Computer Security Final Report

Ruhao Xin

New York University, Shanghai

May 15, 2022

1 Introduction

I study some security vulnerabilities in IoT platforms which use information transfer protocol MQTT. MQTT is a lightweight protocol designed exactly for IoT platforms. I design a system called SWS (Shared With Security) to solve these security vulnerabilities. It shows that this system is very efficient to prevent some specific attacks, such as unauthorized will message attack, unauthorized retained message attack and non-updated session attack.

2 Background

2.1 Cloud-based IoT platform

The structure of a cloud-based IoT platform can be seen in the Figure 1:

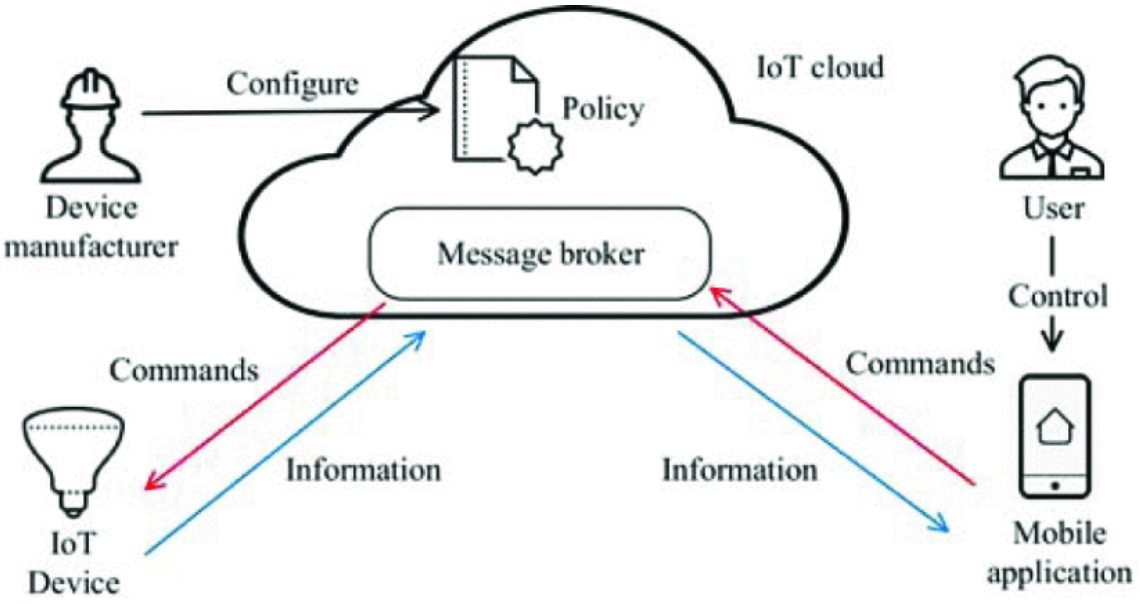

Figure 1: A cloud-based IoT system typically includes three components: the cloud, the IoT device, and the user's management console (mobile apps in particular) to control the device [1]

2.2 MQTT protocol

How the MQTT protocol works can be seen in the Figure 2:

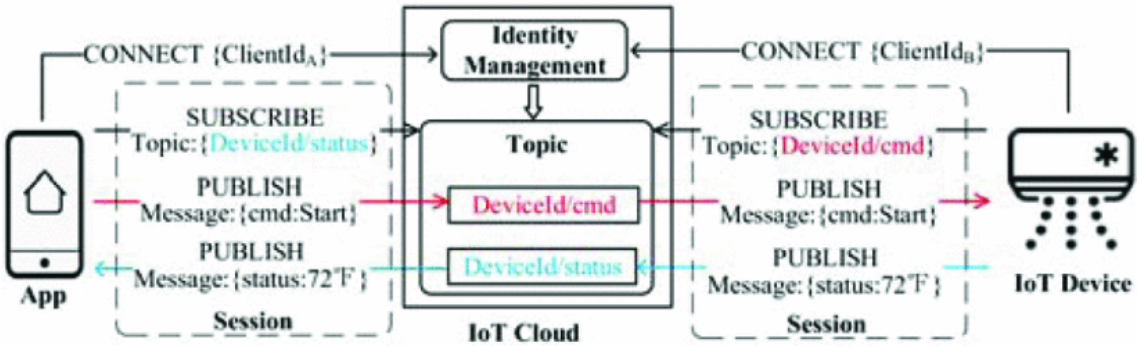

Figure 2: In the MQTT communication, the client sends three basic types of messages to the broker, CONNECT, PUBLISH and SUBSCRIBE. First, the MQTT client, e.g., a smart air conditioner or an app, sends a CONNECT message to the broker for establishing an MQTT session (if the broker accepts the connection). The session and the client are uniquely identified by a ClientId field (embedded in CONNECT message), which is similar to a web session cookie. In the established session, an IoT device subscribes to its associated topic (e.g, /DeviceId/cmd) by sending a SUBSCRIBE message (including its topic) to the broker. The broker maintains the subscription status for each session and delivers the MQTT message published to a topic to its subscribers. [1]

In this process, the whole MQTT communication relies on four entities: identity (ClientId), message, topic and session. Hence, whether these entities have been well protected is critical to the protocol's secure application to the IoT environment.

2.3 Current protection of MQTT on IoT Clouds

Since MQTT is not designed to work in an adversarial environment, it lacks build-in authentications and authorizations. Therefore, it is the cloud platform owners' task to build some mechanisms to protect the message communication.

Client authentication

IoT cloud platforms authenticate their MQTT clients using their own platform-layer identities, such as Amazon account, Google account, Facebook account and so on. Besides this, the authentication done by the TLS protocol is also supported in some IoT platforms.

Client authorization

The IoT cloud platform aims to ensure that each user can only send commands to and receive messages from the devices the person is allowed to use. For this purpose, the cloud enforces a set of security policies. Examples include the topics and messages a client is allowed to access and the actions (e.g., publish or subscribe) it can take.

3 Threat Model

We assume that the adversary can have a valid platform-layer identities to use the IoT cloud. Besides, they are also capable of collecting and analyzing network traffic between clients, cloud and the devices. However, we assume that the ID information is protected very well (although in reality they are not) by some encryption methods so that the adversary cannot get any ID information of clients and devices through the network flow. More importantly, since device-sharing situations of IoT devices is very pervasive, it is reasonable to assume that the adversary can have temporary accesses to target devices. Based on these assumptions, the followings are some attacks that the adversary can do:

3.1 Unauthorized Will Message Attack

This attack takes advantage of the function of the IoT system that once the client is accidentally disconnected (i.e., not sending a DISCONNECT message to the broker), the broker will publish

the Will Message to all subscribed clients of the topic, allowing them to take corresponding actions. In this case, a malicious ex-user can strategically register a Will Message to trigger it later when he no longer has the access privilege, to stealthily issue commands when the device is serving other users.

3.2 Unauthorized Retained Message Attack

It takes advantage of the function of IoT system that the MQTT client can register a Retained Message with a topic (by setting the retained flag in a regular MQTT message), which allows the broker to keep the last Retained Message on the topic, and publish it immediately to any future subscribers to the topic. In this case, a malicious ex-user can stealthily command a device he no longer has access to by purposefully leaving a retained message to a topic and waiting for other users to subscribe to it.

3.3 Non-updated Session Attack

It takes advantage of the flawed structure of MQTT protocol. Currently, the security policy of the MQTT protocol is that it treats the clients as subject and devices as objects. Under this structure, when a device is reset by a new user (for removing exusers' access), although permissions of the ex-user (and his/her client) for accessing the device are revoked (i.e., publish/subscribe messages through the device's topic), there is no concept of revoking the permission of a device for accessing its topic. In this case, an adversary can get the topic information of a device through the network analysis when he still has the privilege. Then the adversary can use their own ID to pretend as a device to publish some wrong message to this topic even if his privilege has been revoked, and influence users subscribing to this topic.

4 System Design

The reason why these kinds of attack happen is that by default, both the clients and devices are able to receive and execute any message from the broker, which actually violates the principle of fail-safe default and complete meditation. As a result, to fix this issue, it is required to implement a checking process for authentication when the clients and devices want to receive a message from the broker.

Access Control Policy

Subjects: The set of the clients in the communication, such as devices and users.
Objects: The set of messages that subjects hold rights on, like the CONNECT, SUBSCRIBE and PUBLISH message.
Operations (Rights): Receive a message.
Access Control Policy: Attribute-based access control

System Design

In summary, the attribute here is a Boolean value representing whether the receiver of the message is in the same trusted group with the sender of the message. If so, then the receiver have the right to receive the message. Or this message should be discarded. To achieve this, the following components are required:

Initialization

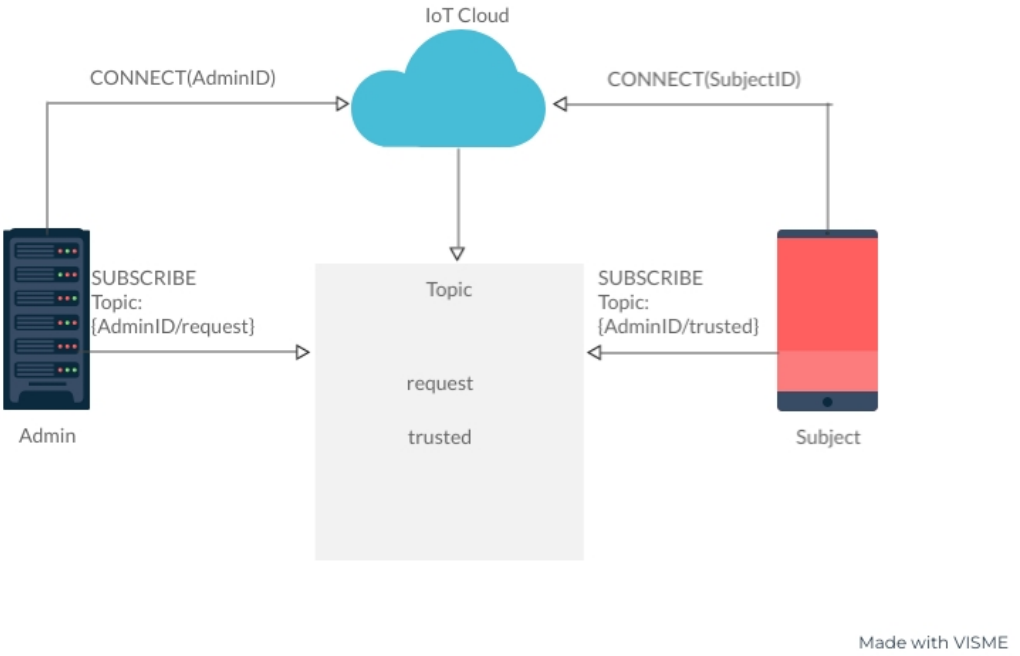

Figure 3: This component is used to create a topic for the administrator and the subjects (including both clients and devices). One point of this initialization is that each IoT Device can be only allocated by one administrator.

Editing the trusted groups

This component doesn't need the structure of IoT Cloud because in this system, the trusted group is stored in the terminal of administrators. It is the administrators themselves that determine which clients or devices can be added into the same trusted group or which clients or devices should be removed from the same trusted group.

Request from IoT Subjects

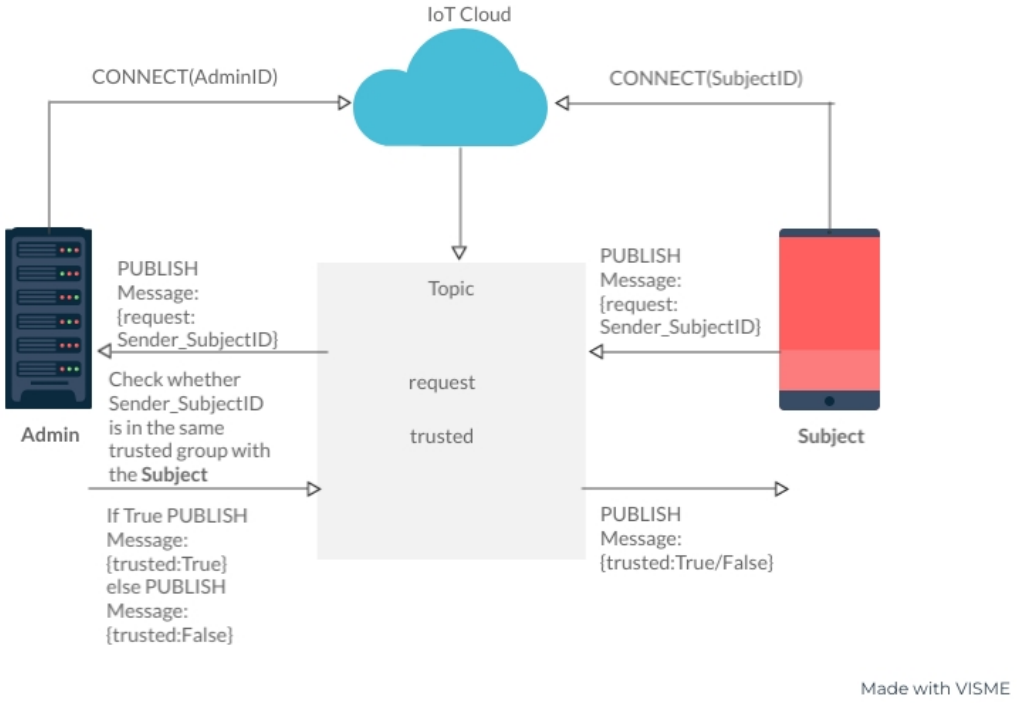

Figure 4: This component is used for a IoT subject to request whether the sender of a message is in the same trusted group with itself before it receives this message.

Reference Monitor

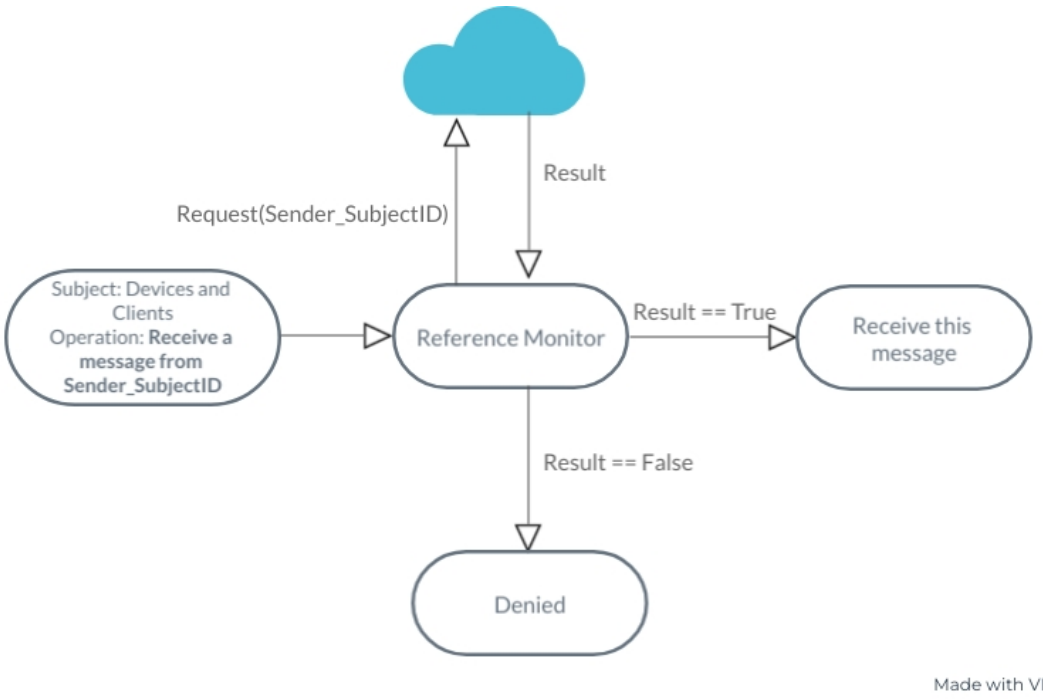

Figure 5: This component is used to implement the access control.

5 Implementation

To implement this system, firstly it is necessary to simulate an IoT cloud platform based on MQTT in Python, which contains the following parts:

object.py: Implement the message class in MQTT protocol. The message class has three attributes: owner (representing the sender of this message), topic (representing the topic this message is sent to) and content (representing the content of the message).

subject_user.py: Implement subject class in MQTT protocol, which is allowed to do connect, subscribe, publish and receive messages. Moreover, the admin is a special kind of subject, which can not only do all things subjects can do, but also maintain a trusted group. As a result, the admin class is inherited from the subject class.

server.py: Simulate the IoT cloud based on MQTT protocol, which is used to handle messages communications.

After that, the reference monitor can be implemented:

reference_monitor.py: Implement a reference monitor based on MQTT protocol. It will check whether a message's owner is in the same group as the message's receiver. If true, then it will return 1 to the receiver. Or it will return 0. Moreover, if a message's owner is the admin, the reference monitor will not go through the checking process and let the message go directly.

The complete code can be seen in my github site.

6 Attack Analysis

Based on the structure above, next I will analyze how the reference monitor can prevent the attacks I mentioned in the threat model section.

Unauthorized Will Message Attack

If there is no reference monitor, this attack works in the following way:

```
Initilization IoT environment...
Build 1 admin, 1 user, 1 adversary and 2 devices.
Admin ID:0
device1 ID:1
device2 ID:2
user ID:3
adversary ID:4
Connect them to the IoT Cloud:
The adversary has a will message 'Some malicious commands'
Subscription: device 1 -> 1/cmd
 admin -> 1/status
 user -> 1/status
 adversary -> 1/cmd
Subscription: device 2 -> 2/cmd
 admin -> 2/status
 user -> 2/status
 adversary -> 2/cmd
Then both users publish commands to both devices
User ------Start1 from user------> 1/cmd
User ------Start2 from user------> 2/cmd
Adversary ------Start1 from adversary------> 1/cmd
Adversary ------Start2 from adversary------> 2/cmd
What device1 get until now: [(3, '1/cmd', 'Start1 from user'), (4, '1/cmd', 'Start1 from adversary')]
What device2 get until now: [(3, '2/cmd', 'Start2 from user'), (4, '2/cmd', 'Start2 from adversary')]
After that, the adversary tries to purposefully disconnect from the cloud, triggering the will message
What device1 get after disconnection of adversary: [(3, '1/cmd', 'Start1 from user'), (4, '1/cmd', 'Start1 from adversary'), (4, '1/cmd', 'Some malicious commands')]
What device2 get after disconnection of adversary: [(3, '2/cmd', 'Start2 from user'), (4, '2/cmd', 'Start2 from adversary'), (4, '2/cmd', 'Some malicious commands')]
```

Figure 6: You can see that if the cloud platform doesn't implement the reference monitor, both the device1 and device2 will receive the malicious will message from the adversary.

After implementing the reference monitor, this attack will be prevented in the following way:

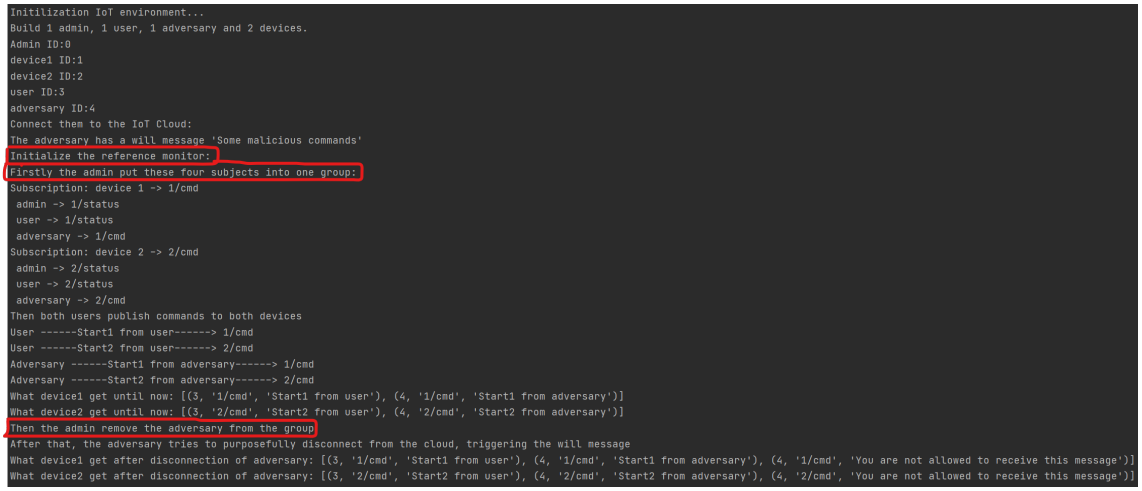

Figure 7: After considering that the adversary should lose its privilege, the admin have to remove it from the group immediately. In this case, the reference monitor will find that these will messages don't come from the same group of the devices so that it will block the devices receiving them, replacing its content as 'You are not allowed to receive this message'. The red part in the figure demonstrates how this system works in this simulated environment.

Unauthorized Retained Message Attack

If there is no reference monitor, this attack works in the following way:

Figure 8: You can see that if the cloud platform doesn't implement the reference monitor, both the device1 and device2 will receive the malicious retained message from the adversary.

After implementing the reference monitor, this attack will be prevented in the following way:

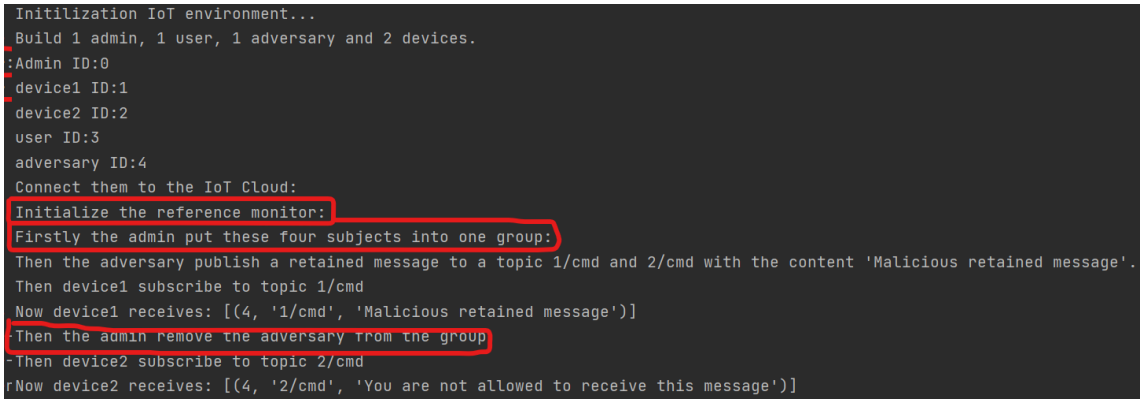

Figure 9: In this case, the device1 is allowed to receive retained message from the adversary while the device2 cannot. The reason is that after device1 receives the retained message, the admin removes the adversary from the trusted group. By doing this, this system can prevent the Unauthorized Retained Message Attack while allowing valid retained message communication between trusted subjects. The red part in the figure demonstrates how this system works in this simulated environment.

Non-updated Session Attack

If there is no reference monitor, this attack works in the following way:

```
Initilization IoT environment...
Build 1 admin, 1 user, 1 adversary and 2 devices.
Build 1 admin, 1 user, 1 adversary and 2 devices.
Admin ID:0
device1 ID:1
device2 ID:2
user ID:3
adversary ID:4
Connect them to the IoT Cloud:
Subscription: device 1 -> 1/cmd
 user -> 1/status
Then the user ask the device to close
Unfortunately, it fails. So the device plans to respond 'Fail' to the user
However, the adversary is faster than the device to respond 'Success' to the user
Now the user receives: [(4, '1/status', 'Success'), (1, '1/status', 'Fail')]
```

Figure 10: You can see that if the cloud platform doesn't implement the reference monitor, the user1 will firstly receive the message from the adversary, telling it that the device1 closes successfully, which contradicts with the reality.

After implementing the reference monitor, this attack will be prevented in the following way:

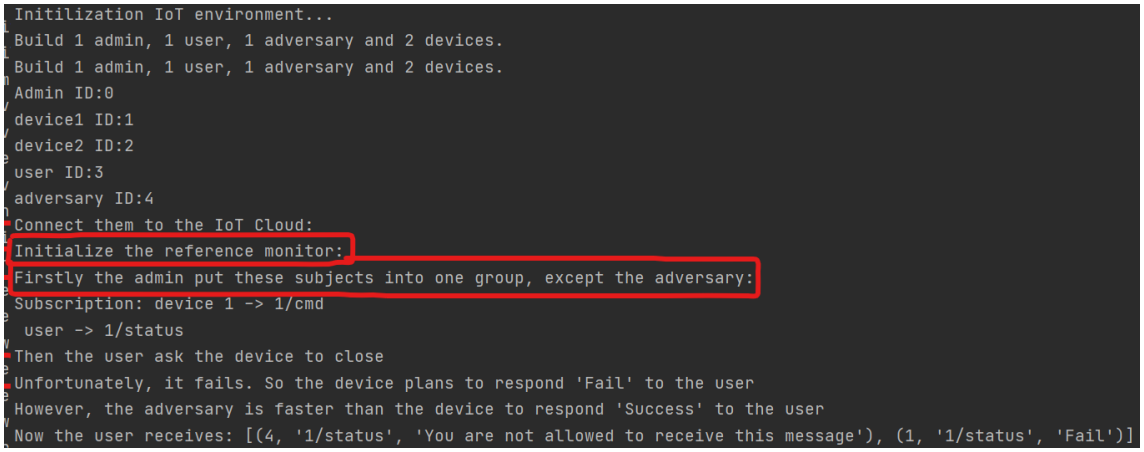

Figure 11: In this case, even if the user1 receives the message from adversary first, the reference monitor finds that the adversary is not in the same trusted group with user1. As a result, the reference monitor blocks user1 receiving this message. So the consequence is user1 only receives one message from device1, telling it that its command fails. The red part in the figure demonstrates how this system works in this simulated environment.

References

[1] Yan Jia, Luyi Xing, Yuhang Mao, Dongfang Zhao, XiaoFeng Wang, Shangru Zhao, and Yuqing Zhang. Burglars' iot paradise: Understanding and mitigating security risks of general messaging protocols on iot clouds. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 465–481, 2020.