A black and red sign with white text

Description automatically generated

Swinburne University of Technology

**Internet and Cybersecurity for Engineering Applications**

Cybersecurity Assessment Report for MQTT Broker

Marco Giacoppo (104071453)

Monday 12:30

2023 Semester 2

# Introduction

## Problem Statement:

The rapid expansion of the Internet of Things (IoT) has introduced a new set of challenges in securing IoT environments. One critical component in IoT architecture is the MQTT broker, which plays a pivotal role in facilitating communication between IoT devices. The primary challenge is to ensure the security and integrity of the MQTT broker to prevent potential vulnerabilities and threats.

## Objectives:

The primary objectives of this report are as follows:

1. Identify potential vulnerabilities in the MQTT broker used in the IoT infrastructure.
2. Recommend security measures to address these vulnerabilities and enhance the overall security of the IoT solution.

# MQTT Broker Overview:

The MQTT broker is the central component of the IoT infrastructure. It serves as a message broker, receiving messages from IoT devices and relaying them to subscribed clients. The MQTT protocol is known for its lightweight nature and efficiency in IoT applications. As the broker manages critical data, its security is of paramount importance.

* **MQTT Broker**: Serves as a message broker responsible for receiving and relaying messages between IoT devices.
* **Publisher**: A component responsible for generating data and posting messages to specific topics.
* **Subscriber**: A component that subscribes to topics and prints received messages on the screen.

# MQTT Broker Security Assessment:

As part of this cybersecurity assessment, the MQTT broker was evaluated, and several security issues were identified:

1. **Authentication Weaknesses:** The current authentication mechanism may not be sufficient to verify client identities effectively.
2. **Encryption Concerns:** The lack of end-to-end encryption exposes data during transit to potential threats.
3. **Authorization Limitations:** Existing authorization policies need refinement to ensure secure data access.

***Recommendations:***

To strengthen MQTT broker security, the following recommendations are provided:

1. **Enhance Authentication:** Implement more robust client authentication methods to verify client identities.
2. **Implement Encryption:** Introduce end-to-end encryption to safeguard data privacy during transmission.
3. **Strengthen Message Authentication:** Enhance message authentication mechanisms to maintain data integrity.
4. **Review Authorization Policies:** Ensure strict authorization policies to prevent unauthorized access.
5. **Optimize Broker Configuration:** Align the MQTT broker’s configuration with security best practices.

# Code Functionality:

The solution’s core functionality can be divided into the following major components:

1. **MQTT Connection Setup:**

* The code initiates a connection to the MQTT broker hosted on ‘rule28.i4t.swin.edu.au’ over port 1884, the default MQTT port.
* For enhanced security, the MQTT client configuration is tailored with a unique client ID, username, and password. This setup ensures secure authentication and access to the MQTT broker.

1. **Message Publication and Subscription:**

* Publisher Component (cpubsub.py):
* This component is responsible for generating threat data and publishing messages to the ‘104071453/threats’ private topic.
* The publisher is equipped to subscribe to ‘public/#’, which allows it to monitor and display public messages from the broker.
* Subscriber Component (csub.py):
* The subscriber component connects to the MQTT broker with a client ID, username, and password for secure authentication.
* It also subscribes to the ‘104071453/threats’ private topic to receive threat information and subscribes to ‘public/#’ to monitor public messages.
* In addition to message reception, the subscriber provides logging functionality by displaying logs in the terminal.

1. **Screenshots:**

* To provide a comprehensive understanding of the code, here are several key screenshots that illustrate different aspects of the functionality:

# 

Figure : MQTT Connection Configuration.

This screenshot provides a detailed insight into the MQTT connection configuration, a crucial component of the solution.

* **Broker Address:** The ‘broker’ field is set to ‘rule28.i4t.swin.edu.au’, designating the MQTT broker’s address where the code establishes the connection.
* **Port Configuration:** The port number ‘1883’ is specified, which is the default MQTT port for communication with the broker.
* **Client ID:** The ‘client\_id’ in the script is populated with ‘PubSubClient’, representing the client’s unique identification. This ensures secure and unique access to the MQTT broker.
* **Security Credentials:** The field ‘username’ and ‘password’ are set to ‘104071453’, providing the necessary authentication for secure access to the broker.

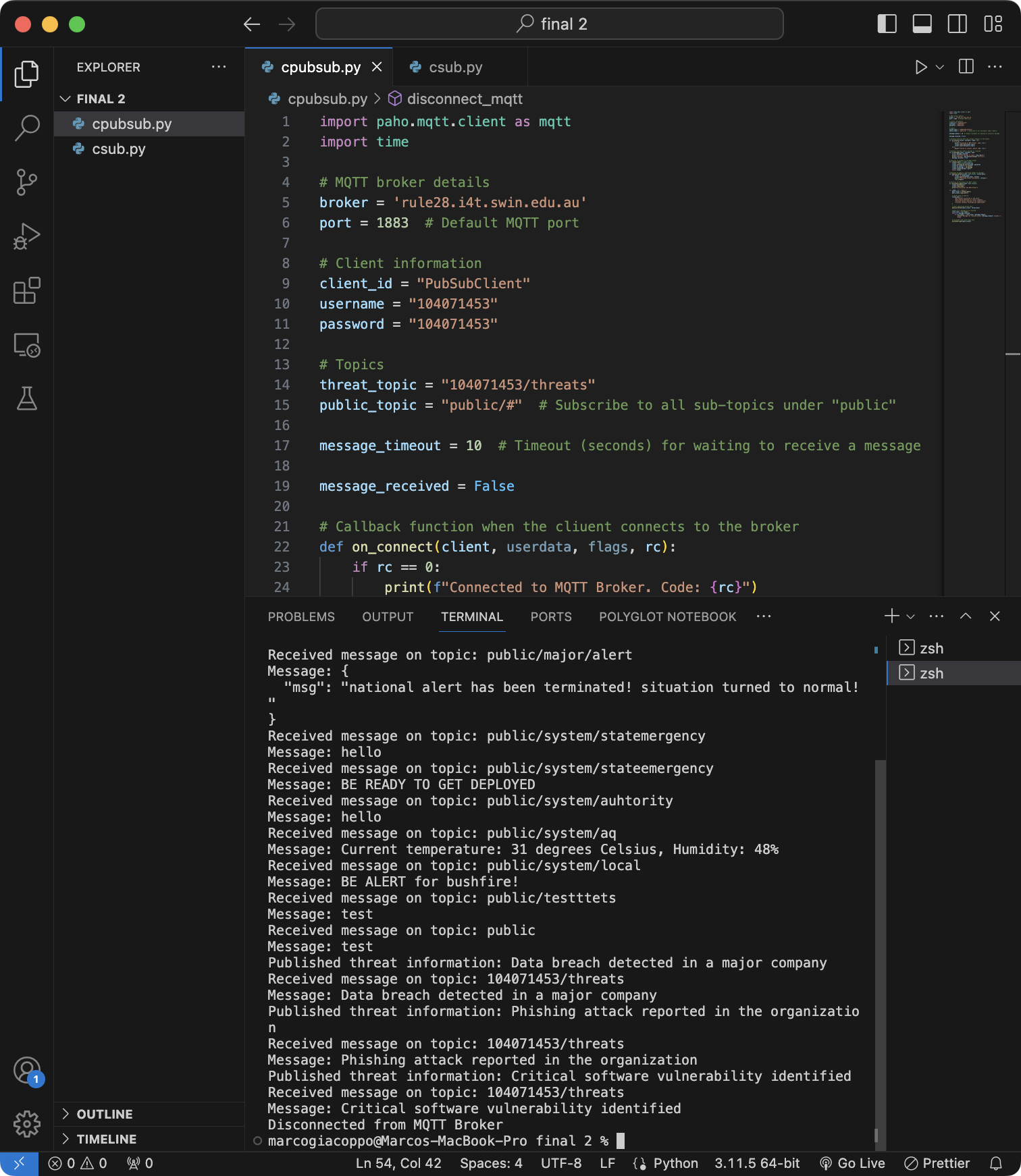


Figure : Publisher Components.

A snapshot of the publisher component, showing the publishing process of threat information. Each threat message is sequentially published to the ‘104071453/threats/ private topic, where it becomes availability for subscribers. The screenshot provides a visual representation of the code’s ability to generate and publish threat information.

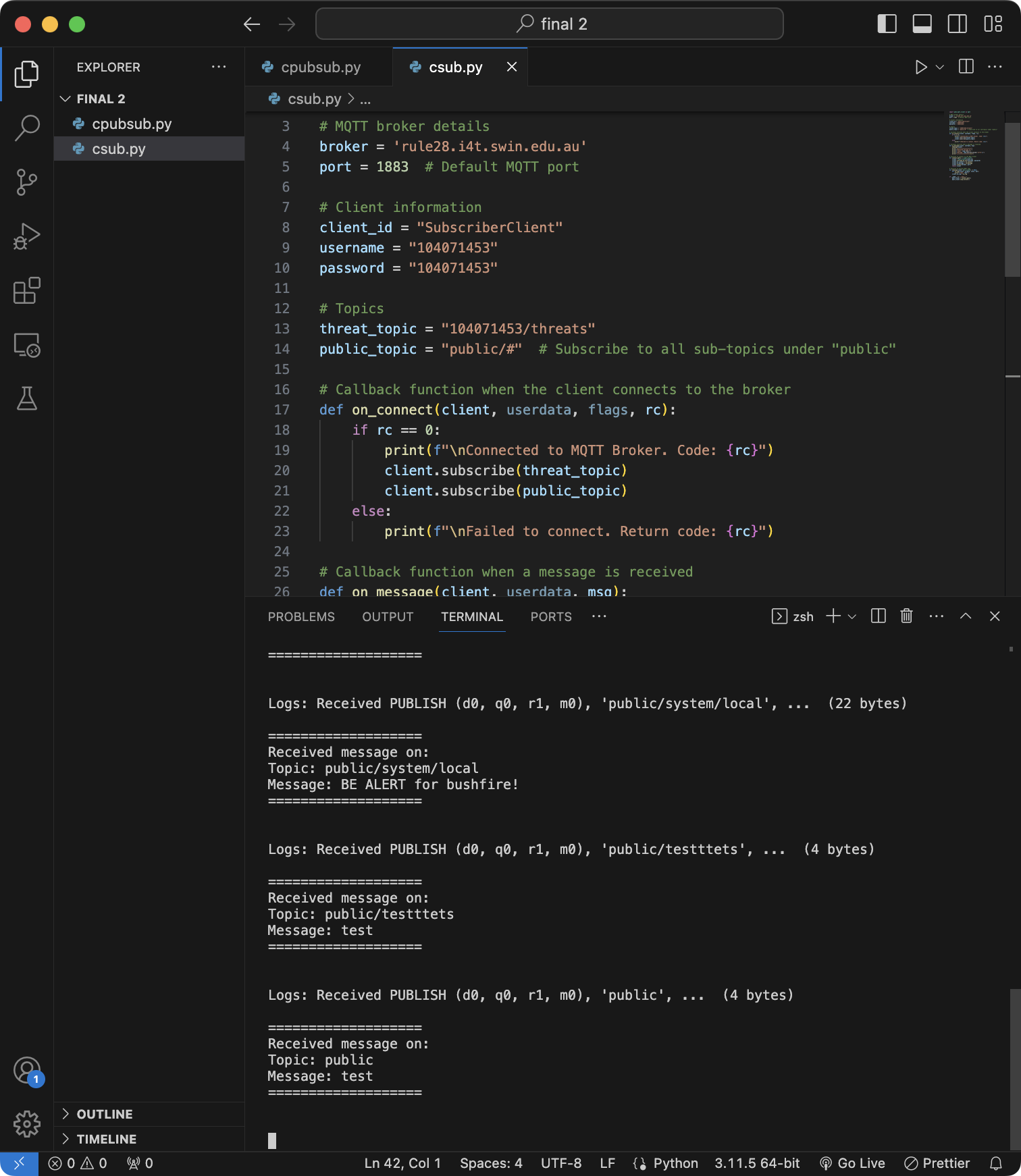


Figure : Subscriber Component.

In this screenshot, the ‘csub.py’ code demonstrates the functionality of the subscriber component.

* **Message Reception:** The subscriber component is actively receiving and displaying messages from the MQTT broker.
* **Threat Data Display:** The subscriber effectively displays received threat data, showing the messages previously published by the publisher script.
* **Public Message Display:** Additionally, the subscriber successfully monitors and displays public messages from the ‘public/#’ topic, highlighting the broader utility of the code.

These screenshots collectively offer a visual representation of the code’s functionality and the MQTT broker’s interaction with the code. They provide insights into the code’s rile in publishing and subscribing to messages, reinforcing the understanding of its operation and security aspects.

# Code Execution Instructions:

1. **Prerequisites:**

* Make sure you have Python3 installed on your system.

1. **Install Required Libraries:**

* You may need to install the Paho MQTT library if it’s not already installed. You can do this using pip: **`pip3 install paho-mqtt`**

1. **Run the script:**

* Open your terminal or command prompt.
* Navigate to the directory where the script is located.
* Run the script with Python3 by entering the following command:

**`python3 csub.py`**

* The script will start running and connect to the MQTT broker. It will subscribe to specific topics and print received messages to the terminal.

1. **Leave the Subscriber Running:**

* Let the subscriber script run in the background.

1. **Open Another Terminal:**

* Open a separate terminal or command prompt to run the Publisher script.
* Navigate to the directory where the publisher script is located.
* Execute the publisher script using Python3 by entering the following command:

`**python3 cpubsub.py`**

* The script will start publishing messages to the MQTT broker.

1. **Observe the Subscriber Terminal:**

* Switch back to the terminal where the subscriber script is running.
* You will see the messages being received and printed to the terminal.

1. **Check for Successful Communication:**

* Verify that the subscriber script successfully receives and displays the messages published by the publisher script.

1. **End the Scripts:**

* You can stop both the publisher and subscriber scripts manually by pressing `**Ctrl + C`** in their respective terminals.

# Conclusion:

An in-depth analysis of the functionality of the publisher and subscriber MQTT scripts has been given in this report. The real-world applications of these scripts, which go beyond simple code execution, are highlighted by the practical demonstration of them.

These scripts can be used as efficient alert systems for various scenarios in a practical setting. For instance, they can act as watchful defenders, alerting users right away to potential data breaches, cybersecurity dangers, or important events in an IoT environment. They are useful tools in the field of cybersecurity due to their adaptability and simplicity of use.

Finally, the MQTT publisher and subscriber scripts presented here are both educational and useful tools that can be used to increase security and vigilance in the constantly changing Internet of Things environment.

# Acknowledgement:

I would like to express my sincere gratitude to my tutor, Dragi Klimovski, for imparting knowledge and providing valuable guidance throughout my learning journey. Your support and expertise have been instrumental in shaping my understanding. Thank you for your unwavering dedication to my education. Hope I’ll see you again someday!