected to in the clear when associated to the AP. In our scenario, we observe that the user's phone connects to the AP with the SSID "EC:4F:82:73:D1:1A". Using the open source tool "airodump-ng" we observe that the SSID is called "Prancing Pony", it is operating on channel 1, and it has 11 devices connected to it (see Figure 6 for the command and corresponding output). Next, we perform a simple MAC address lookup using Wireshark's OUI Lookup Tool to determine the name of the company that manufactured the network card for each device connected to "Prancing Pony" (the results can be seen in Figure 7) [16].



```
root@gimli:/home/gimli# airodump-ng wlan1 --bssid ec4f8273d11a

 CH 11 ][ Elapsed: 3 mins ][ 2017-08-27 16:53

 BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 EC:4F:82:73:D1:1A  -31      321      391    5   1  54e  WPA2 CCMP   PSK  Prancing Pony

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 EC:4F:82:73:D1:1A  EC:1A:59:E4:FD:41  -28    0e- 0e     0       21
 EC:4F:82:73:D1:1A  B8:27:EB:09:1A:81  -30    0e- 0e     0      115
 EC:4F:82:73:D1:1A  94:10:3E:2B:7A:55  -31    0e- 1e     0       18
 EC:4F:82:73:D1:1A  B4:75:0E:0D:33:D5  -43    0e- 1e    26       15
 EC:4F:82:73:D1:1A  60:38:E0:EE:7C:E5  -49    0e- 1e     0       39
 EC:4F:82:73:D1:1A  A0:18:28:33:34:F8  -57    0e- 0e     0       30
 EC:4F:82:73:D1:1A  14:91:82:C8:6A:09  -31    0e- 0e     0       16
 EC:4F:82:73:D1:1A  EC:1A:59:F1:FB:21  -34    0e- 0e     0       16
 EC:4F:82:73:D1:1A  B4:75:0E:0D:94:65  -36    0e- 0e     0       14
 EC:4F:82:73:D1:1A  08:66:98:ED:1E:19  -38    1e- 0e     0        3
 EC:4F:82:73:D1:1A  14:91:82:24:DD:35  -41    0e- 0e     0       41
```

Figure 6. Scanning command along with output.


OUI search

B4:75:0E:0D:33:D5
B4:75:0E:0D:94:65
94:10:3E:2B:7A:55
14:91:82:C8:6A:09
EC:1A:59:F1:FB:21
14:91:82:24:DD:35
60:38:E0:EE:7C:E5
B8:27:EB:09:1A:81
A0:18:28:33:34:F8
08:66:98:ED:1E:19

Results

08:66:98	Apple, Inc.
14:91:82	Belkin International Inc.
60:38:E0	Belkin International Inc.
94:10:3E	Belkin International Inc.
A0:18:28	Apple, Inc.
B4:75:0E	Belkin International Inc.
B8:27:EB	Raspberry Pi Foundation
EC:1A:59	Belkin International Inc.

Figure 7. Media Access Control Organizationally Unique Identifier search and results.

The MAC addresses and manufacturers allow the observer to make her first classification: which devices may be controllers (Apple and Raspberry Pi) and smart home devices (Belkin). This information is used to filter traffic in CITIoT. The next four steps occur within CITIoT, are depicted in Figure 8, and described below.

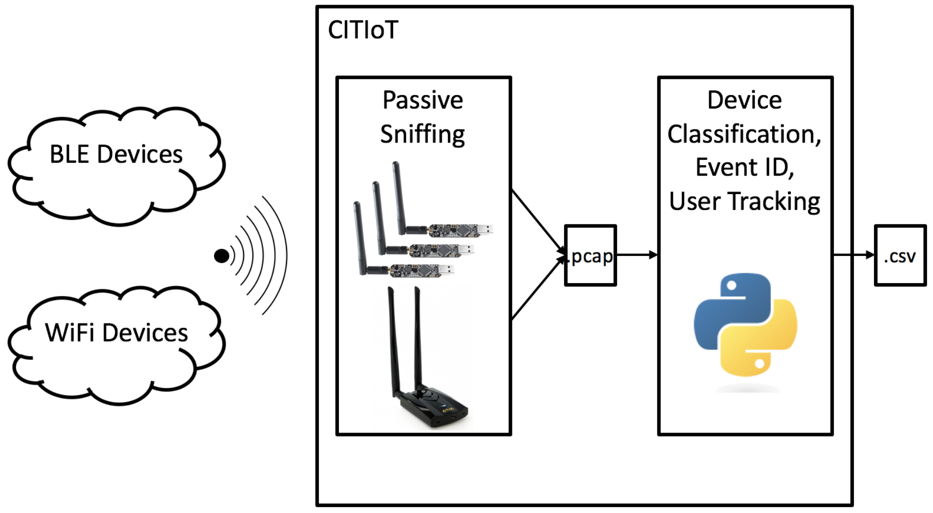
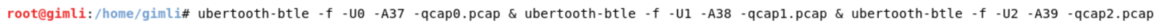


Figure 8. CITIoT Architecture

Passive Sniffing.

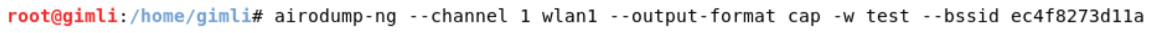
Sniffing is accomplished with three Ubertooth One devices (one for each advertisement channel) for BLE packets and an Alfa Card for WiFi packets. The command used to start BLE sniffing is shown in Figure 9. The ’-f’ flag sets follow mode, the ’-U’ flag sets which Ubertooth device to use, the ’-A’ flag sets which advertising channel to listen to, and the ’-q’ flag outputs the captured files to a PCAP file. At the end of collection, each PCAP is merged to create one BLE capture file.



```
root@gimli:/home/gimli# ubertooth-btle -f -U0 -A37 -qcap0.pcap & ubertooth-btle -f -U1 -A38 -qcap1.pcap & ubertooth-btle -f -U2 -A39 -qcap2.pcap
```

Figure 9. Command used to sniff Bluetooth Low Energy packets.

The command used to start WiFi sniffing is shown in Figure 10. The ’–channel’ flag sets which WiFi channel to capture packets on, the ’–output-format’ flag sets the format type for the capture, the ’-w’ flag sets the name of the output capture, and the ’–bssid’ filters packets by basic service set identifier (BSSID).



```
root@gimli:/home/gimli# airodump-ng --channel 1 wlan1 --output-format cap -w test --bssid ec4f8273d11a
```

Figure 10. Command used to sniff WiFi packets.

After traffic collection is complete a preprocessing script is ran on the WiFi captures to pull out the data needed for classification and identification. As CITIoT makes no attempt to break WiFi encryption and the Data-Link Layer is encrypted, the only information we have access to is from the physical layer. To dissect the PCAP captures into the data of interest, the script utilizes pyshark, a python wrapper based off of Wireshark dissector, tshark [11]. First, the dissector pulls the time, source, destination, and frame size (FSize) for each packet. Then, using the list of smart home devices found above, the script distinguishes between sent and received data for each device. This creates two comma-separated values (CSV) files for each device: one which the device is the source of each packet and the other which the device is the destination of each packet. These CSV files are then passed to the device classification unit.

Device Classification.

WiFi device classification happens using FSize for packets in which the device is the destination. Each packet destined for a specific device is parsed and depending on the trend of packets received (timing, size, and source), the device is classified as either a sensor, outlet, or camera. Figures 11, 12, and 13 illustrate classification using a FSize distribution created by plotting a histogram for the size of each frame (in bytes) for a given device. Specifically, a packet frame of size greater than 650 bytes is indicative of an outlet, a device that receives no packet frames greater than 275 bytes points to a sensor device, and a device that receives a packet with FSize between 275 and 300 bytes is a camera device. These criterion are summarized in Table **??**.

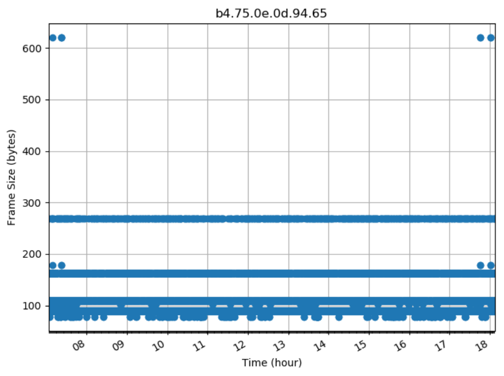


Figure 11. Outlet device.

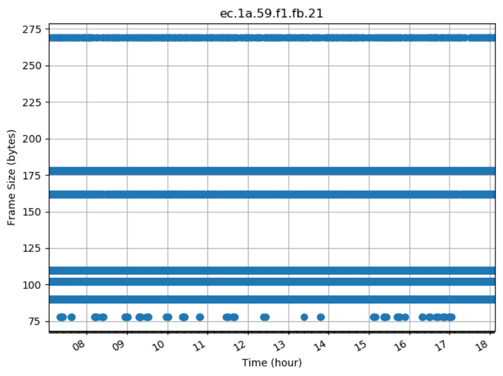


Figure 12. Sensor device.

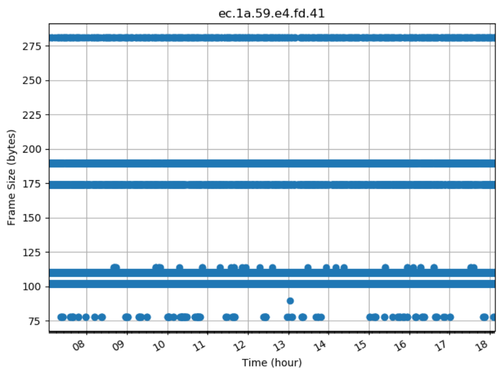


Figure 13. Camera device.

Classification of BLE devices occur in two stages: first, each device name and address is parsed from advertisement and scan response packets and stored in a local database; and, second, the tool maps all addresses from connection requests with the name from the database. This mapping provides names of devices the user connected to and the times of these connections.

Event Identification.

Event Identification occurs after Device Classification and relies on correct classification of devices. Depending on the classification of device, different event thresholds are established. For outlets, the incoming packets' FSize are observed (a user turning on or off a device). If the frame is greater than 600 bytes but less than 1000 bytes, then this is an outlet event. For sensors, the outgoing packets' cumulative FSize over one second are observed (a motion sensor informing a hub that an event has occurred). From experience, a sensor will send a burst of packets for each motion event with the overall size being greater than 10,000 bytes, but less than 100,000 bytes. A similar method is used for camera events, but these bursts are typically greater than 100,000 bytes. These criterion are summarized in Table **??**.

User Tracking.

User tracking is based off of device WiFi MAC address. The timing of packets sent from a device to any destination are tracked throughout the day. If at any time a device does not send packets for more than five minutes, that device is categorized as being no longer within the home. As soon as the device sends a packet, that device is considered back in the home.

0.6 Results

0.7 Synthesis and Discussion

Applications.

Recommendations.

Periodically changing MAC addresses Encrypting lower-layer data packets Chaffing and Winnowing

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 privacy? For example, the BLE specification defines security procedures to encrypt the payload, generate private addresses, and provide authentication [5]. However, implementation of security is left up to the designer and each additional security measure contributes to increased energy consumption [10]. Limiting power, developing devices quickly, and other design constraints drive developers towards poor security implementation, leaving devices with essentially no Link Layer authentication or encryption. Also, while network and computer security has seen more adversarial pressure, the smart home is relatively new. Until recently outlets, switches, and light-bulbs have not been connected to networks. This is the same evolution vehicles are facing as they become more connected to the Internet.

Limitations.

0.8 Conclusion

Bibliography

1. *Wireless LAN Medium Access Control, MAC, and Physical Layer, PHY, Specification.*

2. Consumer Technology Association. "Record Year Ahead: Consumer Enthusiasm for Connectivity to Propel Tech Industry to Record-Setting Revenues," 2017. [Last accessed August 27, 2017], *https://www.cta.tech/News/Press-Releases/2016/January/Record-Year-Ahead-Consumer-Enthusiasm-for-Connect.aspx.*

3. J.S Atkinson et al. Your wifi is leaking: What do your mobile apps gossip about you? *Future Generation Computer Systems*, 2016.

4. Bluetooth SIG. *Specification of the Bluetooth System Core Version 4.0*, 2010. *https://www.bluetooth.com/specifications/adopted-specifications/legacy-specifications.*

5. Bluetooth SIG. *Specification of the Bluetooth System Core Version 4.2*, 2010. [Last accessed on August 30, 2017], *www.bluetooth.com/specifications/bluetooth-core-specification.*

6. B. Bonne, A. Barzan, P. Quax, and W. Lamotte. Wifipi: Involuntary tracking of visitors at mass events. *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2013.

7. A. Das, P. Pathak, C. Chuah, and P. Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers.

8. devbobo. "homebridge-platform-wemo," 2017. [Last accessed on August 30, 2017], Available at *npmjs.com/package/homebridge-platform-wemo/.*

9. Raspberry Pi Foundation. "Raspberry Pi 3 Model B Speicfications," 2016. [Last accessed on August 30, 2017], Available at *raspberrypi.org/products/raspberry-pi-3-model-b/.*

10. Robin Heydon. *Bluetooth Low Energy: The Developer's Handbook.* Pearson Education, Inc., Crawfordsville, Indiana, 2013.

11. KimiNewt. pyshark, 2017. Available at *https://github.com/KimiNewt/pyshark.*

12. K. Xu X. Yu X. Hong M. Zhou, Z. Tian and H. Wu. Scanme: Location tracking system in large-scale campus wi-fi environment using unlabeled mobility map. *Expert Systems with Applications*, 41(7):3429–3443, 2014.

13. nfarina. "Homebridge," 2015. [Last accessed on August 30, 2017], Available at *github.com/nfarina/homebridge/.*

14. A. Rose, J. Gutierrez del Arroyo, and B. Ramsey. Bluefinder: A range-finding tool for bluetooth classic and low energy. *Proceedings of the Twelfth International Conference on Cyber Warfare and Security*, 2017.

15. A. Rose and B. Ramsey. Picking bluetooth low energy locks from a quarter mile away. 2016. presented at DEF CON 24 *(media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/)*.

16. Wireshark. Oui lookup tool. Available at *https://www.wireshark.org/tools/oui-lookup.html* .