Sandissa: A Simplified Smart Home Model

Sagindyk (Sandy) Urazayev
Electrical Engineering and Computer Science
University of Kansas
Lawrence, Kansas 66044
Email: ctu@ku.edu

Anissa Khan
Electrical Engineering and Computer Science
University of Kansas
Lawrence, Kansas 66044
Email: anissakhan@ku.edu

***Abstract*—The Internet of Things (IoT) has become more prevalent in our everyday lives through the use of smart home devices. As such, the security of smart home systems has become of paramount importance. In this study, we design and implement a simplified smart home model that is meant to serve as a testbed for a wide range of smart home systems. Once implemented we perform a security analysis of the whole system and put security controls in place. In doing so, we demonstrate ways in which the general IoT smart home system can be secured.**

***Keywords*—** Smart Home, Internet of Things (IoT), Testbed, Security, Abstract Model

1. Introduction

The Internet of Things (IoT) is quickly becoming an important part of our everyday lives. End-user spending on IoT devices is forecast to grow to 594 billion US dollars by the end of 2022 and the more specific smart home IoT market is projected to reach 53.45 billion U.S. dollars in size by 2022 [1], [2]. The more prevalent IoT is in our lives, the more information we share with IoT systems. The more information we share, the more important it is to be capable of analyzing system security and implementing effective security measures.

In order to perform such security analyses, we designed and implemented an abstract model that reflects real world IoT applications. Specifically, our model is designed to represent an IoT system deployed in a user's home. These systems are often referred to as smart home systems. Smart home systems consist of a set of heterogeneous devices that perform different functions throughout the home. The goal of such devices is to provide support to the homeowner by making certain tasks more efficient and streamlined. Examples of such devices include 'smart' versions of: alarm systems, security cameras, door locks, doorbells, light bulbs, fridges, and home assistants among many others.[1] Each of these devices has a different set of capabilities through which the user can interact.

In our system we simplify the smart home model to broaden the applicability of our security evaluation. In our model, IoT devices in the home fall into two categories. Each category has a single, general capability. The first category consists of devices that are controlled by the user. We will refer to these as controllable devices. The second category consists of devices that sense information about the user or the user's environment. We will refer to these as sensing devices. Each smart home device discussed earlier may fall into one or both of these device categories.

For instance, a door lock would be considered a controllable device as the user is able to control the device by performing actions: lock and unlock the door. A security camera on the other hand would be considered a sensing device as it is monitoring the environment and transmitting that information to the user. A home assistant could serve as both a controllable device and a sensing device as it could add items to a user's grocery list and also listen for sounds of breaking glass when a user is not home. By simplifying our device categories we are able to focus on the specific security concerns facing each of these general capabilities and in turn apply solutions to these concerns to a wide range of smart home devices.

The rest of this paper is organized as follows: In Section 2, we present the layout and components of our simplified smart home system. In Section 3, we introduce the threat model that we used to guide the security measures we put in place. Section 4 delves into the specific security controls we implemented. Section 5 discusses attack scenarios as they relate to the attacker's goals and how our security controls prevent such attacks. Section 6 discusses limitations and potential expansions that could be made to the system. In Section 7 related work is reviewed, and finally in Section 8 we conclude.

2. System Overview

The implementation of our simplified smart home model consists of two sensors that are powered by our IoT device. The device is a Raspberry Pi and the sensors are a temperature sensor and an LED bulb. The LED bulb is a controllable device representing devices that can be controlled; the temperature sensor is a sensing device representing devices whose main function is to sense their surroundings. Our system connections can be seen in Figure 1.

The device, however, is only a minor (albeit important) portion of the IoT system we have built. In order to accurately perform a complete security analysis, we must implement a more complete system architecture. This architecture can be seen in Figure 2. This model serves as a testbed for already existing IoT applications. The main components of the architecture are: a User, Webpage, Server, Database, MQTT Broker, and IoT devices.

2.1. System Components

The primary stakeholder in the system is the user. The user owns the IoT device and onboards the IoT device onto the platform.

1. Examples of such devices are as follows: Ooma alarm systems, Google Nest security cameras, Schlage smart door locks, Ring doorbells, Philips Hue light bulbs, Samsung Smart fridges, Amazon Echo home assistants

The user's expectations of the system include having control over the IoT device and having access to all of the device's collected data.

The webpage is the user's main access point to the system's capabilities. It provides a friendly interface to the user that separates them from the non-user-friendly complexities of the backend implementation. It has a login interface that prevents unregistered users from accessing the dashboard. Once logged in, the dashboard displays the information collected by the temperature sensor and provides on/off buttons that control the LED bulb. The dashboard can be seen in Figure 3.

The server is the backbone of the backend. It is the intermediary between the frontend webpage and the backend data sensing layer. It also retrieves requested data from the database on demand.

The database is where authorized user credentials and historical temperature readings are stored. The database's priority is to provide reliable, secure storage of the data.

The MQTT broker relays messages between the server and the IoT devices. We implement MQTT protocol using Mosquitto.[2]

Finally, the IoT devices are as described earlier.

2.2. System Component Interactions

The user interacts directly with the webpage via connection A (Figure 2). Connection B connects the webpage to the server. Data is sent across B via the REST protocol. The server is connected to the database via connection C and to the MQTT broker via connection D. In our implementation, the database and MQTT broker both live on the server; however for demonstrational purposes we treat them as separate entities. Connection C uses TCP. Over connection D, the server is an MQTT subscriber and publisher. It subscribes to the Temperature topic and publishes control commands to the LED topic via the MQTT protocol. The IoT devices are connected to the MQTT broker via connections E. In our implementation, we have only one device; however in a broader application we can imagine the deployment of multiple devices in this IoT system. Similar to the server, the IoT devices are also MQTT subscribers and publishers; they subscribe to the LED topic to listen for control commands and publish temperatures to the Temperature topic every second. Both are done via the MQTT protocol.

2. Official MQTT broker implementation by Eclipse Foundation, https://mosquitto.org/

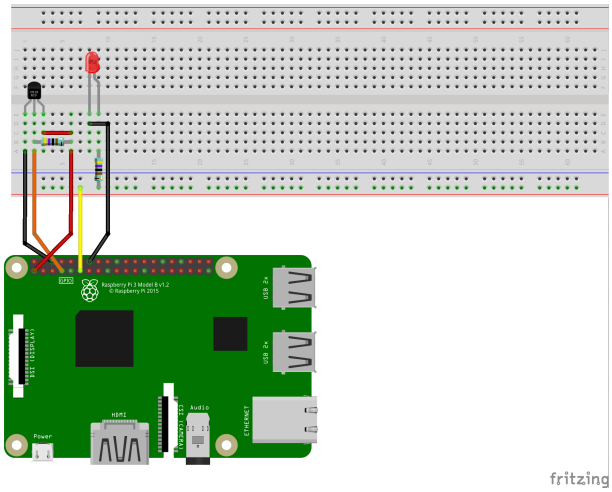

Figure 1: Physical Connections

3. Threat Model

IoT devices are susceptible to a range of attack scenarios. Recall that our simplified smart home model consists of two categories of devices—controllable and sensing devices. The adversary's intention in our model is four-fold. The attacker's goal could be to:

1) gain access to the current state of the controllable device
2) send spoofed control commands to the controllable device
3) gain access to sensitive information collected by the sensing device
4) spoof data reported by the sensing device.

In our scenario this includes knowing whether or not the LED light is on, controlling the LED light, knowing the temperature readings, or spoofing the readings reported by the temperature sensor. Though seemingly innocuous, as discussed earlier our model is a simple representation of more complex IoT devices.

For instance, if the controllable device is a user's smart front door lock, this device controls physical access to a user's home. If the attacker is able to remotely gain access to the current state of the door lock (whether or not the door is unlocked), the attacker could gain physical access to the user's home or determine door locking behaviors of a user (does the user leave their door unlocked at specific times during the day). If the attacker is able to send spoofed control commands to the door lock, the attacker can gain physical access to the user's home or grant another malicious entity access to a user's home.

If the sensing device is a smart glucose monitor [3], this device would collect continuous sensitive health related information. If the attacker is able to gain access to this information, they could report this sensitive information to malicious parties or they could be able to physically take advantage of the user when they know they are in a weak state of health. If the attacker is able to spoof the data reported by the glucose monitor to the user, this could put the user in immediate danger if that user is diabetic.

The information gained by an attacker about each of these devices can be combined with other information about a user to learn about a user's behavior. We assume that direct access to the physical IoT devices is outside the scope of our analysis. In each example provided above, we assume the attacker carries out each attack remotely rather than physically manipulating the door lock or glucose monitor hardware directly. Therefore we do not implement hardware-based security measures.

4. System Security

The primary security goals of our system include confidentiality and integrity. As observed previously, it is important for sensitive information to remain confidential so that an attacker cannot use such information to their advantage. Integrity is vital especially when a sensor reports health related data or data related to the security of a user's home. We don't want an attacker to be able to change data reported by sensing devices or sent to controllable devices.

The security measures we put in place are aimed at achieving our security goals and at preventing the aforementioned attack scenarios discussed in Section 3. In addition to other mechanisms, the majority of our connections are secured using SSL/TLS which ensures both confidentiality and integrity.

4.1. Connection A

Referring back to Figure 2's connection mapping, connection A is the physical connection between the user and the webpage

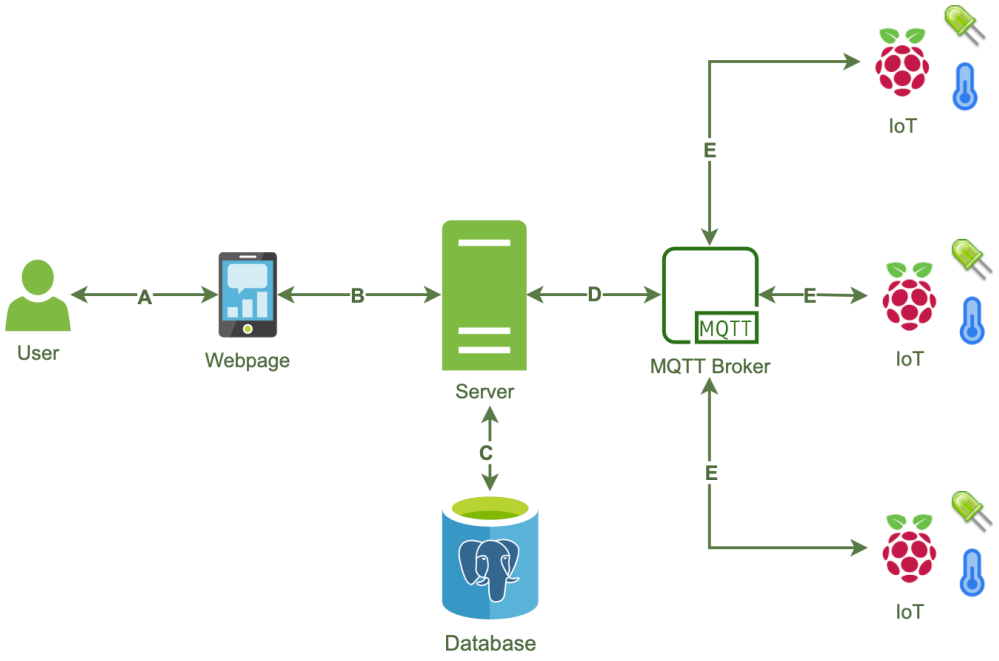

Figure 2: System Architecture

as discussed earlier. This connection is secured by password based authentication. When a user enters their credentials, the webpage forms an authentication request and sends it to the server. The server then verifies against the database whether or not the user exists and is authorized. Only authorized users are able to access the webpage's dashboard. The process of registering new users to access the webpage's dashboard is outside of the scope of our system. All authorized user information is stored in the database in a hashed format.

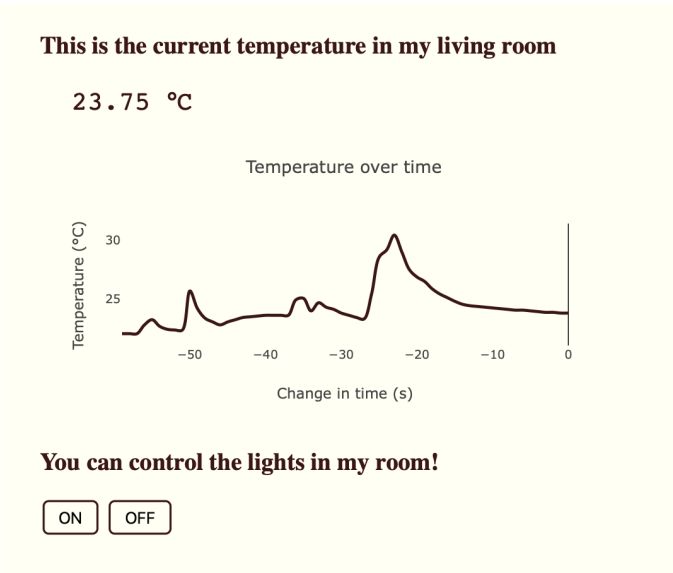

Figure 3: User-facing interface

With respect to user credentials, we follow standard security protocols regarding credential storage: all passwords are stored in the database as hash digests using SHA512. Whenever a user makes a login attempt, the entered password is hashed and verified against the corresponding hash digests stored in the database. If the combined username and password hash match an entry in the database, the authentication process is successful and the user is granted access to the dashboard. By using a strong hashing algorithm, the risks of hash collision and false positive login attempts are mitigated making it unlikely for an attacker to successfully brute force the login credentials.

4.2. Connection B

Connection B between the webpage and the server uses the REST protocol as discussed earlier. REST is secured via SSL/TLS. The server certificate we generated is signed by Let's Encrypt, which is "a free, automated, and open certificate authority" [4]. This connection is further secured via a firewall on the server, which we set up using the iptables[3] program on Linux. The firewall has all ports for incoming connections closed except for port 3737 which is the port we use for REST communication. Additionally, our server implements authentication standard RFC 7617. RFC 7617 is the 'Basic' HTTP Authentication Scheme and is not considered to be a secure method of user authentication unless it is used in combination with a secure mode of communication such as TLS, which we do use in this application [5]. The server further uses CORS (Cross Origin Resource Sharing) which is an HTTP based header mechanism that allows the server to specify the origins from which a browser is permitted to load resources [6]. In our implementation, the server only allows REST requests of type GET and POST that originate from https://sandyuraz.com.

3. https://linux.die.net/man/8/iptables

4.3. Connection C

Connection C between the server and the database is a TCP connection that also uses SSL/TLS connection where the database is the certificate authority (CA). The database also has a firewall, which prevents any requests from being made to any ports besides port 5432. It accepts authorized connections on global path elephant.sandyuraz.com. Furthermore, the database rejects remote logins to its superuser account. This prevents anyone with remote access to the database from logging in with excessive privileges.

4.4. Connections D and E

For connections D (between the server and MQTT broker) and E (between the devices and MQTT broker), the MQTT protocol is used. The protocol used here is the secure version, MQTTS, which employs SSL/TLS for secure communication. Here, the broker serves as the certificate authority. In our implementation, mutual authentication is used, meaning both the server and client authenticate each other. Furthermore, the MQTT broker has a firewall that blocks all incoming connections except those on port 8883 — the port used for secure MQTT communication. Our broker accepts authorized connections on global path mqtt.sandyuraz.com

To further secure the system, we have implemented security controls on the MQTT broker itself. From here on out, we will refer to it as the broker for brevity. As mentioned earlier, the broker supports and requires mutual authentication, meaning all clients must provide a valid and signed certificate. Although the broker is verifying all connections from clients, the broker itself poses a security risk in that it could be a single point of failure. It is responsible for both the transmission of data from sensing devices and for the transmission of device control requests from the user. Therefore, if the broker fails the whole system fails. Recall that the broker is the CA for the MQTT system, meaning all clients' certificates must be signed by the broker's private key. If this private key becomes compromised, any malicious actor would be able to generate faulty certificates for a period of time.

To prevent such key theft from occurring, additional steps have been taken to securely store master private keys. Private keys are stored on a separate mountable volume that is encrypted with LUKS.[4] The MQTT server on the broker's machine will have access to the volume only when it needs to load the keys into its memory. When the broker's server has started, the volume is re-encrypted and unmounted so that it no longer remains in memory and is therefore inaccessible to attackers who may have access to the broker's server. This way we reduce the exposure of infrastructure-critical credentials to insecure filesystems.

In addition to using MQTTS, the MQTT Mosquitto software has a set of options that can be enabled to further secure the communication. One such option is to reject all anonymous logins, thereby forcing all connecting clients to identify themselves with a Client ID. Moreover, the Client ID must contain a special prefix, otherwise the connection request will also be rejected. This allows us to restrict connection access to only clients that possess a valid signed certificate and are able to present a verified Client ID. These security options are enabled via the Mosquitto configuration settings.

5. Attack Scenarios

Each of the four attack goals discussed in Section 3 are mitigated via the security controls discussed in Section 4. We will discuss each of those goals here. Recall that our controllable device is the LED bulb and our sensing device is the temperature sensor.

5.1. Attacks Associated with Threat Model Goal 1

The first attack goal is to gain access to the current state of the controllable device. The attacker could achieve this goal in a few ways. In the MQTT system, the broker retains one message per topic. This message contains the current state of controllable devices. Newly connected clients can receive this last known good message from the broker. If the attacker is able to intercept this packet they could find this last known good message in the payload. Alternatively they could compose a properly formatted MQTT last known message request and impersonate one of the clients to get this information from the broker.

Our system prevents this type of attack from occurring by preventing anonymous logins, requiring a specified Client ID prefix, and by implementing MQTTS—the secure version of MQTT as discussed earlier. By doing so, the attacker can't impersonate any MQTT clients. Additionally, all of the packets sent between the MQTT clients and broker are encrypted so that an attacker can't perform an MitM[5] attack with the intention of reading the command payload. Furthermore, the REST protocol implemented between the webpage and the server also uses TLS thereby securing the user's command sent from the webpage to the server.

Another way the attacker might attempt to achieve this goal is via the webpage. However, the status of the LED light is not persistent on the webpage. When an authorized user selects the 'ON' or 'OFF' button, the webpage displays text that reads 'Turned ON!' or 'Turned OFF' for 1 second, indicating whether or not the LED is on. This text is only displayed on the user's webpage; so even if the attacker was logged into the dashboard they wouldn't be able to see this indicator text. The only other way the attacker could access this information is by having direct physical access to the IoT device. Since hardware access is out of the scope of this study, we do not consider this a feasible attack approach.

5.2. Attacks Associated with Threat Model Goal 2

The second attack goal is to send spoofed control commands to the controllable device. An attacker could attain this goal by gaining access to the dashboard, in which case the control commands wouldn't be spoofed because they would be coming from the dashboard on the verified webpage. The attacker could access the dashboard by obtaining a user's credentials by gaining access to the database and determining the hash algorithm used to hash the password. If the attacker is able to determine the hash algorithm we used — SHA 512 — they can perform a password dictionary attack followed by brute force hash comparison. Another way the attacker could control the controllable device is by performing an MitM attack to capture the packets containing control commands. Once captured, they could perform a reverse engineering attack to determine the format of the packet. By doing so, they could create their own spoofed packets that could then be sent by the attacker to control the device.

Our system prevents this type of attack from occurring by implementing user authentication; however as discussed, if the attacker is able to perform a brute force attack on the stored credentials in the database, they would be able to gain access to the dashboard. In order to make such brute force attacks more costly to the attacker, we could add a salt and/or pepper to the password prior to hashing. Another way our system prevents this attack goal

4. Linux Unified Key Setup, gitlab.com/cryptsetup/cryptsetup

5. Man in the Middle

is by encrypting the communication transmission between all of the connections using TLS. Because of this security control, the attacker would not be able to reverse engineer the packets captured in an MitM attack and therefore could not spoof their own packets.

5.3. Attacks Associated with Threat Model Goal 3

The third attack goal is to gain access to sensitive information collected by the sensing device. To accomplish this goal, the attacker could gain access to the dashboard via the methods mentioned in attack goal two. The current temperature data reported by the sensor (as well as temperatures over the previous 60 seconds) is displayed on the webpage. Therefore, simply gaining access to the dashboard will allow the attacker to accomplish this goal. Another way in which they can gain access to the sensing device's readings is by intercepting and reading the packets that report the current data from the IoT device to the server via an MitM attack. Finally, if the attacker is able to access the database, they will be able to read all the temperature readings ever recorded by the system.

These attacks are prevented by the user authentication control discussed earlier. Because the connections are secured via SSL/TLS, the packets will be encrypted and therefore the MitM wouldn't be able to read any of the information in said packets. It is unlikely the attacker would be able to gain access to the database because of the firewall rules that are in place and because of the restriction placed on remote superuser account access.

5.4. Attacks Associated with Threat Model Goal 4

The fourth attack goal is to spoof data reported by the sensing device. This goal could be achieved via attack scenarios similar to those used in goal two: by performing an MitM attack to capture packets, and then by reverse engineering those packets so that they can create their own spoofed packets.

Again similar to the preventions to the second attack goal, the MitM wouldn't be able to perform a reverse engineering attack to spoof packets because all packets are encrypted via SSL/TLS.

5.5. Other Attacks

Another relevant attack could occur through reverse engineering via information gained from error messages. In this type of attack, the attacker would supply bad requests to a system on purpose so that they could receive unique error messages. These unique error messages could provide the attacker with information that would help them reverse engineer the system. We saw this occur in [7]. In our system, if the authentication server returned error messages that specified whether the provided username or password is wrong, they could verify the existence of specific usernames. In doing so, the attacker is one step closer to accessing the dashboard.

This type of attack is prevented in our system through the use of obtuse error language. During the authentication process, our RESTful server only serves unique error messages until the point of actually verifying the user's credentials. When a user enters the incorrect information, they are presented with a general error message: 'bad user credentials'. This error message doesn't provide an attacker with unnecessary information that could be used to gain illegitimate access to the dashboard.

6. Limitations and Discussion

One of the primary limitations of our paper is that we do not focus on availability as a primary security goal. As a result, we do not have any protections that prevent availability attacks such as DoS attacks.

Second, there are no security controls that would prevent replay attacks. This could be easily remedied by adding a freshness token to the payload of the packets. Such a token could be a timestamp or a freshly generated nonce that expires after a set length of time. This becomes important for door locks, which would be susceptible to replay attacks without a freshness token [8].

Another limitation that could be rectified is the fact that the sensing device data stored in our database is stored in plaintext. In our application, this data is the historical temperature data. If an attacker were able to gain access to the database, they would be able to read all of the data. Depending on the system's functionality, we could encrypt the temperatures with a user's encryption key.

Finally, if an attacker has direct access to the MQTT broker server, they might be able to get the private keys when the MQTT server loads the keys from the mountable volume into memory. However, for such an attack to occur the attacker would need direct access to the MQTT server. This is highly unlikely, but we still consider the possibility.

6.1. System Extensions

In addition to implementing security controls that solve the aforementioned limitations, we could additionally expand upon our system to make it more applicable to a larger variety of IoT systems.

One such expansion involves the onboarding of new devices via a bootstrapping protocol. Currently our system supports only one IoT device and assumes the user has already onboarded that device. In order to make this system more akin to the general smart home environment, we would need to enable the user to connect new devices to their smart home system. This requires a bootstrapping protocol, which is currently an active area of research and therefore greatly expands the attack surface of the system.

Another expansion that directly follows is the user registration process. In our current setup, all authorized user credentials are stored in the database and are assumed to be pre-registered. There is no way for a user to create an account for the first time.

A final expansion that we have discussed includes allowing multiple users to access the same device at differing (or the same) levels of privilege. For instance, if multiple people live in a home that is equipped with smart home devices and each wants to have access to those devices on separate accounts they would be able to do so. Alternatively, if a vacation rental property has a smart lock on the front door and they want to provide temporary access to the renters, we could use RBAC (Role Based Access Control) to enable this. This would allow the property manager to have an admin account while the renters have temporary accounts with fewer capabilities.

7. Related Work

There are several papers where similar Smart Home testbeds are designed and implemented.

In [9], at a bootcamp for the Norwegian national team for the European cyber security challenge, attendees trained by implementing smart home testbeds. They subsequently had their teammates attack their system to identify security weaknesses. During our development process we had a fellow security-savvy student try to poke holes in our system, which led to our transitioning to obtuse error language.

The authors in [10] develop an IoT Smart Home design that focuses on power and security management. Their design involves the use of sensors related to home security — temperature, smoke, and motion sensors. This work focuses primarily on the organization of the system as opposed to its security. Our work focuses both on the organization and implementation as well as on the security of the system as a whole.

Most similar to our study is [11] where the authors analyze security problems that are unique to IoT smart home systems. They focus on "security requirements, threat models, and attacks from the smart home perspective." [11] Rather than developing and deploying a testbed as we did, the authors develop a method called the IoT Security Management System that is meant to guide IoT developers in creating a secure smart home system.

8. Conclusion

In this study, we have implemented a simplified smart home model that focuses on two broad types of devices: controllable devices and sensing devices. After design and implementation we performed a full system security analysis. Based on the results, we implemented security controls across all connections of the architecture. By creating a broadly applicable system we are able to apply the controls we implemented to a wide range of IoT smart home devices.

Acknowledgments

The authors would like to thank Dr. Fengjun Li for all the knowledge and expertise that she has imparted to us throughout the past couple of months. Additional thanks to Dylan Polson for his service as a test attacker to identify vulnerabilities within our system.

References

[1] "Forecast end-user spending on iot solutions worldwide from 2017 to 2025." https://www.statista.com/statistics/976313/global-iot-market-size/. Access Date: April 30, 2021.

[2] "Forecast market size of the global smart home market from 2016 to 2022." https://www.statista.com/statistics/682204/global-smart-home-market-size/#statisticContainer. Access Date: April 30, 2021.

[3] T. N. Gia, M. Ali, I. B. Dhaou, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Iot-based continuous glucose monitoring system: A feasibility study," *Procedia Computer Science*, vol. 109, pp. 327–334, 2017.

[4] "About let's encrypt." https://letsencrypt.org/about/. Access Date: May 1, 2021.

[5] "The 'basic' http authentication scheme." https://tools.ietf.org/html/rfc7617. Access Date: May 1, 2021.

[6] "Cross-origin resource sharing (cors)." https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. Access Date: May 2, 2021.

[7] H. Fereidooni, J. Classen, T. Spink, P. Patras, M. Miettinen, A.-R. Sadeghi, M. Hollick, and M. Conti, "Breaking fitness records without moving: Reverse engineering and spoofing fitbit," in *International Symposium on Research in Attacks, Intrusions, and Defenses*, pp. 48–69, Springer, 2017.

[8] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pp. 461–472, 2016.

[9] M. M. Yamin, B. Katt, E. Torseth, V. Gkioulos, and S. J. Kowalski, "Make it and break it: An iot smart home testbed case study," in *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*, pp. 1–6, 2018.

[10] P. Gupta and J. Chhabra, "Iot based smart home design using power and security management," in *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, pp. 6–10, IEEE, 2016.

[11] S. Erfani, M. Ahmadi, and L. Chen, "The internet of things for smart homes: An example," in *2017 8th Annual Industrial Automation and Electromechanical Engineering Conference (IEMECON)*, pp. 153–157, IEEE, 2017.

Appendix

Our source code is at https://github.com/thecsw/sandissa-dev