

A Privacy Vulnerability in Smart Home IoT Devices

by

Michael W. Denko

**A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering
(Computer Engineering)
in the University of Michigan-Dearborn
2017**

Master's Thesis Committee:

**Associate Professor Hafiz Malik, Chair
Professor Paul Richardson
Professor Weidong Xiang**

Table of Contents

List of Tables	iv
List of Figures	v
Abstract	vii
Chapter 1 Introduction	1
Chapter 2 Related Work	3
Chapter 3 Experimentation	5
3.1 Experiment Setup	5
3.2 Smart Light Bulbs	7
3.3 TP-Link Smart LED Light Bulb	7
3.4 Philips Hue Smart Light Bulb	9
3.5 Sengled Element Classic Smart Light Bulb	11
3.6 Smart Light Bulb Discussion	13
3.7 Smart Plug – Belkin WeMo	15
3.8 Smart and WiFi Thermostats	16
3.9 Vine WiFi Thermostat	17
3.10 Sensi WiFi Thermostat	21
3.11 Smart and WiFi Thermostat Discussion	22
3.12 Amazon Echo Dot	24
Chapter 4 Privacy Vulnerability Discussion	27
4.1 Privacy Vulnerability	27
4.2 Lack of Encryption of Transmitted Data over Local WiFi Network	28

4.3 WiFi Vulnerabilities	28
4.4 Vulnerabilities in TLSv1.2 Protocol	30
Chapter 5 Solutions to the Privacy Vulnerability	32
Chapter 6 Conclusion	35
Appendix A: Collected Data from Smart Home IoT Devices	36
A.1 TP-Link LED Smart Light Bulb Recorded Data	36
A.2 Philips Hue Smart Light Bulb Recorded Data	38
A.3 Sengled Element Classic Smart Light Bulb Recorded Data	42
A.4 Belkin WeMo Smart Plug Recorded Data	44
A.5 Vine WiFi Thermostat Recorded Data	48
A.6 Sensi WiFi Thermostat Recorded Data	52
A.7 Amazon Echo Dot Recorded Data	55
Bibliography	56

List of Tables

Table 3-1 Smart Light Bulb Security and Privacy Analysis Results	14
Table 4-1 Security and Privacy Analysis Results	27

List of Figures

Figure 3-1 EAPOL Message Capture in Wireshark	6
Figure 3-2 TP-Link Smart LED Light Bulb Communication Diagram	7
Figure 3-3 Data Transmitted From Smartphone to Turn Smart Light Bulb On	8
Figure 3-4 Philips Hue Communication Diagram	10
Figure 3-5 Sengled Element Communication Diagram	12
Figure 3-6 Sengled Communication with Amazon Server	13
Figure 3-7 Belkin WeMo Communication Diagram	15
Figure 3-8 Belkin WeMo XML SOAP Format	16
Figure 3-9 WiFi Thermostat Communication Diagram	18
Figure 3-10 Vine WiFi Thermostat Schedule Change Data	20
Figure 3-11 Vine WiFi Thermostat Encrypted Data	24
Figure 3-12 Comparison Between Vine WiFi Thermostat Encrypted Data Packets	24
Figure 3-13 Amazon Echo Dot Communication Diagram	25
Figure 3-14 Data Transmitted By Amazon Echo Dot When Not in Use	26
Figure A-1: Turn Smart Light Bulb On (From Smartphone)	36
Figure A-2 Turn Smart Light Bulb On Response (From Light Bulb)	36
Figure A-3 Turn Smart Light Bulb Off (From Smartphone)	37
Figure A-4 Turn Smart Light Bulb Off Response (From Light Bulb)	37
Figure A-5 Turn Smart Light Bulb On Binary Data (From Smartphone)	38
Figure A-6 Turn Smart Light Bulb On ASCII Data (From Smartphone)	38
Figure A-7 Turn Smart Light Bulb On Response Binary Data (From Light Bulb)	39
Figure A-8 Turn Smart Light Bulb On Response ASCII Data (From Light Bulb)	39
Figure A-9 Turn Smart Light Bulb Off Binary Data (From Smartphone)	40
Figure A-10 Turn Smart Light Bulb Off ASCII Data (From Smartphone)	40
Figure A-11 Turn Smart Light Bulb Off Response Binary Data (From Philips Hue Hub)	41
Figure A-12 Turn Smart Light Bulb Off Response ASCII Data (From Light Bulb)	41
Figure A-13 Communication between Smartphone and Sengled Server	42

Figure A-14 Turn Smart Light Bulb On Binary Data (Example 1)	42
Figure A-15 Turn Smart Light Bulb On Binary Data (Example 2)	43
Figure A-16 Turn Smart Light Bulb Off Binary Data (Example 1)	43
Figure A-17 Turn Smart Light Bulb Off Binary Data (Example 2)	44
Figure A-18 Turn On Smart Plug Binary Data (From Smartphone)	45
Figure A-19 Turn On Smart Plug ASCII Data (From Smartphone)	46
Figure A-20 Turn Off Smart Plug Binary Data (From Smartphone)	47
Figure A-21 Turn Off Smart Plug ASCII Data (From Smartphone)	48
Figure A-22 Screenshot of Communication Between Smartphone and Vine Server	48
Figure A-23 Turn WiFi Thermostat On Binary Data (Before Firmware Update)	49
Figure A-24 Turn WiFi Thermostat On ASCII Data (Before Firmware Update)	49
Figure A-25 Turn WiFi Thermostat Off Binary Data (Before Firmware Update)	49
Figure A-26 Turn WiFi Thermostat Off ASCII Data (Before Firmware Update)	49
Figure A-27 Change Weekly Schedule Binary Data (Before Firmware Update)	50
Figure A-28 Change Weekly Schedule ASCII Data (Before Firmware Update)	51
Figure A-29 Turn WiFi Thermostat On ASCII Data (After Firmware Update)	52
Figure A-30 Turn WiFi Thermostat On ASCII Data (After Firmware Update)	52
Figure A-31: Change Weekly Schedule ASCII Data (After Firmware Update)	52
Figure A-32 Communication Between Smartphone and Sensi Server	53
Figure A-33 Turn WiFi Thermostat to Auto Mode (Example 1)	53
Figure A-34 Turn WiFi Thermostat to Auto Mode (Example 2)	53
Figure A-35 Turn WiFi Thermostat to Off Mode (Example 1)	54
Figure A-36 Turn WiFi Thermostat to Off Mode (Example 2)	54
Figure A-37 Amazon Echo Dot Receiving Data Example	55
Figure A-38 Amazon Echo Dot Transmitting Data Example	55

Abstract

Smart home IoT devices are becoming increasingly popular and increasingly prevalent in people's homes. These devices create new potential attack surfaces in people's homes, and therefore it is important that the manufacturers are taking the appropriate measures to secure these devices. The motivation for this work was to determine if these measures were being taken since people could be unknowingly purchasing smart home IoT devices with security or privacy vulnerabilities.

Smart home IoT devices that are available to consumers were purchased and analyzed for this paper. Some of these devices were found to contain privacy vulnerabilities. Therefore, some smart home IoT devices on the market contain a privacy vulnerability, which is they do not encrypt transmitted data over a local WiFi network, and therefore can be subject to a man in the middle attack.

The privacy and security of seven different smart home IoT devices were analyzed including smart light bulbs, WiFi thermostats, a smart plug, and the Amazon Echo. It was found that four of the seven devices do not encrypt transmitted data over the local WiFi network connection, which is a privacy vulnerability. For two of these devices, the transmitted data could be visible in plain text, which can be easily deciphered by an attacker. Three of the four devices that contain a privacy vulnerability were also vulnerable to replay attacks, meaning replaying recorded packets causes the device to perform an action such as turn the lights on. It is discussed how the data obtained due to this privacy vulnerability can be used to track a user's lifestyle habits. From this data an attacker can infer if the user is currently home or away. Lastly, solutions to these vulnerabilities are presented, which includes encrypting the communication data that is transmitted between the different nodes of the smart home IoT devices. For devices that use a point-to-point type of architecture, lightweight encryption techniques are needed and discussed.

Chapter 1 Introduction

Connected technology has started to become increasingly common in people's homes. These connected devices, which can be found in what has been dubbed the "smart home," has made life more convenient for some people and also made it easier for elderly or disabled people to live independently. While these are inarguably beneficial aspects of these IoT connected devices, there are some drawbacks, which include possible threats to privacy and security of personal data of the users. The smart home has essentially taken devices, such as thermostats and light bulbs, which were never connected to the Internet, and added the ability to do so. The result has been that these devices can be controlled remotely by a user, collect and transmit data to a server, and perform actions automatically based off commands from a server or the collected data.

Connecting devices to the Internet has also created a new wireless cyber attack surface in the home that was not present a decade ago. Data is now transmitted between a smartphone or server and the smart home device, leading to the possibility of a malicious entity stealing or injecting data that can cause harm to the user. While the user has some responsibility to protect themselves against these types of threats, by taking actions such as creating strong passwords to be used with their devices and local WiFi network, the bulk of this responsibility falls on the manufacturers of these IoT smart home devices. Manufacturers must ensure that they are taking the appropriate actions when designing a connected smart home device to prevent leakage of private data or disallow the ability of an attacker to cause any harm.

This research was performed and this paper was written on the basis that some manufacturers were not designing cyber security into IoT smart home devices. Therefore consumers have been purchasing smart home IoT devices that contain cyber security and/or privacy vulnerabilities. Companies attempt to be first to market with their smart home IoT device in order to gain an initial large share of the market [1]. Often the consequence of this is cyber security measures are not included in the original design, and it is often difficult to add retroactively after devices are on the market. In addition, after a smart home IoT device has become popular, such as the Nest Thermostat, more inexpensive alternatives are soon to follow

and are released to the market. A sound cyber security design and implementation can add cost to an IoT smart home device, so therefore measures to protect privacy and cyber attacks may not be taken in these instances [2].

In this paper, several IoT smart home devices are analyzed for security and privacy vulnerabilities including smart light bulbs, smart thermostats, a smart plug, and the Amazon Echo. The results of the analysis are several of these devices have a privacy vulnerability, which is transmitting unencrypted data over a local WiFi network. Since this data is transmitted by everyday devices in the home, it is argued that attackers can use this data to learn about the lifestyle of the user, such as when they leave the home. This is obviously a privacy and safety concern.

This paper is organized as follows: chapter 2 will cover related work, chapter 3 will cover the analysis and findings on the purchased smart home devices, chapter 4 will discuss the vulnerabilities that were found, chapter 5 will discuss the solutions to those vulnerabilities, and a conclusion will be provided in chapter 6. The data obtained from the analysis of the smart home devices will be shown in more detail in Appendix A.

Chapter 2 Related Work

Research has been done recently in the area of smart home IoT cyber security. Similar studies have been performed where researchers analyzed smart home devices for security and privacy vulnerabilities. In one such study, authors examined the Nest Thermostat and Nest Protect, which is a smart smoke alarm. In this study, the authors analyzed the network traffic of the Nest devices to see if they were able to determine if the user was home or away. From the network traffic, the authors could determine when a transition between “Home” and “Auto Away” modes occurred 67% of the time, and when a transition between “Auto Away” and “Home” occurred 88% of the time. It was also concluded that there is no efficient method to protect against this type of side channel attack. The authors explained that it has been documented in the past that the Nest Thermostat encrypts all data over the local WiFi network, which is the attack surface analyzed in this paper [3]. No analysis was performed on side channels for this paper.

In reference [4], authors also analyzed the Nest Thermostat for existing vulnerabilities in order to find out how simple it would be for an attacker with limited technical knowledge to exploit. These vulnerabilities included packet analysis, credential attacks, and downloading malicious software onto the Nest device. Both downloading malicious software and analyzing the packets failed to deliver any results, since the device would not allow the authors to gain root access and all communication data from the device was encrypted. However, using the Python script that the authors obtained online, they were able to request the current schedule mode of a thermostat from the Nest servers. The authors determined that if they ran this script at scheduled intervals, they could get an accurate picture of when the user was home or away. Since this script is available publicly and does not take much expert knowledge to run, it becomes quite simple for an attacker to learn the habits of someone with a Nest thermostat.

The authors in [5] analyzed similar devices as this paper including the Philips Hue smart light bulb and the Belkin WeMo smart plug. The Nest smoke alarm was also analyzed, but that is not covered in this paper. Some of the same vulnerabilities were found, including the lack of encryption on data transmitted from the device over WiFi, which was discovered to be in the

Philips Hue smart light bulb and Belkin WeMo smart plug. The authors also demonstrated that a replay attack was possible with the Philips Hue device by showing that the light bulbs could be turned on and off using the data that was collected. It was not known at the time of performing the data collection and analysis for this paper that these vulnerabilities had already been discovered as stated by reference [5]. The authors also did not analyze multiple brands of a type of smart device (for example they did not look at any other smart light bulbs outside of the Philips Hue).

After performing the experiments for this paper, research was done to find additional evidence of analysis of the Philips Hue for security and privacy vulnerabilities. It was documented on a blog in reference [6] that the Philips Hue did not encrypt communication of data over local WiFi. This vulnerability has been documented since 2013, where the author in reference [7] initially discovered it.

Because the Philips Hue is one of the most popular smart light bulbs on the market, it has received a lot of attention in the security community. Researchers also analyzed the communication interface between the Philips Hue hub and the smart light bulbs in reference [8], which use the Zigbee protocol. The authors were able to control the smart light bulbs from 350 meters away by exploiting a vulnerability in the Zigbee interface. This attack surface is not considered in this paper, however, as only communication over local WiFi is analyzed.

Chapter 3 Experimentation

For this study, 5 different devices were analyzed for cyber security and privacy vulnerabilities. This section is organized by the category of smart home IoT device, such as smart light bulbs, and will provide the individual analysis for each device. A discussion on the findings for each category of device will follow the analysis.

3.1 Experiment Setup

Prior to discussing the analysis of each smart home IoT device, the process used to capture and analyze data should be covered. The main tool used to do so is Wireshark, which is a tool that can be used to capture data packets that are transmitted over Ethernet or Wi-Fi. Version 2.2.7 of the Wireshark tool was used on a MacBook Pro for this investigation. The local WiFi network was setup to use WPA2-PSK with AES encryption.

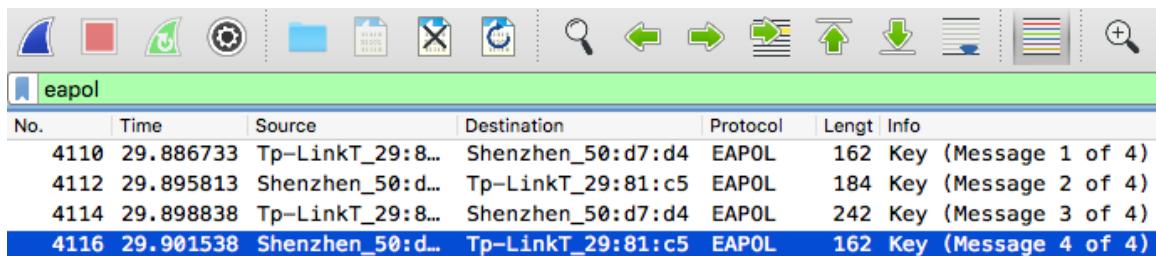
Because the devices that were analyzed for this paper communicate over WiFi, the 802.11 radio on the laptop was utilized to capture transmitted packets. Wireshark is setup to capture data in “Monitor” mode, meaning it monitors all 802.11 (WiFi protocol) traffic that can be captured by the laptop’s radio. Since these experiments were performed in an apartment setting, this meant that there was a lot of unneeded wireless traffic that had to be filtered out. Monitor mode was needed in order to capture the data transmitted between two different devices outside of the laptop, which for these experiments was typically a smartphone and an IoT smart home device.

Since Wireshark was setup in Monitor mode, it would only capture the encrypted data transmitted over the local WiFi networks in the near proximity. Wireshark had to be configured to decrypt the data transmitted over the local WiFi network that the smartphone and smart home IoT devices were connected to. To do so, the SSID and password of the local network were entered into Wireshark. During a data capture, Wireshark has to capture the 4-way handshake that is used in WPA2-PSK (AES), which is transmitted using the EAPOL protocol. With all four EAPOL packets, the SSID, and network password, Wireshark can start decrypting the data transmitted by a device over the local WiFi network. Wireshark will show the IP address of the

transmitting device as the source of the data. The decrypted version of the data will then become available.

Because the 4-way handshake must be captured to decrypt data, it is required that Wireshark is running and monitoring traffic when the smart home IoT device joins the local WiFi network. The easiest way to do this is to power off the device, start the Wireshark data capture, and then power on the device. If the 4-way handshake is captured, then the encrypted data will start appearing in the tool's interface. If not, then this process needs to be repeated until the decrypted data is being captured. It can take several trials in order to capture all four EAPOL packets. It appeared to be more likely for Wireshark to miss one of the EAPOL packets at times of increased wireless traffic, such as in the late evening.

To verify that Wireshark has captured all four EAPOL packets and has started to decrypt the wireless data, it is useful to filter out the rest of the wireless traffic on both the local network and the other WiFi networks in the vicinity. In order to do so, Wireshark was configured to filter on data transmitted or received by the MAC address of the IoT smart home device. The MAC address is often written on the device itself or included in the documentation. It can also be easily found from the IP address currently assigned to the IoT smart home device.



No.	Time	Source	Destination	Protocol	Lengt	Info
4110	29.886733	Tp-LinkT_29:8...	Shenzhen_50:d7:d4	EAPOL	162	Key (Message 1 of 4)
4112	29.895813	Shenzhen_50:d...	Tp-LinkT_29:81:c5	EAPOL	184	Key (Message 2 of 4)
4114	29.898838	Tp-LinkT_29:8...	Shenzhen_50:d7:d4	EAPOL	242	Key (Message 3 of 4)
4116	29.901538	Shenzhen_50:d...	Tp-LinkT_29:81:c5	EAPOL	162	Key (Message 4 of 4)

Figure 3-1 EAPOL Message Capture in Wireshark

Once Wireshark has started capturing and decrypting data, the reverse engineering process can begin. The process followed for this paper was to perform specific actions with the device, write down the timestamp that is recorded in Wireshark, and repeat several times to ensure the recorded data is correlated with the action. Once the data is recorded, Wireshark will show the data captured at the IP address of the smart home IoT device. The data can then be viewed in several forms from the raw hex bytes to the ASCII version of the data.

The tool that was used to view the binary data is Hex Fiend on the MacBook Pro. Hex Fiend shows the hex version of the binary data and can also perform a binary comparison of two

different files. The binary comparison was used to verify that data transmitted at two different occurrences was identical in some scenarios.

3.2 Smart Light Bulbs

The first category of devices that were analyzed was smart light bulbs. These devices are starting to become increasingly popular in the home as it makes it easier to control lights. Smart light bulbs are fairly simple relative to other connected IoT smart home devices. They essentially allow a user to control a light bulb via their smartphone. The light bulbs can be turned on or off, and usually offer a dimming feature as well. Some smart light bulbs can also change color, though none of the products analyzed for this paper offered this feature. For this paper, three different smart light bulbs were analyzed including the TP-Link Smart LED Light Bulb, Philips Hue Smart Light Bulb, and Sengled Element Classic Smart Light Bulb.

3.3 TP-Link Smart LED Light Bulb

The TP-Link Smart LED Light Bulb was the first device that was analyzed. The TP-Link Smart LED Light Bulb does not require a hub to be used, as is the case with many smart light bulbs, which can be an attractive reason for consumers to purchase this device. Instead this device can be controlled directly from the “Kasa” app that can be installed on iOS or Android devices. The Kasa app allows the user to turn the light bulbs on or off, and can also dim the light bulbs to a preferred lighting. Whenever the user changes the lighting using the smartphone app, the app transmits data directly from the smartphone to the smart light bulb. This is shown in figure 3-2.

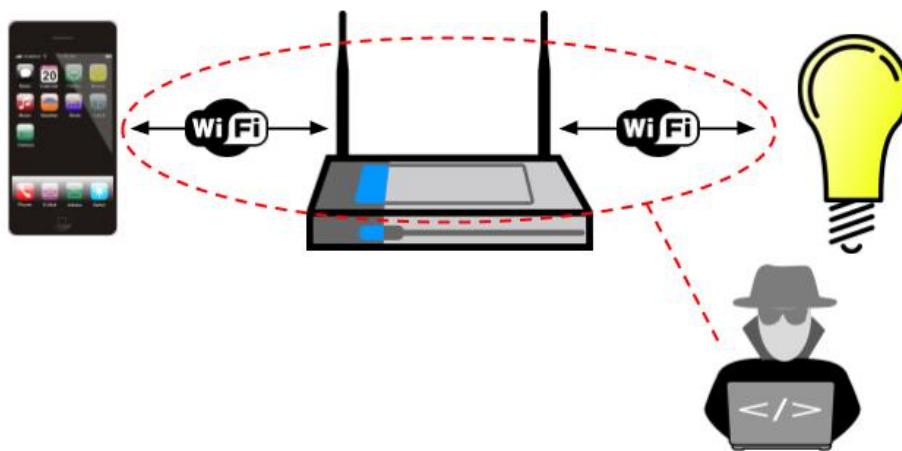


Figure 3-2 TP-Link Smart LED Light Bulb Communication Diagram

To reverse engineer the TP-Link Smart LED Light Bulb, Wireshark was setup to listen to the packets as they were transmitted from the smartphone to the light bulb. The smartphone app was used to turn the light bulb on and off and the timestamp in Wireshark was recorded to ensure packets could be correlated to the action that was performed on the smart device. This was repeated several times so that a pattern of the transmitted data could be obtained.

After taking a look at the resulting data, it became immediately apparent that it was not transmitted in plain text and was either encoded for the application or encrypted. However, the data that was transmitted to turn the light bulb on was identical for each recorded timestamp. The same was true for turning the light bulb off. A binary comparison over the transmitted data between different timestamps was performed in order to prove this. In addition, this same experiment was run with a different smartphone and smart light bulb. The same results were obtained, meaning the data was identical and was independent of the smartphone or smart light bulb used to transmit or receive the commands. Figure 3-3 shows the data transmitted to turn the smart light bulb on.

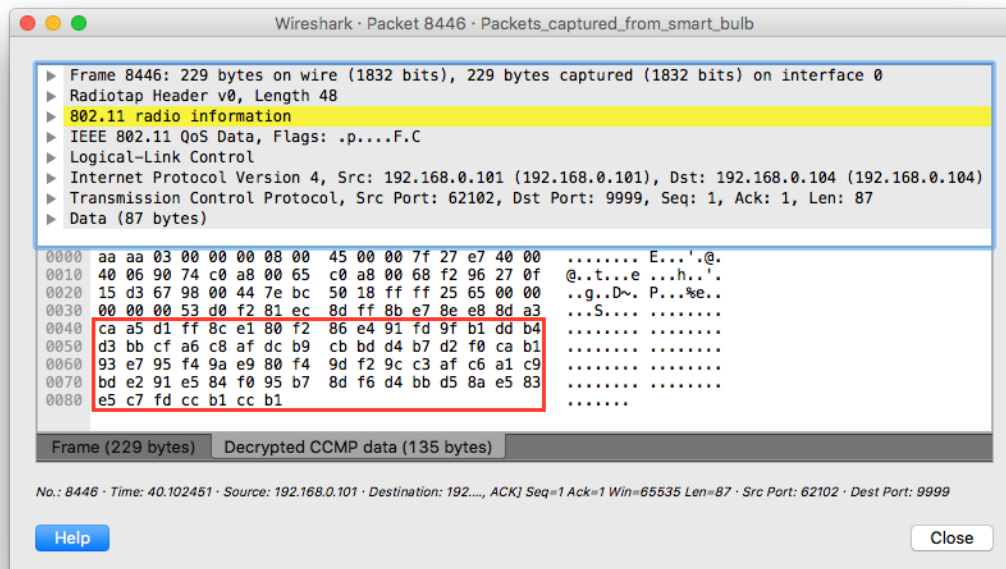


Figure 3-3 Data Transmitted From Smartphone to Turn Smart Light Bulb On

The next step was to determine if a replay attack of the transmitted data was possible. To do so, the recorded transmitted data was replayed from a MacBook Pro by using the terminal to pipe the data to the IP address of the smart light bulb and the port that it was listening on. The

replay attack was successful, meaning the laptop could be used to turn the smart light bulb on and off.

The experiment run on the TP-Link Smart LED Light Bulb proved that there are two vulnerabilities: data transmitted over the local network is not encrypted using temporal data and there is no authentication of the device sending the data. It is known that the data is not encrypted using temporal data since the same data is transmitted every time. If the transmitted data were a ciphertext then it would be the result of encrypting the same plain text every time. Therefore it cannot be concluded that no encryption is used, however it is still possible to determine the commands that have been sent to the smart light bulb. Because a replay attack was possible from a laptop, there is no authentication of the device that is currently sending the command. In addition, there are no security measures in place to prevent a replay attack, such as a packet counter.

Lastly, the original experiment was run on firmware version 1.1.2, which is the version the smart light bulbs were shipped with. The firmware was updated to the latest at the time, which is version 1.4.3, and these vulnerabilities were confirmed to still exist in the product. Therefore these vulnerabilities were still present in the TP-Link Smart LED Light Bulbs at the time this paper was written.

3.4 Philips Hue Smart Light Bulb

The Philips Hue Smart Light Bulb operates differently than the TP-Link Smart LED Light Bulb. The Philips Hue system uses a hub, so there is no direct communication between the smart phone and the smart light bulbs. Instead, the smartphone app sends commands to the hub via WiFi and the hub in turn communicates with the smart light bulbs via the Zigbee protocol. For the purpose of this paper, only the WiFi attack surface was considered, though there have been known vulnerabilities in the Zigbee attack surface as shown in reference [8]. Figure 3-4, below shows a diagram of the Philips Hue system and the attack surface that was monitored for this paper.

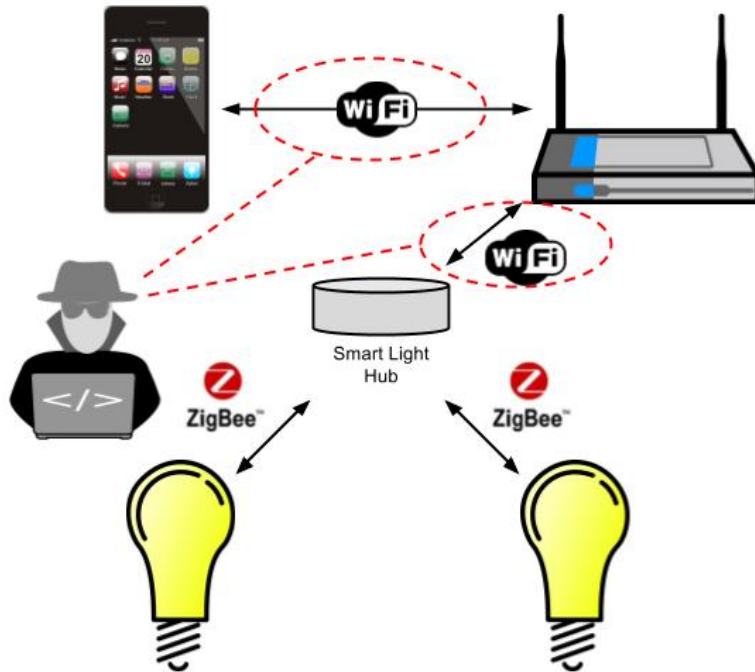


Figure 3-4 Philips Hue Communication Diagram

The same experiment was performed with the Philips Hue Smart Light Bulb. Wireshark was used to collect data while the Philips Hue was turned on and off using the smart phone app. From recording the timestamps and then examining the data, it became clear which data packets were transmitted to control the Philips Hue smart light bulb. For this experiment, the data transmitted from the smartphone to the Philips Hue hub and vice versa was captured.

The results of the experiment showed that the data transmitted by the smartphone to the Philips Hue hub was not encrypted and furthermore was sent in plain text. The response from the hub was transmitted in plain text as well. The Philips Hue uses the JSON format for transmitting data and is always listening on port 80 for commands sent from the smartphone. Further analysis of the data shows that in order to turn a smart light bulb on or off, the smartphone sends a command to a “group.” The Philips Hue hub then broadcasts that command to all smart light bulbs that are in that group. Data was collected using two different smart light bulbs on different days, and the only observed difference between the two was the IP address of the Philips Hue bridge had changed.

The next step was to see if a replay attack was successful using the data that was recorded by Wireshark. Similar to the TP-Link Smart LED Light Bulb, the data was piped to the IP address of the Philips Hue hub, which could be found easily since the MAC address was known.

The result was the replay attack was successful and therefore the smart light bulbs could be controlled from the terminal on a laptop. Because the data transmitted between the smartphone app and Philips Hue bridge is in plain text and therefore visible to an attacker, it can be manipulated to find and control groups of light bulbs that may be used in a home or small business.

The firmware of the Philips Hue was updated from 1705121051 to the latest at the time of the analysis, which is 1707040932, and the same vulnerabilities could be observed. With both the firmware version the Philips Hue was shipped with and the updated firmware version, plain text data could be recorded and replay attacks were successful. The lack of encryption on the Philips Hue has been discovered by other sources and well documented at this point in references [5], [6], and [7]. This was not known prior to purchasing the Philips Hue for this paper and running this experiment.

3.5 Sengled Element Classic Smart Light Bulb

The last smart light bulb that was analyzed for this paper was the Sengled Element Classic. With the Sengled Element Classic Smart Light Bulb, the smartphone app titled “ElementHome” can be used to control the light bulbs. Similar to the Philips Hue topology, the Sengled Element Classic Smart Light Bulb uses a hub, which communicates with the smart light bulbs via the Zigbee protocol. Where the Element Classic differs, however, is the smartphone app does not communicate directly with any other Sengled Element device over the local network. Instead the ElementHome app sends commands to a server, which then communicates with the Sengled Element Classic hub. The hub then relays those commands to the smart light bulbs. For the sake of this paper, only the communication between the smartphone and the server, and the sever and the hub were considered. The communication between the hub and the smart light bulbs was not analyzed. Figure 3-5 shows the communication in order to control the smart light bulbs in the Sengled Element Classic system.

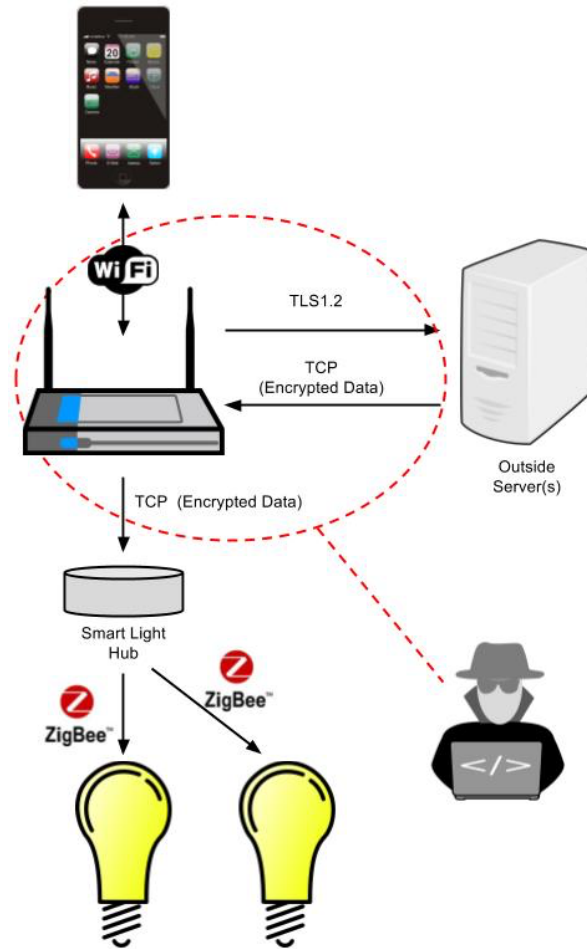


Figure 3-5 Sengled Element Communication Diagram

Data was captured using Wireshark and analyzed to determine if any security or privacy vulnerabilities exist in the Sengled Element Classic Smart Light Bulbs. The data transmitted by the smartphone and received by the Sengled Element Classic server were isolated. It was determined that the communication between the smartphone and the Sengled server is encrypted using the TLSv1.2 protocol. Anytime the ElementHome app on the smartphone was used to send a command to the smart light bulbs, a TLSv1.2 communication session could be seen in Wireshark. The communication from the server to the Sengled Element hub used the TCP protocol. TCP packets of data were transmitted from the server to the Sengled Element hub in order to turn the smart light bulbs on and off. Each time the smart light bulb was turned on and off, different data was transmitted from the server. Therefore, it appears that the data in the TCP packets transmitted from the server are all encrypted via an unknown protocol. The vulnerability

that exists in both the TP-Link Smart LED Light Bulbs and Philips Hue Smart Light Bulbs is not present in the Sengled Element Classic Smart Light Bulbs.

One observation that was made during this analysis is that the Sengled Element Classic appears to use an Amazon Web Services (AWS) server (see figure 3-6) for some communication. AWS offers secure communication through protocols like TLSv1.2. Using an AWS server may make it easier to setup the infrastructure for encrypted communication between the different entities in the system. To include an AWS server, however, Sengled chose a different system architecture than Philips and TP-Link by designing their system to have all communication go through an external server. This makes it easier to encrypt data, however it also makes the product reliant on the AWS service. It can also degrade the performance since communication is now occurring outside of the local network, which can cause an increased latency.

No.	Time	Source	Destination	Protocol
10143	61.904073	192.168.0.105	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	TCP
10152	61.959775	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	192.168.0.105	TCP
10154	61.960784	192.168.0.105	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	TCP
10168	62.059476	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	192.168.0.105	TCP
10171	62.093995	192.168.0.105	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	TCP
10224	62.400023	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	192.168.0.105	TCP
10226	62.400825	192.168.0.105	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	TCP
10257	62.434711	192.168.0.105	ec2-34-212-15-42.us-west-2.compute.amazonaws.com	TCP

Figure 3-6 Sengled Communication with Amazon Server

Though the TCP packets transmitted by the external server all appear to be encrypted, it was still possible recorded packets could be used for a replay attack. The packets that were collected during the experiment were replayed, however the attack was unsuccessful. Therefore the Sengled Element has measures in place to prevent replay attacks.

3.6 Smart Light Bulb Discussion

As mentioned in the previous sections, three different smart light bulbs were analyzed for security and privacy vulnerabilities. The results have been summarized in table 3-1 below. A privacy vulnerability, which is the lack of encryption on the data transmitted over the local WiFi network, was found in two of the three smart light bulbs that were analyzed. The Sengled Element Classic did not have this privacy vulnerability so therefore it is considered to be the most secure out of all three. The Philips Hue was the worst offender since the transmitted data is visible in plain text, and therefore does not take as much technical expertise to view the data. Though the TP-Link Smart LED Light Bulb does not transmit data in plain text, it does still contain this privacy vulnerability and can be subject to a man in the middle (MITM) attack.

Smart Light Bulb	Does Not Encrypt Data	Data Visible in Plain Text	Replay Attack Successful	Vulnerable to MITM Attack
TP-Link	Unknown	No	Yes	Yes
Philips Hue	Yes	Yes	Yes	Yes
Sengled Element	No	No	No	No

Table 3-1 Smart Light Bulb Security and Privacy Analysis Results

One may not think that being able to see the data transmitted by a smart light bulb is a problem since all one has to do to see the state of the lights is look through a window. However, the issue with this privacy vulnerability is that it can be used by an attacker to learn about the lifestyle of the user. With this vulnerability it becomes possible for an attacker to record the data transmitted by a smartphone to a smart light bulb via a man in the middle attack. If that data were to be recorded over a longer period of time, an attacker could infer that the user is not home during the periods when the light bulbs are turned off. Though it is generally known that most people are not home during the weekday and most businesses are closed at night, recorded smart light bulb data could show any deviations from the norm. This information could be used for a break in or just to track the user. Outside of the use case of the malicious attacker, IoT smart home devices should take as much precautions as possible to not leak user data.

The limitation of the argument in the previous paragraph is that an attacker needs access to the local WiFi network in order to record data from the TP-Link or Philips Hue smart light bulbs. This is an obstacle that could stop novice attackers, however, attackers with more advanced technical knowledge could overcome this by exploiting some of the known vulnerabilities in the WiFi encryption protocols. This is discussed more in detail later in this paper, as it is relevant to the other IoT smart home devices that are also analyzed. There is also the use case where people either have weak passwords that can be easily guessed (such as “password” or the letters on the home row of the keyboard) or places where guests are given the password to a network, such as an AirBnb. In these scenarios, smart light bulbs cannot rely on the WiFi protocol to encrypt the data to keep it private. In addition, the fact that one of the smart light bulbs analyzed, the Sengled Element Classic, did encrypt all communication gives credence to the argument that user data transmitted by a smart light bulb application should be protected.

3.7 Smart Plug – Belkin WeMo

One smart plug was analyzed for this paper, which is the Belkin WeMo. The Belkin WeMo allows the user to control the power input to any device that plugs into the wall. This can be quite useful to a user since they can turn a “dumb” device into a smart one that can be controlled from their smartphone, such as a window air conditioning unit. The Belkin WeMo device works similarly to the TP-Link Smart LED Light Bulb since it essentially uses point-to-point communication over local WiFi as shown in figure 3-7. When the user turns the power on or off from the plug using the “WeMo” smartphone app, data is sent directly to the smart plug.

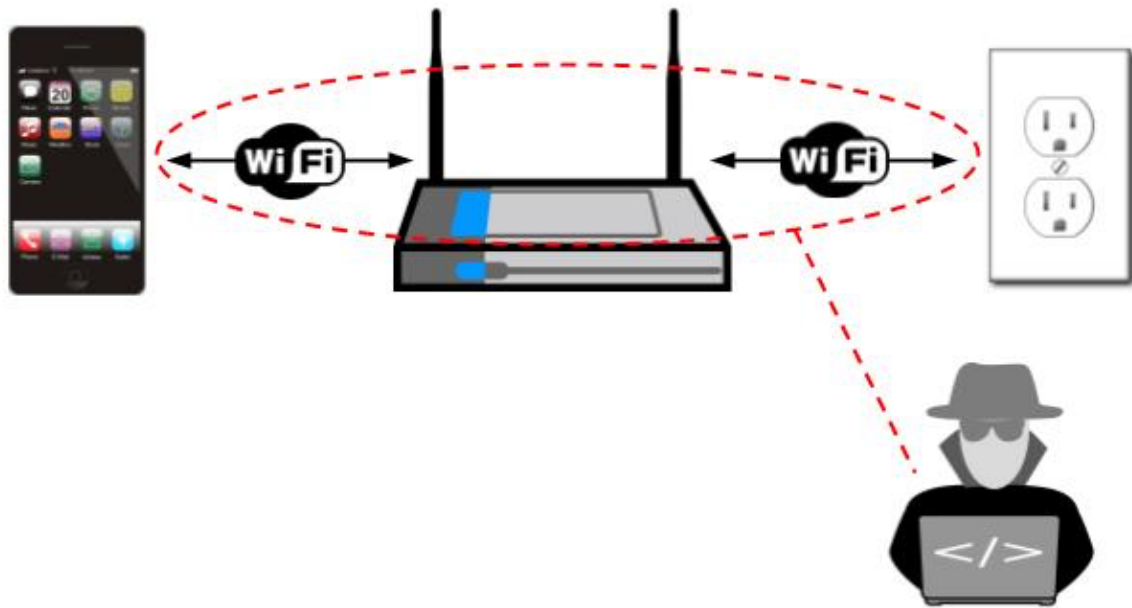


Figure 3-7 Belkin WeMo Communication Diagram

The data collection and analysis process was applied to the Belkin WeMo smart plug. The resulting data showed that data transmitted by the smartphone, and the response data from the smart plug, is in plain text and therefore is not encrypted. The Belkin WeMo uses an XML SOAP format for communication over the local WiFi network (see figure 3-8). In addition, a replay attack of the recorded data was successful, so therefore by piping data from a laptop the Belkin WeMo device could be controlled.

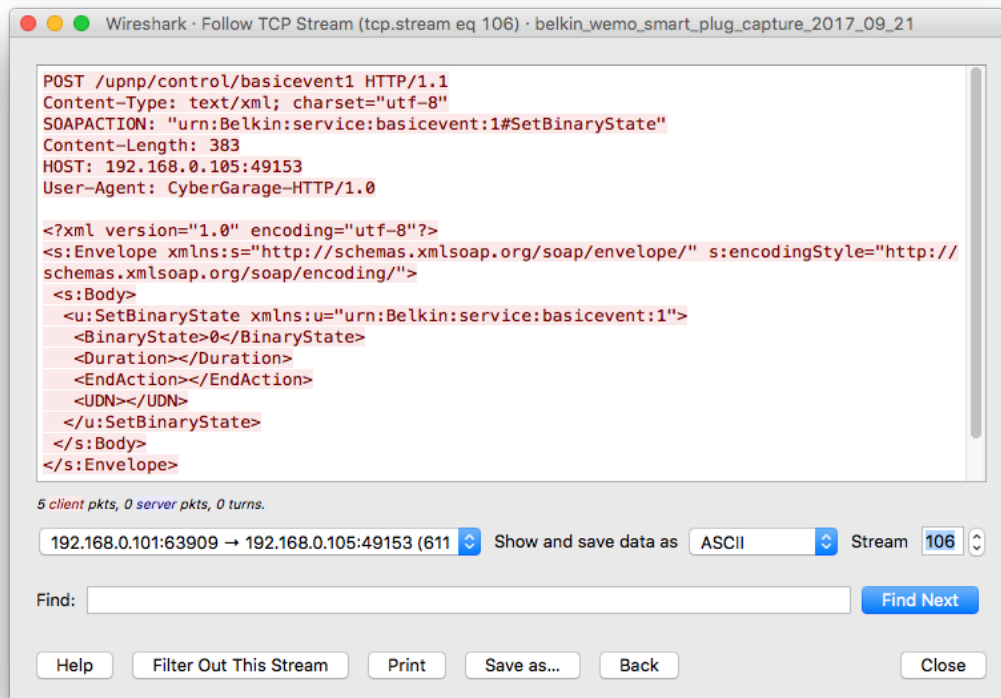


Figure 3-8 Belkin WeMo XML SOAP Format

The Belkin WeMo contains two vulnerabilities: Communication over the local WiFi network is not encrypted and there is no authentication of the user or device making a request. From the plain text data in the SOAP container it can be seen that there is no data used to describe the originator of the request. The information leaked by the device contains the transmitted command and an attacker would know the time the smart plug is used based on when the data was recorded. While this vulnerability is still a privacy concern, this is less of a safety concern for the user. The reason is an attacker from outside of the home would have no knowledge of what the plug is connected to or how often it is used. Turning on and off a plug without knowing what it is connected to may not give much information on user habits or lifestyle. However, from a privacy standpoint it is still user information that is essentially being broadcasted on a local network, which is a privacy vulnerability.

3.8 Smart and WiFi Thermostats

One of the original and more popular smart home IoT devices is the smart thermostat. Many people are interested in these devices because they can make their lives easier, help them

save money on energy costs over the long term, and also help to reduce the environmental footprint of homeownership. Not every single person who owns a smart thermostat purchased one for all of these reasons, however most certainly did so for at least one of them. Because smart or connected thermostats are designed to either learn the behavior of the user or run on a schedule that can be updated from a smartphone app, there is a large amount of personal information that is stored on these devices or servers, and possibly communicated on a daily basis.

For this paper two smart thermostats were analyzed: the Vine WiFi Thermostat and the Sensi WiFi Thermostat. In this section both thermostats will be analyzed to determine if either contain the privacy vulnerability that was discovered to be in some of the smart light bulbs and the Belkin WeMo smart plug. Arguably the most popular smart thermostat currently on the market is the Nest thermostat. It has been stated in reference [4] that the Nest thermostat encrypts data. Though other vulnerabilities have been discovered in the past such as the ability to determine the current mode from running a python script as described in reference [4].

3.9 Vine WiFi Thermostat

The Vine WiFi Thermostat essentially works in the same manner as a normal thermostat, but can be controlled via a smartphone app called “Vine Control.” The actual thermostat has a touchscreen interface. From the smartphone app the user can set the current temperature, turn on and off the AC or heat, and also set a schedule for the thermostat to run on. The ability to toggle the heat or AC from a smartphone is incredibly convenient as one often forgets to manually set the temperature as they are leaving the house. To communicate with the WiFi thermostat, the smartphone app first sends commands to the Vine server. The Vine server then processes the commands and communicates with the Vine WiFi thermostat. Because a WiFi thermostat just controls the heating and cooling of a house, it does not require as much of a real time response as a smart light bulb. Therefore the latency that comes with funneling all communication through a server does not have a severe negative impact on the user experience.

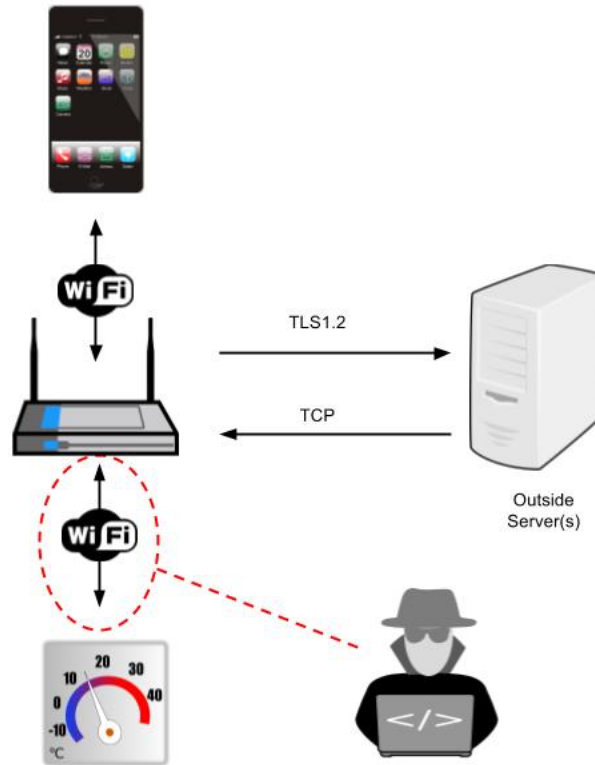


Figure 3-9 WiFi Thermostat Communication Diagram

For this paper the communication between the smartphone and the server, as well as the server and the Vine WiFi Thermostat were analyzed. From the recorded data, it was found that the communication between the smartphone and the server was encrypted via the TLSv1.2 protocol. This means that all data transmitted by the smartphone was found to not have the previously mentioned privacy vulnerability. However, after the analyzing the data that is transmitted by the server to the WiFi thermostat, it became apparent that some of the communication did not only lack encryption, but was also in plain text. This data included commands used to turn the thermostat on and off and to set the current temperature. The most concerning part of this, however, was even the command used to set the weekly schedule could be viewed in plain text by an attacker as shown in figure 3-10 (see Appendix A on more details on how to read the information in this data structure). From the weekly schedule, it would be quite simple for an attacker to infer that if the temperature is set to a higher value in the summer or to a lower value in the winter that those would be times when no one is home. This is a huge privacy concern as the server that communicates with the WiFi thermostat is essentially broadcasting information about the user in plain text. The limitation of this attack, however, is

this data is transmitted by the server and not the WiFi thermostat, meaning this private data can only be recorded when the user is making a change to their schedule. If the user changes their schedule once and leaves it for the entire season, then the attacker would have only one chance to record that communication to learn the user's lifestyle.

```

{"count":"181",
  "t_count":"0",
  "cmd":"device/set_model_info",
  "device_id":"845dd750d7d4",
  "timestamp":1508608716104,
  "mode":"1","limit":"60-85",
  "name":"Summer-01",
  "state":1,
  "model_id":195592,
  "data":
  {
    "unit":"F",
    "items1":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items2":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items3":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items4":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items5":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items6":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"480"},
      {"c":"0","t":"60","h":"840"},
      {"c":"0","t":"78","h":"855"},
      {"c":"0","t":"61","h":"870"},
      {"c":"0","t":"85","h":"1320"}],
    "items7":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"480"},
      {"c":"0","t":"85","h":"1320"}]
  }
}

```

Figure 3-10 Vine WiFi Thermostat Schedule Change Data

A replay attack was attempted on the Vine WiFi thermostat and was unsuccessful. From the data transmitted by the server, a packet counter appears to be used to prevent this. An attempt was made to manipulate this packet counter in order to control the Vine WiFi thermostat from a terminal on a laptop, however this was still unsuccessful.

The last step when analyzing the Vine WiFi Thermostat was to update the firmware and see if the privacy vulnerability still exists. The Vine WiFi thermostat was shipped with firmware version 1.2.3 and was updated to version 1.3.1 for this experiment. After doing so, it could be seen that all communication between the server and the WiFi thermostat was now encrypted. The privacy vulnerability that was found when first analyzing this device was fixed in a firmware update. This was a very interesting finding as it shows that Vine is aware of the privacy vulnerability that exists in their devices as shipped from the manufacturer and was able to resolve the issue retroactively, which can be difficult to do with cyber security solutions. The problem with this implementation is that the Vine WiFi Thermostat did not force the user to update the firmware in order to keep operating the device. The Vine WiFi thermostat was used for several days operating at firmware version 1.2.3 with the security vulnerability without the application ever forcing an update. Since a privacy vulnerability is present in firmware version 1.2.3 and is something that would not be easily noticeable by the user, there should be a mechanism that requires the user to update to a more secure version. Many users either would not bother to perform a firmware update, or may not have the technical capability of doing so. This would leave users operating the Vine WiFi Thermostat in a state where data containing information about their lifestyle could be broadcasted by the server. After updating the firmware, replay attacks were still not successful. Assuming that a packet counter was still being used but was now encrypted, it would make replay attacks increasingly more difficult to be carried out by an attacker.

3.10 Sensi WiFi Thermostat

The Sensi WiFi Thermostat works in a similar manner to the Vine WiFi Thermostat. The user can control the heat, AC, or temperature settings with the physical buttons on the thermostat or with the “Sensi” smartphone app. The communication model of the Sensi WiFi Thermostat is also the same in the fact that the smartphone communicates with a server, which then in turn communicates with the WiFi thermostat itself (see figure 3-9). Therefore all communication between the smartphone app and the thermostat is routed through a server. The attack surfaces

that were analyzed for this paper included the communication between the smartphone and the server, as well as the communication between the server and the WiFi thermostat.

Data was recorded using Wireshark and analyzed to determine if a privacy vulnerability exists. It was found that TLSv1.2 is used to encrypt communication between the smartphone app and the sever, meaning this attack surface did not contain the privacy vulnerability. It was also found that the data transmitted by the server to the Sensi WiFi Thermostat was encrypted as well. A replay attack was attempted on the Sensi WiFi thermostat, by replaying the encrypted TCP packets transmitted by the server, however this was found to have no effect. Therefore it was found that the Sensi WiFi thermostat encrypted all communication and does not contain the privacy vulnerability previously described in this paper.

3.11 Smart and WiFi Thermostat Discussion

Because a thermostat typically changes the environment based on the times that people are present or away, it is crucial that the transmitted data is protected and not available for an attacker to intercept via a man in the middle attack. Two WiFi thermostats were analyzed for this paper, one of which, the Vine WiFi thermostat, was found to have a privacy vulnerability in an older firmware version. Though this vulnerability was fixed in a later firmware version, this is still a flaw since the user is not required to update the firmware of the device. This is also an example of how cyber security was not included in the original design, but rather was added retroactively. Because the Vine WiFi thermostat system design routed all communication between the smartphone and thermostat through a server, it makes it easier to secure this communication channel via a firmware update. Response time was not measured before and after the firmware update, however encryption typically will degrade performance. Therefore it is quite possible additional latency was added to the response time of the WiFi thermostat. In the case of a thermostat, this should not cause the user experience to suffer since the response time to turn on the AC or furnace does not need to be immediate. However, for other types of smart IoT devices, adding cyber security retroactively could degrade performance enough to cause the user experience to suffer. This is especially true in cases where the microcontroller performance requirements did not take encryption or decryption into account during the initial design.

Instead of adding measures to protect the user's privacy retroactively, it is a better practice to include these measures into the device's initial design itself. This will eliminate the need for the user to update the firmware in order to protect their own private data, which is

beneficial since the user cannot always be trusted to perform firmware updates. Designing cyber security into the product ensures all the required sub-systems are in place for the solution. For example, there is often much that is needed on the backend for a cyber security solution including encryption services. This must be designed in such a way that keys used for encryption and decryption do not get re-used, and user experience does not suffer from degraded performance.

In the case of the Vine WiFi Thermostat, after the firmware update to version 1.3.1 was performed, encryption was added to protect the data of the user. However, instead of encrypting the entire packet of transmitted data, the Vine WiFi Thermostat instead encrypts the data within the actual packet. This is shown in figure 3-11. In the data packet there are two values, “k” and “v,” which can be read by an attacker. The “k” value always contains a number and the “v” value always contains encrypted data that cannot be easily read. However, one observation that was made during this experiment is that sometimes packets are transmitted with the same “k” value. This occurred for a “k” value of 77 and there was a match between the beginning portions of the “v” variable data. This suggests that the “k” value is referring to a key that is used to encrypt and decrypt the data, and that keys are also reused to send data securely. The match between the binary data can be seen in figure 3-12.

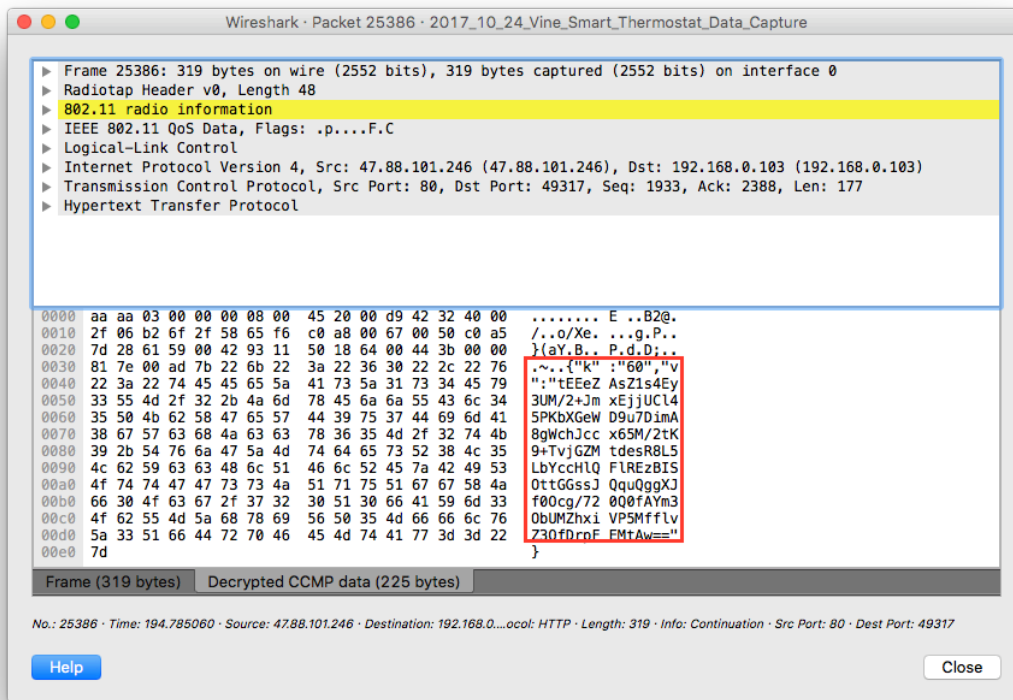


Figure 3-11 Vine WiFi Thermostat Encrypted Data

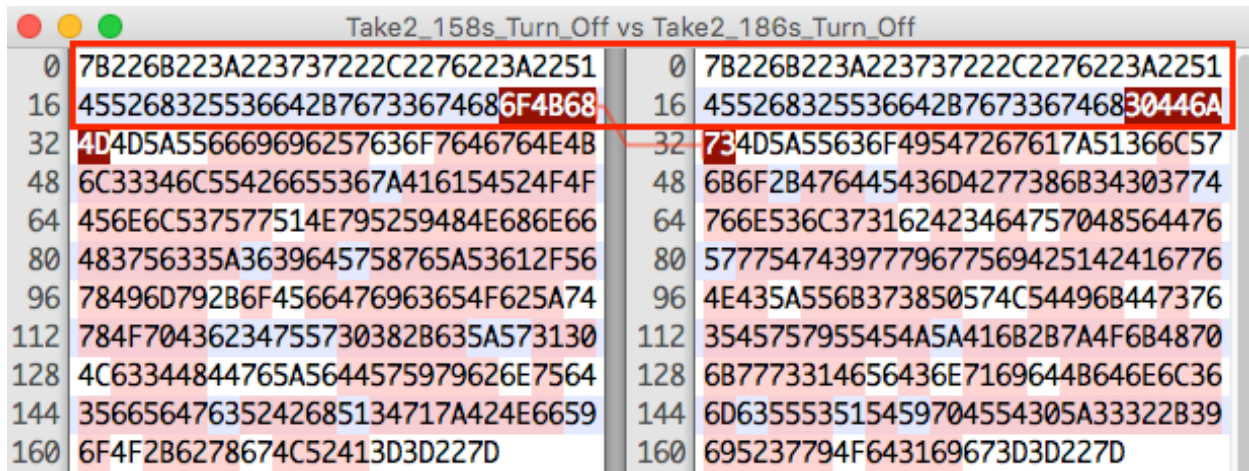


Figure 3-12 Comparison Between Vine WiFi Thermostat Encrypted Data Packets

3.12 Amazon Echo Dot

The last device analyzed for this paper is the Amazon Echo Dot. The Amazon Echo Dot is a voice assistant and is becoming an increasingly popular device with consumers. It can be used to control smart home devices via voice control and can interact with the user. “Alexa” is

the name of the voice personality on the Amazon Echo Dot that interacts with the user. The user can then ask or command Alexa to perform a task and the Amazon Echo Dot follows through on the action. The Amazon Echo Dot connects to external Amazon servers to retrieve data or determine which action should be taken.

Using Wireshark, data was recorded and analyzed when various voice commands, such as having the Amazon Echo Dot read through the news, were performed. When voice commands were used with the Amazon Echo Dot, spikes of transmitted and received data could be seen in Wireshark. It was found that all transmitted and received data from the Amazon Echo dot was encrypted using TLSv1.2. Therefore the data in the communication from the Amazon Echo Dot is protected from any prying eyes. This experiment did not find the previously mentioned privacy vulnerability in the Amazon Echo Dot since all data is encrypted. The communication system used by the Amazon Echo Dot can be viewed in figure 3-13. The user issues voice commands to Alexa on the Amazon Echo Dot and then all data communication occurs with Amazon servers via the TLSv1.2 protocol.

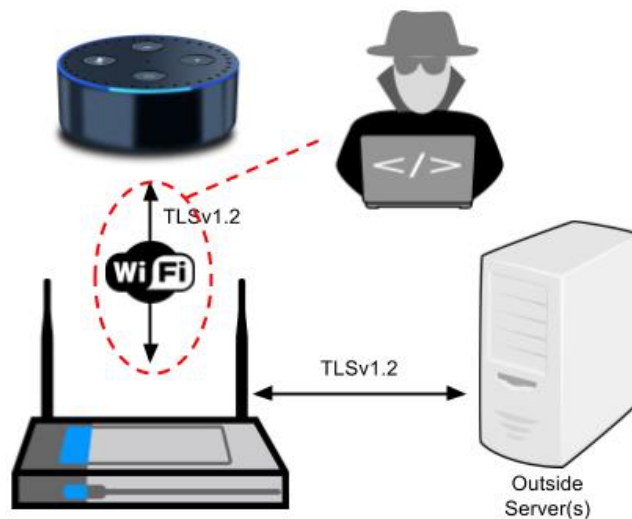


Figure 3-13 Amazon Echo Dot Communication Diagram

Given the size and technical expertise of Amazon, it is not surprising that Amazon would encrypt data transmitted by their IoT devices. It has also been documented in reference [9] that the TLSv1.2 protocol is used for secure communication, though this was not known at the time the experiment was performed.

As mentioned in a previous paragraph, when the Amazon Echo Dot is in use, a spike of communication data between the Amazon Echo Dot and the Amazon servers can be seen with

Wireshark. In addition, Wireshark was used to record the transmitted data when the Amazon Echo Dot is not in use. The data was graphed (see figure 3-14) and shows that about every five minutes, a spike of just less than seven kilobytes of data is transmitted by the Amazon Echo Dot. It is unknown what the exact content is of this data and it may be just a simple heartbeat message transmitted by the Echo Dot to the servers. However, this was an interesting finding as a lot of information can be contained in seven kilobytes of data.

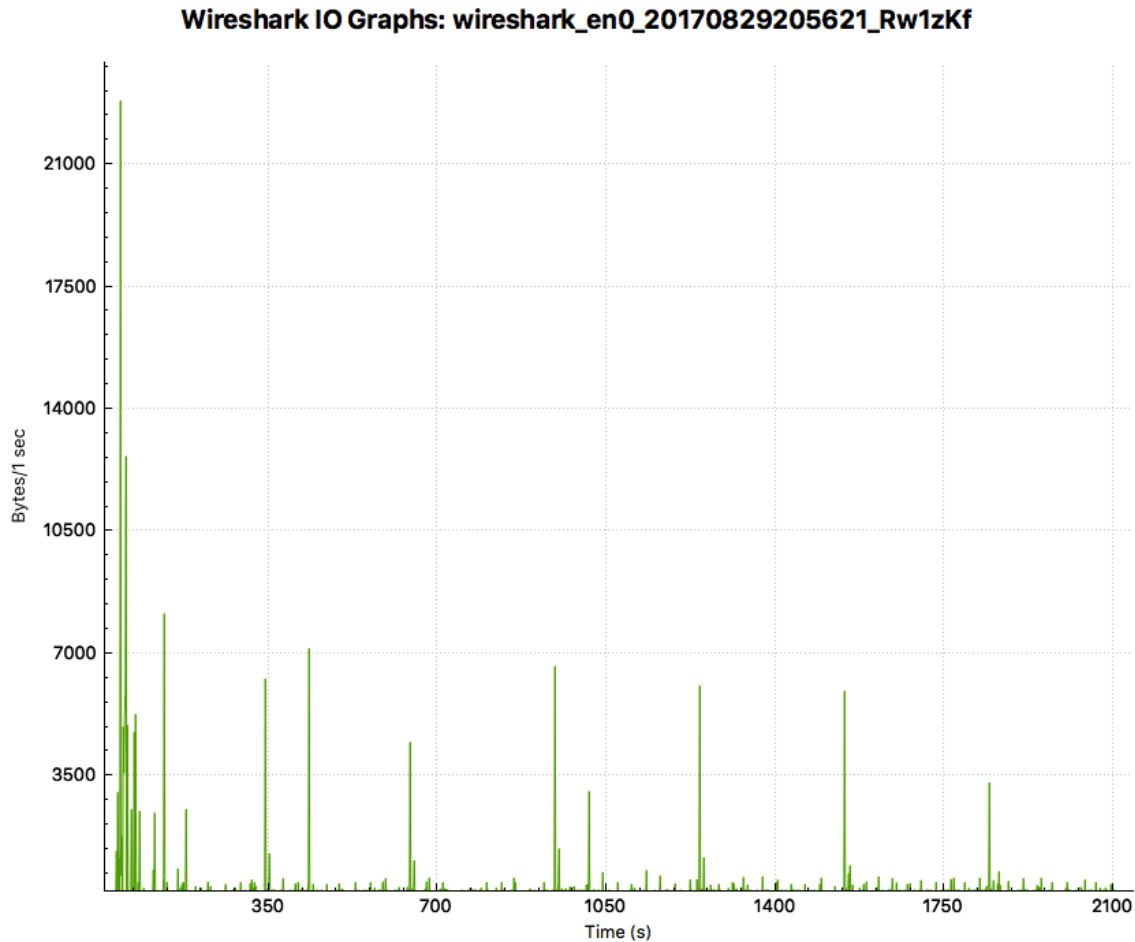


Figure 3-14 Data Transmitted By Amazon Echo Dot When Not in Use

Chapter 4 Privacy Vulnerability Discussion

4.1 Privacy Vulnerability

Seven smart home IoT devices were analyzed for this paper and four of the devices were found to have a privacy vulnerability. As shown in table 4-1, The TP-Link Smart LED light bulb, Philips Hue Smart light bulb, Belkin WeMo smart plug, and the Vine WiFi thermostat all had a lack of encryption on transmitted data over a local WiFi network. Instead, these devices relied on the local WiFi network to encrypt and protect their data. This chapter will discuss why not encrypting data transmitted over a local WiFi network is a privacy vulnerability. It will also discuss the reason the encryption provided by a WiFi network cannot reliably be counted on to protect user data and whether the TLSv1.2 protocol, which is used in several of the analyzed devices communication scheme, contains any vulnerabilities.

Smart Home IoT Device	Does Not Encrypt Data	Data Visible in Plain Text	Replay Attack Successful	Vulnerable to MITM Attack
TP-Link Smart Light Bulb	Unknown	No	Yes	Yes
Philips Hue Smart Light Bulb	Yes	Yes	Yes	Yes
Sengled Element Smart Light Bulb	No	No	No	No
Belkin WeMo Smart Plug	Yes	Yes	Yes	Yes
Vine WiFi Thermostat (v1.2.3)	Yes	Yes	No	Yes
Vine WiFi Thermostat (v1.3.1)	No	No	No	No
Sensi WiFi Thermostat	No	No	No	No
Amazon Echo Dot	No	No	No	No

Table 4-1 Security and Privacy Analysis Results

4.2 Lack of Encryption of Transmitted Data over Local WiFi Network

One concern that many individuals have over smart home IoT devices is privacy. Consumers have to trust the manufacturers and developers of these devices that they are protecting their data, only storing data that is absolutely required for the use of the device, and not exposing them to any cyber or physical threats. In the case of four of the analyzed devices, the privacy of the user's data was not being protected. Anyone who knows the credentials to the WiFi network that these devices are connected to can listen to the transmitted data and observe from the data when these devices are used. An attacker can also infer from the data transmitted by a smart home IoT device if anyone is home. If this data is tracked over a longer period of time, an attacker can determine that when the lights are turned off that the user is either asleep or has left the home. The thermostat optimizes the climate of the home to save money when the user is away and to keep the building at ideal temperatures when the user is home. This data conveys the lifestyle of the user to an attacker. An attacker could then use this information to decide when to break into a home, knowing that no one will be there at specific times. Homes that contain smart IoT devices that broadcast this information could become a target. This scenario may seem unlikely, however the privacy vulnerability discovered in four of the seven devices analyzed for this paper certainly makes this possible.

There are many more smart home IoT devices on the market that were not analyzed for this paper. Given that over half of the analyzed devices for this paper were found to have a privacy vulnerability, it is likely that this same privacy vulnerability exists in additional smart home IoT devices. Manufacturers are either overlooking this vulnerability, or are not making cyber security enough of a priority when designing these devices.

4.3 WiFi Vulnerabilities

Devices that do not encrypt the transmitted data over a WiFi network are essentially relying on the WiFi network encryption to protect the data. The problem with this is that some of the encryption protocols used for WiFi networks have known security vulnerabilities. In some cases, an attacker can gain authorized access to a network or can retrieve the key to read the transmitted data over the network. One of the most obvious security flaws in WiFi networks is the use of weak passwords. This can include passwords that are easy to guess such as "password" or the address of the home. However, according to reference [10] even passwords that are less than twenty characters can be broken. The issue with twenty characters is it can be

much tougher to remember, so people often choose a shorter and easy to remember password, which can be subject to a dictionary attack. Nowadays, the default password on many WiFi networks uses a long string of alphanumeric characters, which provides a sufficient level of security as long as the user does not change it.

The security protocols that are available for use on most home routers include WPA2-PSK and WEP. It has been well documented that the WEP protocol contains security flaws and therefore should not be used to protect a WiFi network. According to reference [11], in the WEP protocol packets can be both forged and replayed. Packets can also be modified in a manner where it cannot be detected. Lastly there is available software that can be used to brute force the key to the network, which would allow an attacker to gain access. Once the attacker has access to the WiFi network they would have the ability to take advantage of the privacy vulnerability found in four of the devices analyzed for this paper. In addition, many wireless routers still support the WEP protocol.

The WPA2-PSK protocol has largely been seen as more secure in comparison with the WEP protocol as long as the Advanced Encryption Standard (AES) protocol is used. WPA2-TKIP, which uses the Temporal Key Integrity Protocol (TKIP) has known vulnerabilities, but according to [12] was never meant to be a long term solution, but rather to overcome the vulnerabilities in WEP. WPA2-PSK using AES did not have any known vulnerabilities until it was recently discovered that some devices were vulnerable to a key re-installation attack. In the key re-installation attack, the attacker replays the third message in the four-way handshake used to authenticate a node onto a WPA2 network. This forces the node to re-install itself onto the network using an already used session key. When this occurs, the packet counter is reset back to the default value of zero, thus making the node vulnerable to replay attacks. In addition, packets can be decrypted meaning data transmitted from devices such as those analyzed for this paper can be seen by an attacker [12]. The devices analyzed for this paper were not analyzed by the authors of the referenced paper. Therefore it is not conclusive that these devices would be vulnerable to a key re-installation attack. Though the authors mentioned that any device that uses WiFi is vulnerable to some form of a key re-installation attack. Manufacturers of some WiFi devices have since released an update that provides a patch to this vulnerability, however this would require the user to perform a firmware update.

Because there are known security vulnerabilities in WiFi protocols, they cannot be completely relied on to protect the data transmitted by smart home IoT devices. When the vulnerability in the WiFi network is stacked on the vulnerability of a smart home IoT devices as seen in four of the seven analyzed, it gives an attacker the ability to read the data transmitted by the device without even knowing the network password. The result of this is the attacker can learn the lifestyle habits of the user.

4.4 Vulnerabilities in TLSv1.2 Protocol

The Transport Layer Security (TLS) protocol is used by many different types of applications for secure communication. As shown in earlier sections, the TLSv1.2 protocol is used in several of the IoT smart home devices that were analyzed for this paper including the Sengled Element smart light bulb, the Vine WiFi thermostat, the Sensi WiFi thermostat, and the Amazon Echo. In order to verify the security and privacy of the data for these devices, it was explored whether there were any known vulnerabilities in the TLSv1.2 protocol.

One vulnerability was discovered with the TLSv1.2 protocol and it only exists on servers that also support SSLv2 and 40-bit export cipher suites. This attack is referred to as Decrypting RSA using Obsolete and Weakened eNcryption (DROWN). It relies on the fact that servers that use the same certificates for different protocols and therefore the certificates used with the SSLv2 protocol, which is known to have a vulnerability, are also used with TLSv1.2. In the case of SSLv2 the 40-bit export cipher suite means that only 40 bits out of the 128 bits used for a key are encrypted, the remaining 88 bits are transmitted in plain text. The DROWN attack is able to exploit the weak 40-bit cipher to decrypt one in a thousand TLSv1.2 ClientKeyExchange messages, revealing the session key. An attacker can then use the session key to decrypt TLSv1.2 messages in that specific session. The authors of reference [13] estimate that about 33% of HTTPS servers are vulnerable to the DROWN attack. It is unknown whether any of the smart home IoT devices analyzed for this paper are vulnerable to the DROWN attack. Though SSLv2 is known to be obsolete so it is unlikely any of the analyzed devices are vulnerable to this attack.

There are other vulnerabilities that have been discovered to exist in TLS in the past and have been patched on some servers. In reference [14], each of the vulnerabilities are listed as well as the percentage of sites that contain those vulnerabilities. It was found that 37.1% of sites do not have adequate security when using the TLS protocol. However this does take into account the configuration of sites as well as sites that run different versions of TLS. It is unknown if any

of the servers used with the smart home IoT devices that were analyzed for this paper were scanned or if they would be in this group. Because the analyzed devices are all somewhat newer products, it would be unlikely they are communicating with servers that have not been patched for the latest vulnerabilities.

Chapter 5 Solutions to the Privacy Vulnerability

The problem that needs to be solved is a lack of encryption on the communication data used with smart home IoT devices. The simple answer to this problem is just to add encryption. While on the surface, this will solve the problem, the solution is more complicated because of the resource constrained nature of these devices. This chapter will discuss the challenges of encrypting communication with IoT smart home devices, methods that can be used to solve this problem, and finally a recommendation will be given.

Smart home IoT devices have limited processing power and can also have limited battery power. In the case of smart light bulbs, smart thermostats, and smart plugs, they all draw their power from the 120 V AC available in the home so battery power is not a concern. However, in order to manufacture smart home IoT devices at a competitive price, the devices were most likely designed with limited processors. In terms of cyber security, this adds a challenge to securing these systems with functionality such as encryption and decryption. For smart thermostats, light bulbs, and plugs, adding encryption to communication can potentially result in a delayed response time. The software used to control the device has to decrypt transmitted data from a server or smartphone and then encrypt the response, which can take a considerable amount of time depending on the encryption algorithm. Therefore, processing power is one of the main factors that needs to be considered when deciding how to secure smart home IoT devices.

The analyzed smart home IoT devices can be split into two different categories: devices that communicate with an external server and devices that do not have any communication with an external server. For devices where an external server is involved it is easier to solve the lack of encryption problem since the server is able to provide the backend solution for more complex encryption protocols. For this reason, the analyzed devices that used a server, which were the Sengled Element, Vine WiFi thermostat, the Sensi WiFi thermostat, and the Amazon Echo all used the TLSv1.2 protocol for communication between a smartphone and the server. Recent smartphones all have increased processing power as well and can support using the TLSv1.2 protocol without any noticeable degradation in performance. Communication between the server

and the smart device then encrypts TCP packets that are transmitted from the server to the IoT smart device. This is a secure solution, which is already being used by the Sengled Element, Sensi WiFi Thermostat, and the Amazon Echo. The Vine WiFi thermostat had a privacy vulnerability in an older version of firmware, however added encryption to the data inside of the actual TCP packet. The difference is the data is still visible within the packet, but the contents are encrypted. It is unknown what encryption protocol is used, and it is possible this is a secure solution. However, for smart home IoT devices that use a backend server for communication between a smartphone and the actual device, it is recommended to use TLSv1.2 for secure communication between the smartphone and the server and then to encrypt the entire TCP packet transmitted from the server to the smart IoT device.

For IoT smart home devices that use a point-to-point architecture, meaning a smartphone communicates directly with a device over a WiFi network, it is important that a lightweight encryption protocol is used. Encryption protocols used with PCs, such as RSA, are too resource intensive for applications such as a smart light bulb. This would especially be true for the TP-Link smart light bulb, where all processing power is contained in the actual light bulb. In addition, a packet counter should be used to prevent replay attacks. For devices such as a smart light bulb, or smart plug, the actual data transmitted is quite simple, and without a packet counter the contents may not vary by much. The packet counter ensures the plain text data changes and therefore the cipher text data will change as well, which adds a level of difficulty in determining the plain text.

Lightweight encryption typically refers to protocols that can be run on devices with fewer resources. This includes battery-powered devices and devices with constrained processors. Reference [15] analyzed several different lightweight encryption protocols and decided upon a few general rules. Overall, their conclusion was that specialized or configurable hardware, such as an FPGA, could be used to increase the processing time of the encryption protocols. The authors also conclude that stream ciphers are faster than block ciphers. However, for the point-to-point applications that were analyzed for this paper, which includes smart light bulbs and smart plugs, block cipher protocols are a better option than stream ciphers. The reason for this is the commands transmitted to the device are quite simple and just need to turn the device on, off, or set the dimming value in the case of a smart light bulb. A block cipher would create more variation in the ciphertext since it would provide encryption over the entire block of data. In

addition, the amount of data transmitted between the smartphone and the device is always known. According to reference [16], lightweight block ciphers include protocols with smaller block sizes, smaller key sizes, and less computationally complex operations. These devices only need to transmit a small amount of data so therefore large block sizes are not needed as well. While a standard protocol such as AES-128 may give sufficient security for IoT applications, there are even more “lightweight” protocols available that may provide better performance. If the performance is degraded on a smart light bulb or a smart plug to the point that it takes several extra seconds to turn on or off, the user experience will suffer. DESL and PRESENT are lightweight encryption protocols that have a 64-bit block size and would be sufficient for these applications. SIMON and SPECK are other options as well.

In general symmetric cryptographic algorithms require less resources than asymmetric cryptographic algorithms, which is why the suggested algorithms listed above are symmetric. However, reference [17] points out that the invention of Elliptic Curve Cryptography (ECC) has brought asymmetric cryptography to embedded devices. ECC requires less memory and computations than other asymmetric algorithms, such as RSA. The authors of [17] also suggest a scheme that uses identity strings for the public keys instead of certificates for asymmetric encryption. A base station is then used to generate the private keys. Stateful encryption is used, which means some of the computation for public key encryption can be re-used improving the efficiency of the process. The issue with this scheme is that it requires a base station, which is not always available for smart home IoT devices. For example, the TP-Link smart light bulbs do not use any hub or base station.

Chapter 6 Conclusion

The premise that some of the smart home IoT devices currently on the market are not secure and contain cyber security and/or privacy vulnerabilities was explored. Seven different devices were analyzed and it was discovered that four out of the seven devices did not encrypt communication over the WiFi network and could be subject to a man in the middle attack. This is a privacy vulnerability as it can leak data that can be used by an attacker when the user is home or away. Since only four devices were analyzed, definitive proof that this is an industry wide problem cannot be obtained from this analysis, however more than likely there are additional smart home IoT devices available for purchase that contain this same privacy vulnerability. A solution for two different types of smart home IoT architectures were discussed and recommended as a solution to this vulnerability. Both solutions include encrypting communication data so that the contents are not visible to an attacker in order to protect the privacy of the user. This will prevent the leakage of user data from the IoT smart home devices that have otherwise improved the lives of many people.

Appendix A: Collected Data from Smart Home IoT Devices

This appendix contains recorded data that was collected from the smart home IoT devices that were analyzed for this paper. This appendix is organized by device.

A.1 TP-Link LED Smart Light Bulb Recorded Data

The screenshots below contain the recorded data for the TP-Link smart light bulb. Figure A-1 contains the binary data that is transmitted from the smartphone to the TP-Link LED smart light bulb in order to turn it on. Figure A-2 contains the binary data that is transmitted from the smart light bulb to the smartphone after the light has been turned on. This is the response from the smart light bulb.

Hex	ASCII
0 00000053D0F281EC8DFF8BE78EE88DA3	S-ÚÁÏç`äÁéËçf
16 CAA5D1FF8CE180F286E491FD9FB1DDB4	•-`á·ÄÜÛ%ë"ü±>¥
32 D3BBBCFA6C8AFDCB9CBDD4B7D2F0CAB1	"°æ¶»∅<πÀΩ'Σ"±
48 93E795F49AE980F49DF29CC3AFC6A1C9	ìÁÿÛöÈÄÛùÚú√∅Δ°...
64 BDE291E584F095B78DF6D4BBD58AE583	Ω,ëÄÑ*ÿΣç^'°'äÁÉ
80 E5C7FDCCB1CCB1	Â«"Ä±Ä±

0 out of 87 bytes

Figure A-1: Turn Smart Light Bulb On (From Smartphone)

Hex	ASCII
0 000000AAD0F281EC8DFF8BE78EE88DA3	™-ÚÁÏç`äÁéËçf
16 CAA5D1FF8CE180F286E491FD9FB1DDB4	•-`á·ÄÜÛ%ë"ü±>¥
32 D3BBBCFA6C8AFDCB9CBDD4B7D2F0CAB1	"°æ¶»∅<πÀΩ'Σ"±
48 93E795F49AE980F49DF29CC3AFC6A1C9	ìÁÿÛöÈÄÛùÚú√∅Δ°...
64 BDE291E584F095B78DF6D4BBD58AE583	Ω,ëÄÑ*ÿΣç^'°'äÁÉ
80 E5C7FDCCCE0C2AFC0A4C1E3D9FB95FA88	Â«"Ä±-∅¿§i„ÿ'ÿ'ä
96 E584E8CAE6C4ACD9BC9EA494B89AE988	ÄÑÈ Èf"ÿ°ú§ÿÿöÈà
112 FC89FB9AEE87E886A49EAE82A0C3ACC0	,ä°öóáËÛúÆç†√"¿
128 AFDD82F693FE8EAC96A493A393BF9DFF	∅>ç^ì,é"ñ§ì£i∅ù"
144 8DE483EB9FF194E794B68CBD8DBD91B3	ç%Èÿü∅ìÁi∅à∅ç∅è≥
160 D6A4D689EA85E184A69CADC1ACD1	÷§÷äí∅·Ñ¶ú"-"-

0 out of 174 bytes

Figure A-2 Turn Smart Light Bulb On Response (From Light Bulb)

Figure A-3 contains the binary data transmitted from the smartphone to the TP-Link LED smart light bulb in order to turn it off. The response binary data from the TP-Link LED smart light bulb after it has been turned off is shown in Figure A-4.

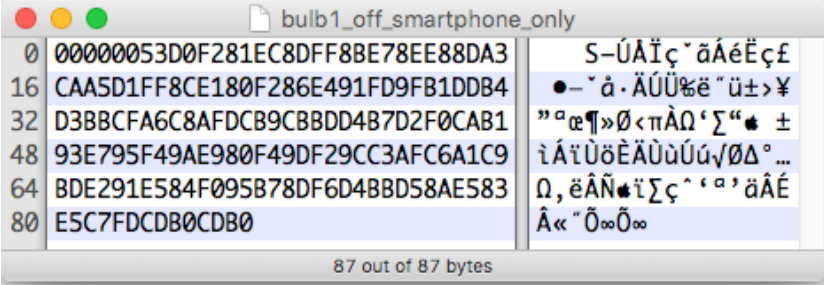


Figure A-3 Turn Smart Light Bulb Off (From Smartphone)

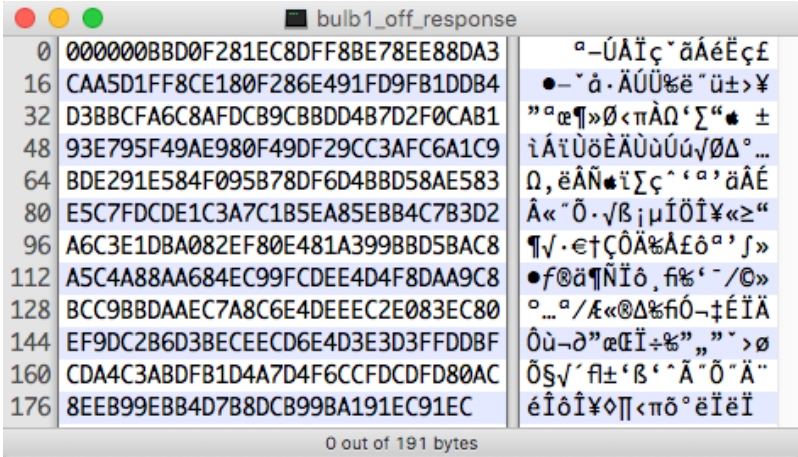


Figure A-4 Turn Smart Light Bulb Off Response (From Light Bulb)

A.2 Philips Hue Smart Light Bulb Recorded Data

This section contains the data that was recorded when analyzing the Philips Hue smart light bulb. Since it was found that data is transmitted in plain text with the Philips Hue application, both the binary data and ASCII version of the data is shown. Figure A-5 contains the binary data that is transmitted from a smartphone to the Philips Hue hub to turn the light bulbs in the group on. The ASCII version of that same data is then shown in Figure A-6.

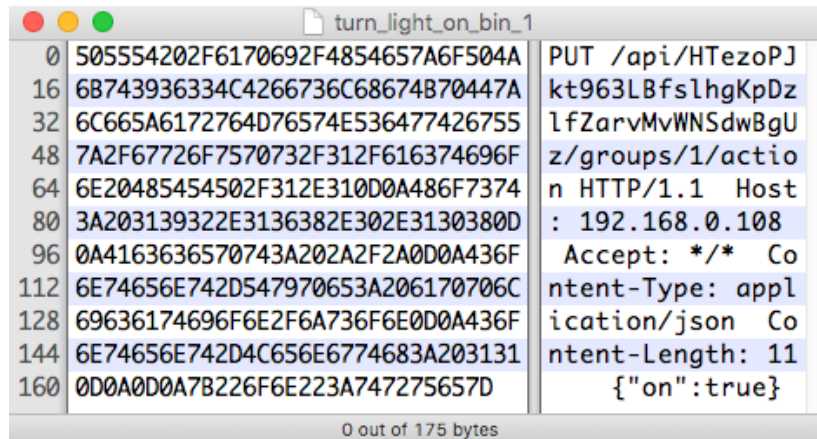


Figure A-5 Turn Smart Light Bulb On Binary Data (From Smartphone)

```
PUT /api/HTezoPJkt963LBfslhgKpDzlfZarvMvWNSdwBgUz/groups/1/action HTTP/1.1
Host: 192.168.0.108
Accept: /*
Content-Type: application/json
Content-Length: 11

{"on":true}:true}
```

Figure A-6 Turn Smart Light Bulb On ASCII Data (From Smartphone)

Once the smart light bulb has received the command from the smartphone to turn on, it issues a response. The binary data for the response can be seen in Figure A-7. Figure A-8 shows the ASCII version of the response data in plain text.



Figure A-7 Turn Smart Light Bulb On Response Binary Data (From Light Bulb)

```

HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Expires: Mon, 1 Aug 2011 09:00:00 GMT
Connection: close
Access-Control-Max-Age: 3600
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE, HEAD
Access-Control-Allow-Headers: Content-Type
Content-type: application/json

[{"success":{"/groups/1/action/on":true}}]

```

Figure A-8 Turn Smart Light Bulb On Response ASCII Data (From Light Bulb)

Figure A-9 contains the data transmitted from the smartphone to the Philips Hue hub to turn the smart light bulbs off. The ASCII version of the data is then shown in Figure A-10. Variable “on” is set to false, which communicates to the lights to turn off.

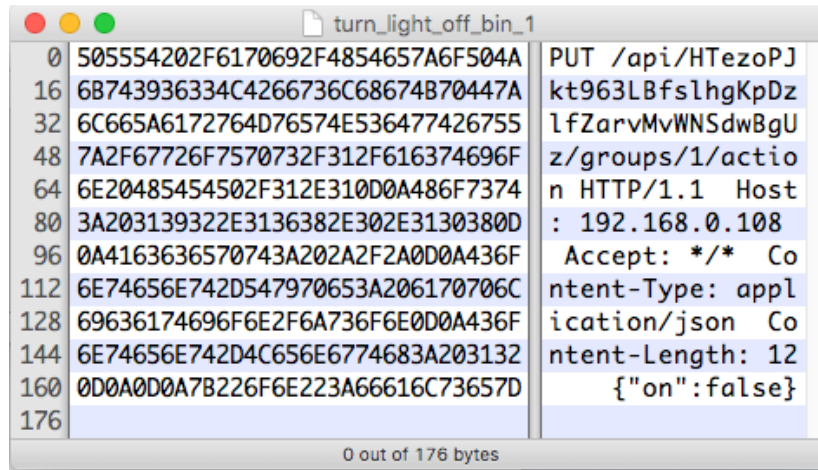


Figure A-9 Turn Smart Light Bulb Off Binary Data (From Smartphone)

```

PUT /api/HTezoPJkt963LBfslhgKpDzlfZarvMvWNSdwBgUz/groups/1/action HTTP/1.1
Host: 192.168.0.108
Accept: /*/*
Content-Type: application/json
Content-Length: 12

{"on":false}

```

Figure A-10 Turn Smart Light Bulb Off ASCII Data (From Smartphone)

Figure A-11 contains the binary response data from the Philips Hue hub after the smart light bulbs have been turned off. Figure A-12 shows the ASCII version of the same data in plain text. It shows the action that was performed on group 1 and communicates that it was successful.



Figure A-11 Turn Smart Light Bulb Off Response Binary Data (From Philips Hue Hub)

```

HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Expires: Mon, 1 Aug 2011 09:00:00 GMT
Connection: close
Access-Control-Max-Age: 3600
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE, HEAD
Access-Control-Allow-Headers: Content-Type
Content-type: application/json

[{"success":{"/groups/1/action/on":false}}]

```

Figure A-12 Turn Smart Light Bulb Off Response ASCII Data (From Light Bulb)

A.3 Sengled Element Classic Smart Light Bulb Recorded Data

This section contains recorded data from the analysis of the Sengled Element Classic smart light bulbs. Since the Sengled Element Classic encrypts transmitted data, only examples of the binary data are shown.

Figure A-13 shows an example of the data transmitted between a smartphone and the Sengled server. As is shown in the screenshot from Wireshark, the detected protocol is TLSv1.2, meaning that all transmitted data is encrypted.

No.	Time	Source	Destination	Protocol	Length
14527	92.542265	us.cloud.sengled.com	192.168.0.100	TCP	170
14541	92.562884	us.cloud.sengled.com	192.168.0.100	TCP	170
14803	94.343669	192.168.0.100	us.cloud.sengled.com	TLSv1.2	582
14804	94.343826	192.168.0.100	us.cloud.sengled.com	TLSv1.2	293
14819	94.429810	us.cloud.sengled.com	192.168.0.100	TCP	170

Figure A-13 Communication between Smartphone and Sengled Server

The next two screenshots (figure A-14 and figure A-15) contain binary data transmitted from the server to the Sengled Element Classic hub to turn the smart light bulbs off. Though the commands were the same, different binary data was transmitted each time suggesting that the TCP packets are encrypted. It can be seen that the binary data in figure A-14 and figure A-15 are different.

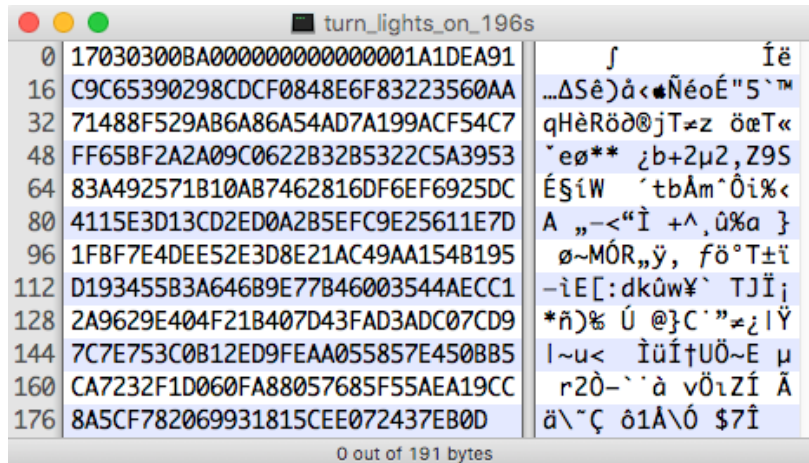


Figure A-14 Turn Smart Light Bulb On Binary Data (Example 1)

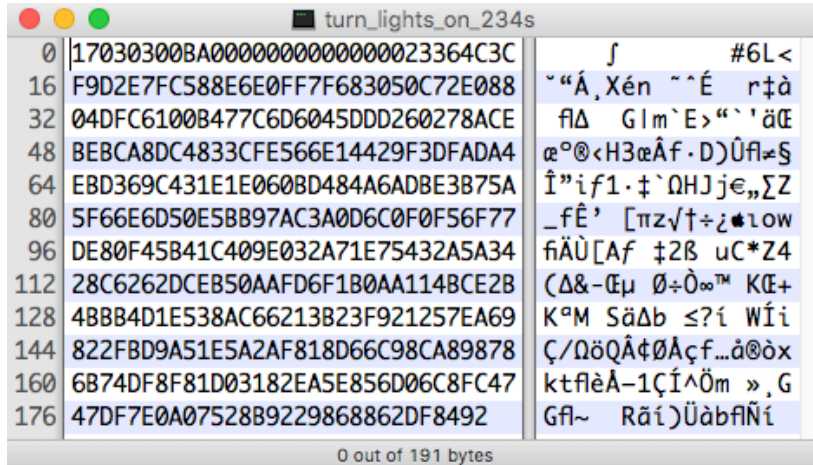


Figure A-15 Turn Smart Light Bulb On Binary Data (Example 2)

The next two figures contain screenshots of the transmitted binary data from the Sengled server to turn the smart light bulbs off. Once again, the transmitted data is different each time, which is consistent with the Sengled Element Classic encrypting transmitted data.

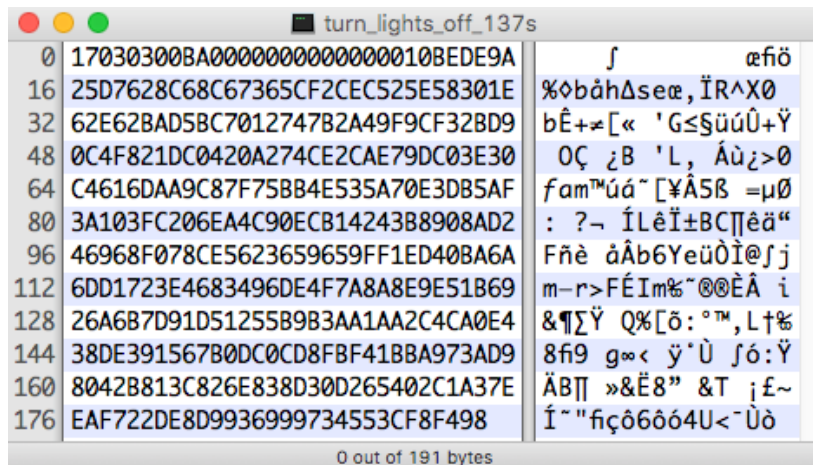


Figure A-16 Turn Smart Light Bulb Off Binary Data (Example 1)

```

turn_lights_off_182s
0 | 17030300BA0000000000000017B82FD1 | f | | / -
16 | CF0698EACB0329544BF599E6AE8150BC | æ ò í À ) TK i ó Ê Æ Á P °
32 | 57D0F24E2E82A63B95CC388BAF163E1A | W - Ú N . Ç ¶ ; ï Ã 8 ä Ø >
48 | FAA5BA723820531368619D6F0D20282E | ` • f r 8 S ha ù o ( .
64 | 0767D066F07B359DC05F6BF668E2181C | g - f * { 5 ù ¿ _ k ^ h ,
80 | 0348D6B42275DF584ED42619190373C1 | K ÷ ¥ " u f l X N ' & s j
96 | 415233128F617F70DAC1217E9C892389 | AR3 è a p / j ! ~ ú â # â
112 | 89F37AF54AF0F40F4774F55306DA5B65 | á Ú z i J * Ù G t i S / [ e
128 | C949E4AA9B02C3C053A822F56872BE7D | ... I % ™ ö √ ¿ S @ " i h r æ }
144 | 2CAD370E242373FD289647899906EE18 | , = 7 $ # s " ( ñ G â ô Ó
160 | F054D64FFE867FF05B10EECDB57FC23C | * T ÷ 0 , Û * [ Ó Ô μ - <
176 | 617ED5518F79D0D7F9DC1588ABDFBE | a ~ ' Q è y - ò ~ < ¶ ' fl æ
0 out of 191 bytes

```

Figure A-17 Turn Smart Light Bulb Off Binary Data (Example 2)

A.4 Belkin WeMo Smart Plug Recorded Data

This section contains the recorded data that was captured when analyzing the Belkin WeMo smart plug. As mentioned in the main text, transmitted data in the Belkin WeMo system can be viewed in plain text so both the binary data and the ASCII version of the data shown. Figure A-18 shows an example of the binary data that is transmitted by a smartphone in order to turn the Belkin WeMo smart plug on.

```

0 | 504F5354202F75706E702F636F6E7472 | POST /upnp/contr
16 | 6F6C2F62617369636576656E74312048 | ol/basicEvent1 H
32 | 5454502F312E310D0A436F6E74656E74 | TTP/1.1 Content
48 | 2D547970653A20746578742F786D6C3B | -Type: text/xml;
64 | 20636861727365743D227574662D3822 | charset="utf-8"
80 | 0D0A534F4150414354494F4E3A202275 | SOAPACTION: "u
96 | 726E3A42656C6B696E3A736572766963 | rn:Belkin:servic
112 | 653A62617369636576656E743A312353 | e:basicEvent:1#S
128 | 657442696E6172795374617465220D0A | etBinaryState"
144 | 436F6E74656E742D4C656E6774683A20 | Content-Length:
160 | 3338330D0A484F53543A203139322E31 | 383 HOST: 192.1
176 | 36382E302E3130353A34393135330D0A | 68.0.105:49153
192 | 557365722D4167656E743A2043796265 | User-Agent: Cybe
208 | 724761726167652D485454502F312E30 | rGarage-HTTP/1.0
224 | 0D0A0D0A3C3F786D6C2076657273696F | <?xml versio
240 | 6E3D22312E302220656E636F64696E67 | n="1.0" encoding
256 | 3D227574662D38223F3E0A3C733A456E | ="utf-8"?> <s:En
272 | 76656C6F706520786D6C6E733A733D22 | velope xmlns:s="
288 | 687474703A2F2F736368656D61732E78 | http://schemas.x
304 | 6D6C736F61702E6F72672F736F61702F | mlsoap.org/soap/
320 | 656E76656C6F70652F2220733A656E63 | envelope/" s:enc
336 | 6F64696E675374796C653D2268747470 | odingStyle="http
352 | 3A2F2F736368656D61732E786D6C736F | ://schemas.xmlso
368 | 61702E6F72672F736F61702F656E636F | ap.org/soap/enco
384 | 64696E672F223E0A203C733A426F6479 | ding/"> <s:Body
400 | 3E0A20203C753A53657442696E617279 | > <u:SetBinary
416 | 537461746520786D6C6E733A753D2275 | State xmlns:u="u
432 | 726E3A42656C6B696E3A736572766963 | rn:Belkin:servic
448 | 653A62617369636576656E743A31223E | e:basicEvent:1">
464 | 0A2020203C42696E6172795374617465 | <BinaryState
480 | 3E313C2F42696E61727953746174653E | >1</BinaryState>
496 | 0A2020203C4475726174696F6E3E3C2F | <Duration></
512 | 4475726174696F6E3E0A2020203C456E | Duration> <En
528 | 64416374696F6E3E3C2F456E64416374 | dAction></EndAct
544 | 696F6E3E0A2020203C55444E3E3C2F55 | ion> <UDN></U
560 | 444E3E0A2020203C2F753A53657442696E | DN> </u:SetBin
576 | 61727953746174653E0A203C2F733A42 | aryState> </s:B
592 | 6F64793E0A3C2F733A456E76656C6F70 | ody> </s:Envelop
608 | 653E0A | e>

```

0 out of 611 bytes

Figure A-18 Turn On Smart Plug Binary Data (From Smartphone)

The following screenshot, figure A-19, contains the ASCII version of the data from figure A-18. It can be seen in figure A-19 that the data transmitted from the smartphone is in an XML SOAP format and has set the “BinaryState” variable to “1” in order to turn the smart plug on.

```
POST /upnp/control/basicevent1 HTTP/1.1
Content-Type: text/xml; charset="utf-8"
SOAPACTION: "urn:Belkin:service:basicevent:1#SetBinaryState"
Content-Length: 383
HOST: 192.168.0.105:49153
User-Agent: CyberGarage-HTTP/1.0

<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>1</BinaryState>
      <Duration></Duration>
      <EndAction></EndAction>
      <UDN></UDN>
    </u:SetBinaryState>
  </s:Body>
</s:Envelope>
```

Figure A-19 Turn On Smart Plug ASCII Data (From Smartphone)

Figure A-20 contains the binary data transmitted from a smartphone in order to turn the smart plug off. The ASCII version of this data is then shown in figure A-21, which is in the XML SOAP format and sets the “BinaryState” variable to “0.”

```

0 504F5354202F75706E702F636F6E7472 POST /upnp/contr
16 6F6C2F62617369636576656E74312048 ol/basicEvent1 H
32 5454502F312E310D0A436F6E74656E74 TTP/1.1 Content
48 2D547970653A20746578742F786D6C3B -Type: text/xml;
64 20636861727365743D227574662D3822 charset="utf-8"
80 0D0A534F4150414354494F4E3A202275 SOAPACTION: "u
96 726E3A42656C6B696E3A736572766963 rn:Belkin:servic
112 653A62617369636576656E743A312353 e:basicEvent:1#S
128 657442696E6172795374617465220D0A etBinaryState"
144 436F6E74656E742D4C656E6774683A20 Content-Length:
160 3338330D0A484F53543A203139322E31 383 HOST: 192.1
176 36382E302E3130353A34393135330D0A 68.0.105:49153
192 557365722D4167656E743A2043796265 User-Agent: Cybe
208 724761726167652D485454502F312E30 rGarage-H|TTP/1.0
224 0D0A0D0A3C3F786D6C2076657273696F <?xml versio
240 6E3D22312E302220656E636F64696E67 n="1.0" encoding
256 3D227574662D38223F3E0A3C733A456E ="utf-8"?> <s:En
272 76656C6F706520786D6C6E733A733D22 velope xmlns:s="
288 687474703A2F2F736368656D61732E78 http://schemas.x
304 6D6C736F61702E6F72672F736F61702F mlsoap.org/soap/
320 656E76656C6F70652F2220733A656E63 envelope/" s:enc
336 6F64696E675374796C653D2268747470 odingStyle="http
352 3A2F2F736368656D61732E786D6C736F ://schemas.xmlso
368 61702E6F72672F736F61702F656E636F ap.org/soap/enco
384 64696E672F223E0A203C733A426F6479 ding/"> <s:Body
400 3E0A20203C753A53657442696E617279 > <u:SetBinary
416 537461746520786D6C6E733A753D2275 State xmlns:u="u
432 726E3A42656C6B696E3A736572766963 rn:Belkin:servic
448 653A62617369636576656E743A31223E e:basicEvent:1">
464 0A2020203C42696E6172795374617465 <BinaryState
480 3E303C2F42696E61727953746174653E >0</BinaryState>
496 0A2020203C4475726174696F6E3E3C2F <Duration></
512 4475726174696F6E3E0A2020203C456E Duration> <EndAct
528 64416374696F6E3E3C2F456E64416374 ion> <UDN></U
544 696F6E3E0A2020203C55444E3E3C2F55 DN> </u:SetBin
560 444E3E0A20203C2F753A53657442696E aryState> </s:B
576 61727953746174653E0A203C2F733A42 ody> </s:Envelop
592 6F64793E0A3C2F733A456E76656C6F70 e>
608 653E0A
  
```

Figure A-20 Turn Off Smart Plug Binary Data (From Smartphone)

```

POST /upnp/control/basicevent1 HTTP/1.1
Content-Type: text/xml; charset="utf-8"
SOAPACTION: "urn:Belkin:service:basicevent:1#SetBinaryState"
Content-Length: 383
HOST: 192.168.0.105:49153
User-Agent: CyberGarage-HTTP/1.0

<?xml version="1.0" encoding="utf-8"?>
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <s:Body>
    <u:SetBinaryState xmlns:u="urn:Belkin:service:basicevent:1">
      <BinaryState>0</BinaryState>
      <Duration></Duration>
      <EndAction></EndAction>
      <UDN></UDN>
    </u:SetBinaryState>
  </s:Body>
</s:Envelope>

```

Figure A-21 Turn Off Smart Plug ASCII Data (From Smartphone)

A.5 Vine WiFi Thermostat Recorded Data

This section contains the recorded data that was collected after analyzing the Vine WiFi thermostat. The Vine WiFi thermostat system uses a server to route communication from a smartphone to the thermostat. Communication between the smartphone and the server is encrypted using the TLSv1.2 protocol. Communication between the smartphone and the xingconnected.com server is shown in figure A-22.

No.	Time	Source	Destination	Protocol	Length
31105	254.269820	xingconnected.com	192.168.0.103	TLSv1.2	717
31109	254.271902	192.168.0.103	xingconnected.com	TCP	150
31110	254.272046	192.168.0.103	xingconnected.com	TCP	154
31119	254.308235	192.168.0.103	xingconnected.com	TLSv1.2	345
31135	254.403087	xingconnected.com	192.168.0.103	TLSv1.2	205
31139	254.404117	192.168.0.103	xingconnected.com	TCP	154
31143	254.406603	192.168.0.103	xingconnected.com	TLSv1.2	557
31144	254.406690	192.168.0.103	xingconnected.com	TLSv1.2	361

Figure A-22 Screenshot of Communication Between Smartphone and Vine Server

Prior to performing a firmware update, the Vine server transmitted data in plain text. Data from before and after the firmware update are shown in this section. Figure A-23 shows the binary data transmitted from the server to turn the WiFi thermostat on and Figure A-24 shows that same data in ASCII format. In the transmitted data, it can be seen that the server set the “power” variable to “1.” There is also a packet counter and timestamp.

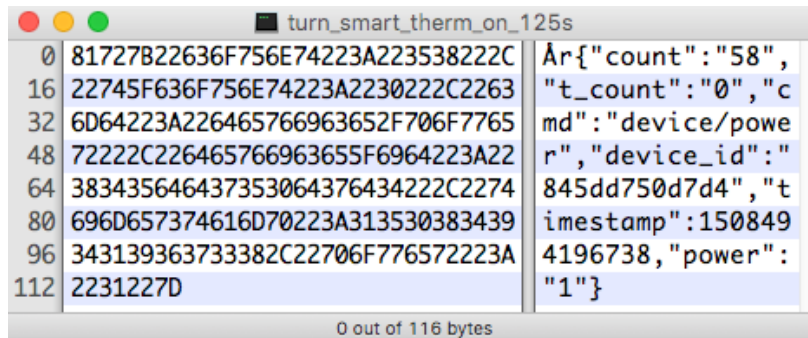


Figure A-23 Turn WiFi Thermostat On Binary Data (Before Firmware Update)

```
{ "count": "58", "t_count": "0", "cmd": "device/power", "device_id": "845dd750d7d4",
  "timestamp": 1508494196738, "power": "1" }
```

Figure A-24 Turn WiFi Thermostat On ASCII Data (Before Firmware Update)

Figure A-25 shows the binary data transmitted from the server to the Vine WiFi thermostat in order to turn it off. Figure A-26 then shows the ASCII version of that data.

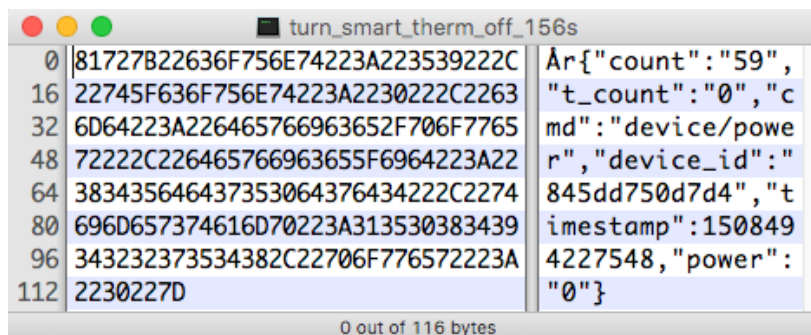


Figure A-25 Turn WiFi Thermostat Off Binary Data (Before Firmware Update)

```
{ "count": "59", "t_count": "0", "cmd": "device/power", "device_id": "845dd750d7d4",
  "timestamp": 1508494227548, "power": "0" }
```

Figure A-26 Turn WiFi Thermostat Off ASCII Data (Before Firmware Update)

The following two figures contain the data transmitted from the server in order to change the weekly schedule. This data was recorded prior to updating the firmware on the Vine WiFi thermostat. Figure A-27 shows the binary data transmitted from the server to the WiFi thermostat and figure A-28 shows the ASCII version of the same data. As mentioned in the main text (and shown in figure A-28) the weekly schedule can be viewed in plain text. The variables “item1” through “item7” contain the days of the week, where item1 is Monday. Variable “t” contains the temperature in degrees Fahrenheit and “h” contains the time of day in minutes. So for example, if “t” is equal to 360, then the temperature change would take effect at 6 am. If this data were to be intercepted, an attacker would know the entire thermostat weekly schedule.


```

smart_therm_update_sched_14s
0 817E04F37B22636F756E74223A223138 31222C22745F636F756E74223A223022 A~ U{"count": "181", "t_count": "0"
32 2C22636D64223A226465766963652F73 65745F6D6F64656C5F696E666F222C22 , "cmd": "device/set_model_info",
64 6465766963655F6964223A2238343564 6437353064376434222C2274696D6573 device_id": "845dd750d7d4", "times
96 74616D70223A31353038363038373136 3130342C226D6F6465223A2231222C22 tamp": "1508608716104", "mode": "1",
128 6C696D6974223A2236302D3835222C22 6E616D65223A2253756D6D65722D3031 limit": "60-85", "name": "Summer-01
160 222C227374617465223A312C226D6F64 656C5F6964223A3139353539322C2264 ", "state": "1", "model_id": "195592", "d
192 617461223A7B22756E6974223A224622 2C226974656D7331223A5B7B2263223A ata": {"unit": "F", "items1": [{"c":
224 2230222C2274223A223835222C226822 3A2230227D2C7B2263223A2230222C22 "0", "t": "85", "h": "0"}, {"c": "0",
256 74223A223738222C2268223A22333630 227D2C7B2263223A2230222C2274223A t": "78", "h": "360"}, {"c": "0", "t":
288 223835222C2268223A22343830227D2C 7B2263223A2230222C2274223A223738 "85", "h": "480"}, {"c": "0", "t": "78
320 222C2268223A2231303230227D2C7B22 63223A2230222C2274223A223835222C ", "h": "1020"}, {"c": "0", "t": "85",
384 2274223A223835222C2268223A223022 7D2C7B2263223A2230222C2274223A22 "h": "1320"}], "items2": [{"c": "0",
416 3738222C2268223A22333630227D2C7B 2263223A2230222C2274223A22383522 "t": "85", "h": "0"}, {"c": "0", "t":
448 2C2268223A22343830227D2C7B226322 3A2230222C2274223A223738222C2268 "78", "h": "360"}, {"c": "0", "t": "85"
480 223A2231303230227D2C7B2263223A22 30222C2274223A223835222C2268223A ", "h": "480"}, {"c": "0", "t": "78", "h
512 2231333230227D5D2C226974656D7333 223A5B7B2263223A2230222C2274223A "1020"}, {"c": "0", "t": "85", "h":
544 223835222C2268223A2230227D2C7B22 63223A2230222C2274223A223738222C "1320"}], "items3": [{"c": "0", "t":
576 2268223A22333630227D2C7B2263223A 2230222C2274223A223835222C226822 "85", "h": "0"}, {"c": "0", "t": "78",
608 3A22343830227D2C7B2263223A223022 2C2274223A223738222C2268223A2231 ", "h": "360"}, {"c": "0", "t": "85", "h
640 303230227D2C7B2263223A2230222C22 74223A223835222C2268223A22313332 ": "480"}, {"c": "0", "t": "78", "h": "1
672 30227D5D2C226974656D7334223A5B7B 2263223A2230222C2274223A22383522 "020"}, {"c": "0", "t": "85", "h": "132
704 2C2268223A2230227D2C7B2263223A22 30222C2274223A223738222C2268223A ", "items4": [{"c": "0", "t": "85",
736 22333630227D2C7B2263223A2230222C 2274223A223835222C2268223A223438 ", "h": "0"}, {"c": "0", "t": "78", "h":
768 30227D2C7B2263223A2230222C227422 3A223738222C2268223A223130323022 "360"}, {"c": "0", "t": "85", "h": "48
800 7D2C7B2263223A2230222C2274223A22 3835222C2268223A2231333230227D5D }, {"c": "0", "t": "85", "h": "1320"}]
832 2C226974656D7335223A5B7B2263223A 2230222C2274223A223835222C226822 ", "items5": [{"c": "0", "t": "85", "h
864 3A2230227D2C7B2263223A2230222C22 74223A223738222C2268223A22333630 ": "0"}, {"c": "0", "t": "78", "h": "360
896 227D2C7B2263223A2230222C2274223A 223835222C2268223A22343830227D2C }, {"c": "0", "t": "85", "h": "480"},
928 7B2263223A2230222C2274223A223738 222C2268223A2231303230227D2C7B22 {"c": "0", "t": "78", "h": "1020"}, {"c
960 63223A2230222C2274223A223835222C 2268223A2231333230227D5D2C226974 "0", "t": "85", "h": "1320"}], "it
992 656D7336223A5B7B2263223A2230222C 2274223A223835222C2268223A223022 ems6": [{"c": "0", "t": "85", "h": "0"
1024 7D2C7B2263223A2230222C2274223A22 3738222C2268223A22343830227D2C7B }, {"c": "0", "t": "78", "h": "480"}, {"
1056 2263223A2230222C2274223A22363022 2C2268223A22383430227D2C7B226322 "c": "0", "t": "60", "h": "840"}, {"c"
1088 3A2230222C2274223A223738222C2268 223A22383535227D2C7B2263223A2230 ": "0", "t": "78", "h": "855"}, {"c": "0
1120 222C2274223A223631222C2268223A22 383730227D2C7B2263223A2230222C22 ", "t": "61", "h": "870"}, {"c": "0", "
1152 74223A223835222C2268223A22313332 30227D5D2C226974656D7337223A5B7B t": "85", "h": "1320"}], "items7": [{"
1184 2263223A2230222C2274223A22383522 2C2268223A2230227D2C7B2263223A22 {"c": "0", "t": "85", "h": "0"}, {"c": "
1216 30222C2274223A223738222C2268223A 22343830227D2C7B2263223A2230222C "0", "t": "78", "h": "480"}, {"c": "0",
1248 2274223A223835222C2268223A223133 3230227D5D7D7D "t": "85", "h": "1320"}]}}
O out of 1271 bytes

```

Figure A-27 Change Weekly Schedule Binary Data (Before Firmware Update)

```

{"count":"181",
  "t_count":"0",
  "cmd":"device/set_model_info",
  "device_id":"845dd750d7d4",
  "timestamp":1508608716104,
  "mode":"1","limit":"60-85",
  "name":"Summer-01",
  "state":1,
  "model_id":195592,
  "data":
  {
    "unit":"F",
    "items1":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items2":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items3":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items4":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items5":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"360"},
      {"c":"0","t":"85","h":"480"},
      {"c":"0","t":"78","h":"1020"},
      {"c":"0","t":"85","h":"1320"}],
    "items6":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"480"},
      {"c":"0","t":"60","h":"840"},
      {"c":"0","t":"78","h":"855"},
      {"c":"0","t":"61","h":"870"},
      {"c":"0","t":"85","h":"1320"}],
    "items7":[
      {"c":"0","t":"85","h":"0"},
      {"c":"0","t":"78","h":"480"},
      {"c":"0","t":"85","h":"1320"}]
  }
}

```

Figure A-28 Change Weekly Schedule ASCII Data (Before Firmware Update)

Figures A-29, A-30, and A-31 all contain transmitted data from the server to the Vine WiFi thermostat after the firmware update had been performed. Figure A-29 contains the data when the WiFi thermostat is being turned on, figure A-30 contains the data when the WiFi thermostat is being turned off, and figure A-31 contains the data when the weekly schedule is being updated. In all three figure it can be seen that there is a “k” variable that contains a number and then a “v” variable that contains encrypted data.

```
{ "k": "97", "v": "4BEOEEWaDiPnoP6qf4YI0U3yUyhxWvxGSO+0lR8+rhNfoTVlQFevcWkr8K3y
pMtSpIXC895ja/g+HnbxNp/JdFHx8x2Z6VOOtXvaVfUBW4DE5XweQ+uyGfAmbR4ngvuNqKub4sI
iW2AKXtHcwNaphU30YQ==" }
```

Figure A-29 Turn WiFi Thermostat On ASCII Data (After Firmware Update)

```
{ "k": "77", "v": "QERh2U6d+vs6thoKhMMZUfiibWcovFvNK1341UBfU6zAaTROOEnlSuwQNYRY
HNhnfH7V3Z69dWXvZSa/VxImy+oEfGiceObZtxOpCb4uW08+cZW10Lc4HDvZVDWYybnud5fVgCr
BhQ4qzBNfyOo+bxgLRA==" }
```

Figure A-30 Turn WiFi Thermostat On ASCII Data (After Firmware Update)

```
Å~] { "k": "09", "v": "mz1SBThLbCJlq5HHWey3J9D/3pgWBKbc93o+M8gFEUaU1TdBlge83EO5L
RI4uM1kaSlI1lsElrjX/5EYVRl4x6+bgrXpJap+iotdWdLyvtDDRca7rtYZK6Ev15bkXGxYV09X
7lc8Br522/dvKA6RKXSBkQkF9zqWQBoAouD1PBQiu07Y7Ng3goyfauCTORVNWbbZbsaJQkR4KeX
socEM81CoXMjukPelPQpmlsnLtV33YGfEvmozvknCTkQHJYyk/Wqf8OhrwEwfcveZKRFxVxwnbI
+RWXowIV9d5i0abxn72e8x4RULXTx/fjzQ+lhAHIPtqF6n4azfRiK5RcYGjz6c4iS0YR6zQjmV8
5cWi0sLYLv1OhdaQxcIshigfSYNlrEwM4M41qR5UnNGhx5HybZXFvHIx9zfNuf6tLucmAYgICJX
4ui943PL51TDsnpbBIB72Dvfb6DObQyuhbYAjElX4kKuIeHt9CMG0xDwulFyspxbu7Sehqtxxu5
Ywe+OT6OGVv537oQXeu0wSkphGeCdKbXqWUWlOs6wCrEofrawAuQ34sblKbaDvuaTK04A+nuBKS
nRwx2p633iRBTmEnO5gs0cOn7vPt0HXJTJbOzUC6ilbk+2Pyrwy8hR344ogAmudEvSUiHk/+p3E
Nlv2ROXQH8rkcwKyahg9duC+0ylnblyV53JWGUW5a4V1CPV+irV8D3ASoeIKTeRtZfJmJxpamVe
KDZsVbjjBqrvhmsGxJBUYuxeXthb3nzj+X29Tgsehq6G14nFh+d8gXouy1gAzjvZXdT56cki06
Za3lRBRgOIQcud9BWCpvrE7qh7lRMvagEJRUTsN90/yiqJsCKkr7b3Ufu9AYewglW+E76f8Wf01
v41IuF4cXBDXmjCJfulVURFL3DyVcUDC79k8sEwp66qMUFQAPZLgzlXTT84K2UYMPYrMn893ne
rIz1t1YzfsJPTMo8Ymw05Q0JtNhs5mTBT4EGO5PPk0uz4kCIh9YscbKFFqVFpubA8XtpQQq4Jt
/vJaXvvGPNjVeCca069qdyPElCD1YQFK66LcVvYyJXbDm2mDjtmIMApz4tvuIqwnqkppzUuYSM6
5EqIchTdRWI51Ci8+ix1kl8vQtYURqr5mmg4itkIDYDVBnmofMUfMrX+/t21Dr1I4d2FuyM0AzQ
ie+w+3xazw94qbdnYs2q9mpWQgF5anvqjn4ep7OgKYjSx9hxfGtxHq7St4Igf88L1rfNhnyc5hP
8LhCEqyf+qvFpUwGObG39hWJZY31NN2y5lT8b/utUz2AZ5uEz/xvjtY4rEeCsH9VSTBePdGxknn
W11r9PrgKtCACTKeUFfF3ZggkNVMagTPUW5JiXKiWkIh8jQfSsqLN1/747tftHw87VzJE8egbhs
HCOBo5p4cAhIc/8I6TDuAwyAXK
```

Figure A-31: Change Weekly Schedule ASCII Data (After Firmware Update)

A.6 Sensi WiFi Thermostat Recorded Data

This section contains the recorded data that was collected when analyzing the Sensi WiFi thermostat. As discussed in the main text, the communication between the smartphone and the Sensi server is encrypted using TLSv1.2. This is shown in figure A-32.

No.	Time	Source	Destination	Protocol	Length
13371	104.6777...	192.168.0.101	rt.sensiapi.io	TLSv1.2	266
13442	105.0118...	192.168.0.101	rt.sensiapi.io	TCP	154
13446	105.0122...	192.168.0.101	rt.sensiapi.io	TCP	154
13452	105.0157...	192.168.0.101	rt.sensiapi.io	TLSv1.2	244
13482	105.1213...	192.168.0.101	rt.sensiapi.io	TCP	154
13488	105.1236...	192.168.0.101	rt.sensiapi.io	TCP	154

Figure A-32 Communication Between Smartphone and Sensi Server

Figures A-33 and A-34 show examples of the binary data used to turn the thermostat to Auto mode, which is transmitted from the server to the Sensi WiFi thermostat. The TCP packets transmitted from the server are encrypted. It can be seen in the screenshots below that the data for each example is different.

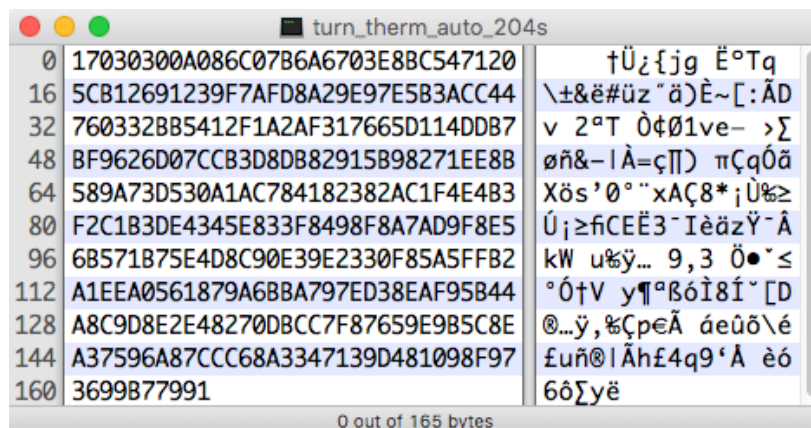


Figure A-33 Turn WiFi Thermostat to Auto Mode (Example 1)

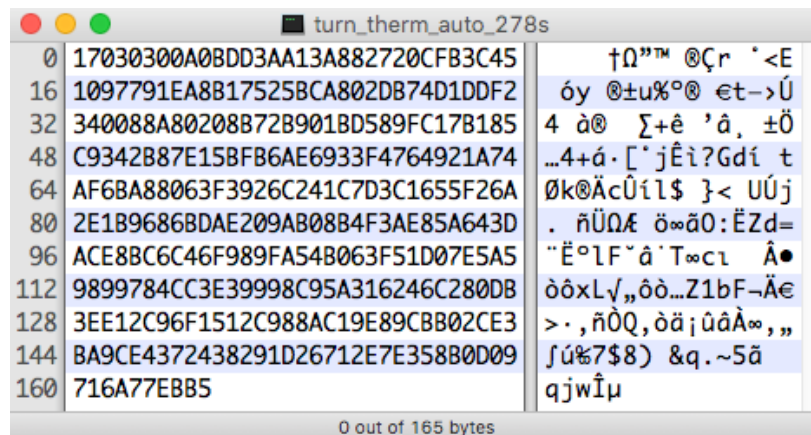


Figure A-34 Turn WiFi Thermostat to Auto Mode (Example 2)

Figure A-35 and figure A-36 contain screenshots of the binary data transmitted from the server to the Sensi WiFi thermostat to turn it off. In each instance, the binary data is encrypted.

```

turn_therm_off_186s
0 17030300A0196324FF148498E97C8F39      † c$` ÑòÈ!è9
16 155E796D684CD76F4D4175E584211F10     ^ymhLóoMAuÂÑ!
32 E7D56AB90B4E7F8F80C293EAF068F79C     Á'jπ N èÄ-ìÍ#h`ú
48 08F36BF27418C8D7F54E38CEE84F09CB     ÛkÚt »óιN8ÈËO À
64 71F3968832AC2C8B2D5E205101086E87     qÛñã2",ã-^ Q ná
80 F8EA279EB43B118F382A8946C070BA25     -Í'ú¥; è8*âF¿pJ%
96 A142B2767414A3C388EB84BEC7D8C74D     °B≤vt f√àÎÑæ«ÿ«M
112 8980888BA628B9F9445231531346FC91     áÄäã¶(π`DR1S F,ë
128 48BBE04E8E66CAB39D00BF985B1AA8A0     H°‡Néf ≥ù øò[ @†
144 BC4FB62507E9709656F6A6498C52C7C8     °Oð% ÈpñV`¶IáR«»
160 BE2680D31F                             æ&Ä"
0 out of 165 bytes

```

Figure A-35 Turn WiFi Thermostat to Off Mode (Example 1)

```

turn_therm_off_220s
0 17030300A01B7815589B824FCDABAE59      † x XðÇOÖ`ÆY
16 DA94F5BE301D695C9FEA6C6D53F5D196     /iιæθ i\üÍlmSι-ñ
32 1EC5E1B8BEC488EEA9F7FF84B475602A     ≈.fæfàÓ@~`Ñ¥u`*
48 385FBBD93BE15C0DBF8AC5B83F80546E     8_°ÿ;.\ øä≈[]?ÄTn
64 294728C4BA6AAC20CC7F92FBA6A732DC     )G(fsj" Ä í'¶B2<
80 7D021530698FAD5A9913C37D577490B1     } Øiè×Zò √}Wtê±
96 F2453B81D39791C7924183D3B0C3FFF5     ÚE;Ä"óë«íAÉ"∞√`ι
112 038ABCCC4C9718639BCA94EE865C7148     ä°ÄLó cõ iÓÜ\qH
128 E730ADB285F88201B7AA7E65C947BA66     Á0×≤Ö`Ç Σ™~e...Gjf
144 E174927A7A5C802DA6F89407B9481F42     ·tízz\Ä-¶`î πH B
160 582A450140                             X*E @
0 out of 165 bytes

```

Figure A-36 Turn WiFi Thermostat to Off Mode (Example 2)

A.7 Amazon Echo Dot Recorded Data

This section contains screenshots of examples of the Amazon Echo Dot communicating with the servers. Figure A-37 shows the server transmitting data to the Amazon Echo Dot and figure A-38 shows the Amazon Echo Dot transmitting data to the server. In both cases, TLSv1.2 is used to encrypt the communication session so that it cannot be subject to a man in the middle attack.

No.	Time	▲	Source	Destination	Protocol	Length
7574	47.089694		Android.local	dp-gw-na.amazon.com	TLSv1.2	344
7575	47.089890		Android.local	pindorama.amazon.com	TCP	1598
7576	47.090020		Android.local	pindorama.amazon.com	TLSv1.2	1080
7582	47.146048		dp-gw-na.amazon.com	Android.local	TLSv1.2	148
7584	47.146245		dp-gw-na.amazon.com	Android.local	TLSv1.2	187

Figure A-37 Amazon Echo Dot Receiving Data Example

No.	Time	▲	Source	Destination	Protocol	Length
7642	47.292882		Android.local	224.0.0.22	IGMPv3	142
7645	47.293172		Android.local	pindorama.amazon.com	TLSv1.2	265
7646	47.293317		Android.local	pindorama.amazon.com	TLSv1.2	1017
7651	47.311938		Android.local	pindorama.amazon.com	TLSv1.2	584

Figure A-38 Amazon Echo Dot Transmitting Data Example

Bibliography

- [1] M. G. Samaila, M. Neto, D. A. B. Fernandes, M. M. Freire, and P. R. M. Inácio, “Security Challenges of the Internet of Things,” *Internet of Things Beyond the Internet of Things*, pp. 53–82, 2017.
- [2] P. Schaumont, “Security in the Internet of Things: A challenge of scale,” *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017.
- [3] B. Copos, K. Levitt, M. Bishop, and J. Rowe, “Is Anybody Home? Inferring Activity From Smart Home Network Traffic,” *2016 IEEE Security and Privacy Workshops (SPW)*, 2016.
- [4] M. Moody and A. Hunter, “Exploiting known vulnerabilities of a smart thermostat,” *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016.
- [5] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, “An experimental study of security and privacy risks with emerging household appliances,” *Workshop on Security and Privacy in Machine-to-Machine Communications (M2MSec'14)*, 2014.
- [6] M. Morgenstern, “Hue! Let there be light!,” *AV-TEST Internet of Things Security Testing Blog*, 12-Jul-2017. [Online]. Available: <https://www.iot-tests.org/2017/06/hue-let-there-be-light/>.
- [7] N. Dhanjani, “Hacking Lightbulbs: Security Evaluation of the Philips Hue Personal Wireless Lighting System,” *Internet of Things Security Evaluation Series*, 2013.
- [8] T. J. Seppala, “Hackers hijack Philips Hue lights with a drone,” *Engadget*, 03-Nov-2016. [Online]. Available: <https://www.engadget.com/2016/11/03/hackers-hijack-a-philips-hue-lights-with-a-drone/>.
- [9] O. Pursche, “Testing Amazon Echo Dot & Alexa App,” *AV-TEST Internet of Things Security Testing Blog*, 31-Mar-2017. [Online]. Available: <https://www.iot-tests.org/2017/02/testing-amazon-echo-dot-alexa-app/>.
- [10] R. Moskowitz, “Weakness in Passphrase Choice in WPA Interface,” *WNN Wi-Fi Net News*, 04-Nov-2003. [Online]. Available: https://wifinetnews.com/archives/2003/11/weakness_in_passphrase_choice_in_wpa_interface.html.
- [11] A. H. Lashkari, M. Mansoor, and A. S. Danesh, “Wired Equivalent Privacy (WEP) versus Wi-Fi Protected Access (WPA),” *2009 International Conference on Signal Processing Systems*, 2009.

- [12] M. Vanhoef and F. Piessens, "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2," *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS 17*, 2017.
- [13] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, Kasper Emilia, S. Cohny, S. Engels, C. Paar, and Y. Shavitt, "DROWN: Breaking TLS using SSLv2," *Proceedings of the 25th USENIX Security Symposium*, Aug. 2016.
- [14] "SSL Pulse," *Qualys SSL Labs - SSL Pulse*. [Online]. Available: <https://www.ssllabs.com/ssl-pulse/>.
- [15] D. J. Rani and S. E. Roslin, "Light weight cryptographic algorithms for medical internet of things (IoT) - a review," *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*, 2016.
- [16] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing*, May 2017.
- [17] S. A. Salami, J. Baek, K. Salah, and E. Damiani, "Lightweight Encryption for Smart Home," *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 2016.