

Security Analysis for Smart Home IoT

Andrew Bohlmann

Computer and Information Technology

Purdue University

West Lafayette, IN, USA

abohlma@purdue.edu

Abstract—This paper outlines the creation and analysis of an Internet of Things (IoT) system. This is the final step of a semester long project for the CNIT581 – IoT Security course at Purdue University. The focus of the course and project was the security of IoT systems and what protocols and device implementations provide certain security and performance benefits and drawbacks.

Index Terms—Internet of Things (IoT), Arduino, Raspberry Pi, MQTT, edge computing, Virtual Machine (VM), TCP/IP, ESSID, access point, aircrack-ng, Wireshark, Kali Linux, deauthentication, SSL/TLS

I. INTRODUCTION

The Internet of Things (IoT) refers to the collection of devices that gather, analyze, and facilitate the transmission of data between sensors and users. This network of devices allows both individuals and corporations to enhance their homes and businesses with valuable data, increasing efficiency and productivity and enhancing quality of life. However, these benefits come at a cost. Because of the low computing power of most IoT devices, they tend to be more vulnerable to cybersecurity attacks than other more powerful internet connected devices. As the world of IoT continues to grow and become more prevalent in our everyday lives, the risk for security breaches increases tremendously, making the need for security even more crucial. The high-risk nature of IoT devices means that manufacturers must take cybersecurity seriously and make sure that their devices do not pose a significant threat to user privacy.

A. IoT Components

IoT systems are typically designed around a specific IoT architecture. Some of these architectures include three-layer, five-layer, and SOA based. Each architecture is divided into several layers that are

comprised of different components of the system. IoT systems do not necessarily have to follow a pre-defined architecture model, but at a basic level, most systems are based around the three-layer model.

The three-layer architecture model consists of the application layer, network layer, and perception layer. The perception layer contains all the physical devices and sensors that collect data from the environment. The network layer contains the hardware and software protocols used to transmit data. This includes protocols such as TCP/IP and devices such as edge computing devices. Finally, the application layer involves how users interact with the data and the system. This includes application protocols such as MQTT or DDS and devices such as cloud computing and storage solutions (AWS, Azure, self-hosted, etc.). Each layer is only able to communicate with layers above and below itself. The system developed for this project follows the three-layer model and is outlined further in the IoT Application Domain and Use Case and Implementation sections.

II. IoT APPLICATION DOMAIN AND USE CASE

Many different IoT domains exist including smart transportation, smart home, smart healthcare, and many more. Smart home consists of IoT devices and infrastructure that are situated in a person's home and connect through their home network. The smart home application domain was selected as a focus for the analysis because of the rise in popularity and abundance of smart home devices. Because smart home devices connect through a user's home network, there is also the added risk that an attacker is able to access personal devices on the network and collect data such as passwords from said devices. All internet connected devices and services have to strike a balance between security and usability,

and smart home devices tend to air on the side of usability, to make it easier for users to install and use the devices. As the number of devices and types of devices grow, security breaches become more common on top of an already inherently insecure system. Therefore, it is crucial to develop strong security techniques and identify new vulnerabilities before they are exploited by an attacker.

A. Smart Garden

The first use case that was proposed was for a smart home garden. In this use case, one or more IoT devices would be placed around the area occupied by a home garden. These devices would each have one or more sensors that would monitor the conditions of the garden and provide data that would help the user maintain optimal plant growth. Some of the data points these sensors would monitor include soil composition, soil and air temperature, humidity, and water/rain levels. Combined, the IoT devices and sensors would make up the perception layer. The network layer would consist of an edge computing device which would connect to the perception layer devices over Wi-Fi through the home router. The edge computing device would then communicate to a cloud server which would be able to process and analyze the data and accept queries from an end user for information about the data collected. To facilitate data communication, the XMPP application protocol would be used to communicate between the IoT devices and the edge device, and a basic TCP/IP internet connection would be used between the edge device and the cloud server. XMPP stands for Extensible Messaging and Presence Protocol and utilizes a stream of XML stanzas to send messages. Each XMPP transmission contains, among other things, the source address and ID of the device sending the message. This would allow each data point to be traceable to a specific device, allowing anomalies to be easily pinpointed to a specific area of the garden. On top of this, XMPP is inherently more secure than other protocols, allowing applications to easily incorporate access control, authentication, and encryption [2].

B. Sleep Environment Monitoring

The second use case that was proposed was for a sleep environment monitoring (SEM) system. In this use case, one or more sensors are connected to an IoT device which is placed somewhere in a user's bedroom. These sensors would monitor the conditions of the environment in the room to help identify possible causes for good and bad sleep. Some of these data points include body and room temperature, noise levels, and ambient light levels. One device would be placed in each room of a user's house where monitoring is desired, and additional devices could be added to act as a failsafe if the primary device malfunctions or is taken offline. Like the smart garden use case, these devices and sensors make up the perception layer. The network layer of the SEM use case also contains an edge computing device. This device acts as the broker for the MQTT application protocol. MQTT utilizes a publish/subscribe model where the devices in the perception layer publish data to a topic on the MQTT broker. Devices in the application layer (such as a cloud server or user device) can then subscribe to the topic and receive real-time data as it is published [2]. The original proposal for communication between the IoT devices and the edge computing device was Bluetooth/Bluetooth Low Energy. The use of a private Wi-Fi network hosted as an access point on the edge computing device was chosen over Bluetooth so as to better suit the native TCP capabilities of the MQTT protocol and make the overall configuration of the SEM system simpler. The edge computing device would then employ a second wireless interface to connect to the user's home network where it would be able to connect to a cloud server over a basic TCP/IP internet connection. The cloud server would be able to analyze the data received from the broker and make predictions about sleep quality and suggestions for how to improve poor conditions using AI and machine learning. From here, a user would be able to connect to the cloud server at the application layer to retrieve data and information from the analysis.

Out of the two proposed use cases, the SEM use case was chosen for the security analysis portion of the project. Details outlining the implementation of

the SEM use case can be found in the Implementation section.

III. IMPLEMENTATION

The following components were used to construct the model SEM smart home system. A diagram of the system model can be found in Figure 1.

1) Arduino Nano 33 IoT

The Arduino Nano 33 IoT was chosen as the primary IoT device because of its IoT centric capabilities. This version of the Arduino Nano offers support for Bluetooth, Bluetooth Low Energy, and 2.4 GHz Wi-Fi. This device is also compatible with the Arduino IDE which made the process of coding and troubleshooting very seamless. The packages required for the temperature sensor below as well as the MQTT protocol came integrated into the Arduino package manager, leading to a straightforward installation process. The Arduino Nano 33 IoT also comes equipped with a dedicated cryptographic processor which was not utilized in testing but could be used to create a more secure system/device in a real-world application.

2) Adafruit MCP9808 Temperature Sensor

The Adafruit MCP9808 temperature sensor sends data to a connected device via the Inter-Integrated Circuit (I2C) protocol. Only one sensor was used during the analysis but the I2C bus would allow multiple sensors to be connected using only a single set of data (SDL) and clock (SCL) pins. Along with the ease-of-use and simplicity of the I2C bus, this sensor was also chosen because of its high degree of accuracy ($\pm 0.25^{\circ}\text{C}$) and precision (up to $1/10000^{\circ}\text{C}$) compared to other readily available temperature sensors like the DHT11.

3) Raspberry Pi 4 Model B

The primary role of the Raspberry Pi 4 Model B within the model system is to function as the broker for the MQTT protocol and facilitate data transmission between the IoT device and the cloud server. The Arduino Nano publishes temperature data to a topic on the broker. The cloud server then subscribes to that topic and collects and stores the data.

4) Windows 11 Laptop

Due to limited access to dedicated hardware, a Windows 11 laptop served as a host for both the cloud server virtual machine (VM) and attacker VM. Oracle VirtualBox was used as the hypervisor to run both VMs on the Windows laptop. The virtualization of the cloud server and attacker machine posed several issues when attacking the system which are covered further in the Challenges section.

5) TP-Link AC1300 USB Wi-Fi Adapter

Because the attacker machine was running as a VM, it did not have direct access to the Wi-Fi network card on the host laptop. In order to give the VM a wireless interface to execute attacks, a TP-Link USB Wi-Fi adapter was connected to the host laptop. VirtualBox was then able to recognize and mount the adapter to the attacker VM, where it showed up as a usable wireless interface.

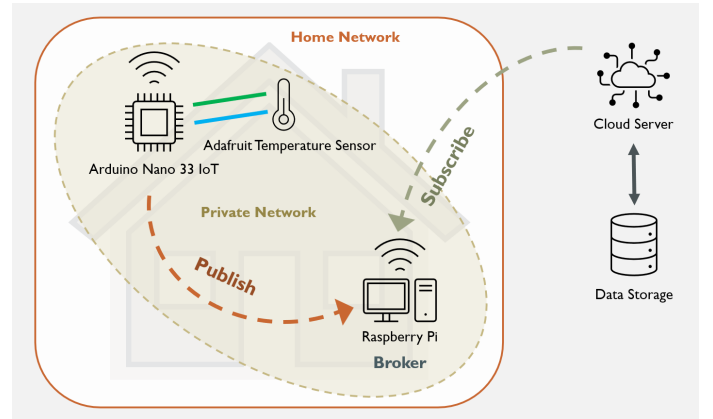


Fig. 1. Diagram of implemented SEM use case

A. System Operations

The model SEM system uses one Arduino IoT device with one temperature sensor, one Raspberry Pi edge computing device, one VM cloud server, and one VM attacker machine to simulate how the system would operate at a very basic level if implemented in a real-world environment. The order of operations within the system is as follows:

- 1) Temperature sensor reads temperature of environment and communicates data to the connected Arduino IoT device using the I2C bus.

- 2) Arduino formats temperature data into a readable message and publishes the message to the MQTT broker (Raspberry Pi edge device) on the topic room/temperature which would change depending on the room in a real-world implementation (i.e. kids/temperature, master/temperature, guest/temperature, etc.).
- 3) Cloud server subscribes to the topic on the broker and saves the collected data to a text file for analysis.
- 4) User connects to a web application hosted on the cloud server and is able to view data in graphical and tabular form (however, no user interface, connection, or device was implemented in the model system).

IV. SECURITY THREATS AND ANALYSIS

A Kali Linux VM was used to analyze the security of the model system and execute attacks. Because of the use of a VM instead of dedicated hardware, an external USB Wi-Fi adapter had to be used in order to give the VM an accessible wireless interface to execute attacks and analyze network traffic from. Two attacks were conducted to test the system's security. Due to hardware limitations, the model had to be modified slightly to execute the attacks. Further explanation can be found in the Challenges section.

A. Deauthentication Attack

The first attack that was conducted was a Wi-Fi deauthentication attack. The aircrack-ng library that comes pre-installed on Kali Linux was used to execute the attack. By first running the airmon-ng command, a list of available wireless networks and access points was given as output with information such as BSSID, ESSID, authentication type, and encryption type. The main piece of information that is relevant to this attack is the ESSID (to make sure the correct network has been identified by using the known name of the network) and the BSSID (used as input in the aircrack-ng commands). After the BSSID of the Raspberry Pi's access point was identified, the airodump-ng command could be run. By providing the BSSID as input, any devices connected to the network and their MAC addresses could be identified. In the case of the SEM model,

the Arduino was identified with the command. Finally, the aireplay-ng command could be run to flood the target device with deauthentication frames, kicking it off the network. This command can either be used to specifically target the Arduino, essentially acting as a denial-of-service attack by keeping it disconnected from the network or used to target the access point itself. By targeting the access point, the Arduino maintains its network connectivity, but is still unable to connect to the access point hosted on the Raspberry Pi edge device. This allows an attacker to configure a replica access point (with the same ESSID and password) to capture the Arduino IoT device. This was successfully attempted during the attack testing. The Raspberry Pi access point was continuously flooded with deauthentication frames. While the deauthentication attack was running, the Arduino attempted to reconnect to the access point. Since it was still being flooded with deauthentication frames, the only available network the Arduino could connect to was the one-to-one copy of the access point running on the attacker's machine. Figure 2 shows the Arduino connected to the fake access point after running the attack.

B. Packet Sniffing Attack

The second attack that was conducted was a packet sniffing attack between the cloud server and the Raspberry Pi broker. Because of difficulties configuring secure MQTT over TLS/SSL, a less secure password protect version of MQTT had to be implemented. Because the MQTT packets were unencrypted, an attacker would be able to easily intercept the packets and see the messages, username, and password in plaintext. Because the cloud server VM and attacker VM were sharing a network interface during the attack, it was very simple to view the packet stream. In a real-world application, the attacker would have to gain control of the access point network or one of the devices in order to easily view the packet stream. By simply using Wireshark and filtering by protocol, the MQTT CONNECT (containing the username and password) and PUBLISH (containing the MQTT message) packets were very easily found. Figure 3 shows the CONNECT packet found in Wireshark.

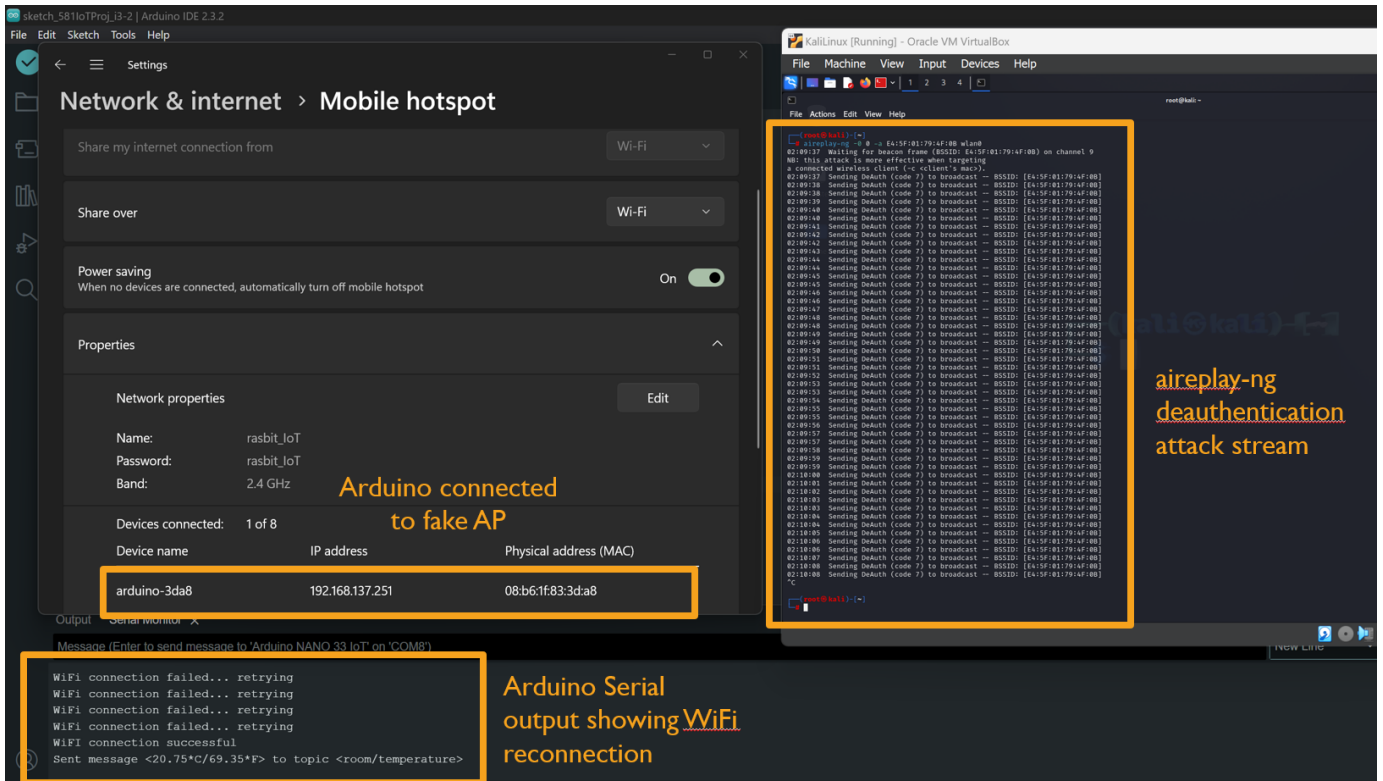


Fig. 2. Arduino IoT device connected to the attacker's fake access point after executing a deauthentication attack against the Raspberry Pi edge device

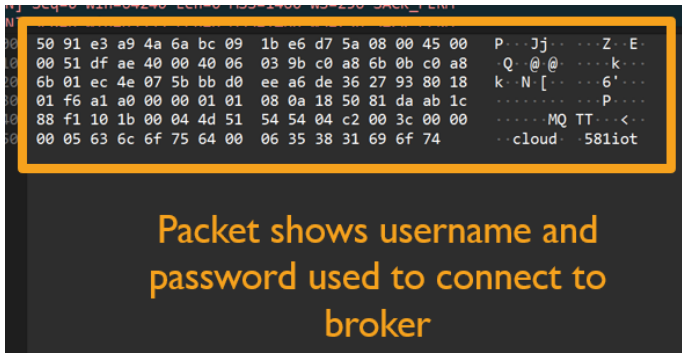


Fig. 3. Username and password used to connect to the MQTT broker found in the MQTT CONNECT packet using Wireshark

C. Analysis

With both attacks combined, an attacker can learn the IP address of the broker, the BSSID of the access point, the MAC address of the Arduino, the MQTT topic, and the MQTT username and password. If the attacker is able to learn the password for the access point to configure the fake access point, then the attacker could execute the deauthentication attack,

gain control of the Arduino, and then intercept the packets to learn valuable information about the MQTT functionality. From here, an attacker could easily execute denial-of-service attacks, or even create a fake device and publish fictitious data to the broker.

D. Suggested Remediations

Based on the attack testing, the model SEM system clearly has several security vulnerabilities. The following solutions could potentially remedy several of the major vulnerabilities present.

1) Implement MQTT over SSL/TLS

The use of keys instead of a username and password, along with SSL/TLS encryption would make it very difficult for an attacker to replicate the packet sniffing attack. Certificates could also be used to authenticate devices to the broker, reducing the risk of an attacker creating a fake device.

2) Implement access point verification on IoT device

By implementing some sort of access point verification (such as checking the MAC address of the access point, or the addressing used) could prevent a device from connecting to/staying connected to a fake access point.

3) Implement secure message transmission

On top of implementing MQTT over SSL/TLS, messages could be encrypted on the IoT device and sent as a random string of alphanumeric characters. The cloud server would then be able to decrypt the messages, preventing an attacker from viewing the messages via a packet sniffing attack. This could also be a way to add message verification, further reducing the risk of fake devices being created by an attacker.

V. CHALLENGES

Several issues were encountered during the security analysis. The main issue with the system was the hardware limitations. Because the attacker machine needed to be run as VM, a USB Wi-Fi adapter was required (since VirtualBox and other type 2 hypervisors cannot directly access wireless hardware interfaces on the host device). This required a slight modification to the overall SEM system. Instead of operating two separate wireless interfaces on the Raspberry Pi edge device, a single interface was used to host a private access point that the Arduino IoT device and cloud server VM would connect to. This would be akin to operating the entire system on a user's home network. Another issue that necessitated this change to the system was issue working with an enterprise network. Since the analysis was conducted on Purdue University's campus, a private network was required in order to properly configure it and so as to not interfere with other users on Purdue's PAL3.0 network.

Issues with configuring the Arduino also required the use of password protected MQTT instead of secure MQTT over SSL/TLS. Bad certificate errors would be encountered each time a SSL/TLS configuration was attempted. One or both of the attacks would most likely have had issues or completely failed if SSL/TLS was successfully implemented.

VI. CONCLUSION AND POSSIBLE FUTURE WORK

MQTT is one of the most widely used protocols for IoT and offers many benefits such as being lightweight and efficient for even low power IoT devices. However, MQTT is inherently insecure and requires an SSL/TLS implementation to achieve security functionality on par with a secure protocol such as XMPP. The insecurity of MQTT and the use of weak passwords on other IoT system components pose major security risks to the smart home domain. The attacks outlined in this paper were surprisingly easy to execute, and even though the system could have been much more secure, there were still some security measures in place. It is important that manufacturers of smart home devices keep this in mind and work to develop more secure devices and systems. Further analysis would need to be done to determine if other protocols (such as XMPP) can outperform MQTT from a security standpoint, and if so, is it worth it to sacrifice on performance and stability.

REFERENCES

- [1] *Inside the smart home: Iot device threats and attack scenarios*. Security News. (n.d.). <https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/inside-the-smart-home-iot-device-threats-and-attack-scenarios>
- [2] I-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4), 2347-2376.
- [3] Nebbione, G., & Calzarossa, M. C. (2020). Security of IoT application layer protocols: Challenges and findings. *Future Internet*, 12(3), 55.
- [4] Swamy, S. N., Jadhav, D., & Kulkarni, N. (2017, February). Security threats in the application layer in IOT applications. In *2017 International conference on i-SMAC (iot in social, mobile, analytics and cloud)(i-SMAC)* (pp. 477-480). IEEE.
- [5] Andy, S., Rahardjo, B., & Hanindhito, B. (2017, September). Attack scenarios and security analysis of MQTT communication protocol in IoT system. In *2017 4th international conference on electrical engineering, computer science and informatics (EECSI)* (pp. 1-6). IEEE.