Secure Code Review for CyberSecuirtyMQTT

<https://github.com/redbackoperations/iot/blob/main/Research/CyberSecurityMQTT/MQTT_data_frame_handler.py>

# Introduction:

This documentation is Secure Code Review for MQTT\_Data\_Frame\_Handler.py which is designed to handle data transmission via MQTT. The code aims to encrypt and transmit data securely over the MQTT protocol using the Paho MQTT client library. In ths documentation, I will be addressing my reviews.

# Code Overview:

The Python code is to transfer the data over MQTT, which facilitates secure communication between devices by encrypting data before transmission and decrypting at the time of receiving.

Working of the code: The script establishes a connection to a predefined MQTT broker address. It subscribes to a specified topic on the broker to receive incoming data. When sending data, the script encrypts the data frame using a secret key before publishing it to the MQTT broker. When receiving data, the script decrypts the encrypted data frame using the same secret key.

Dependencies: The code uses external libraries which are pandas for data manipulation, paho.mqtt.client for MQTT communication, and cryptography for encryption and decryption using the Fernet algorithm.

Encryption Key Generation: The code generates an encryption key dynamically for each execution using the `Fernet.generate\_key()` function. This key is used for both encryption and decryption of data frames.

# Security Analysis:

**A logo with a person running

Description automatically generatedREDBACK OPERTAIONS**

**VULNERABILITY FOUNDED**

**Use of Hardcoded Cryptographic Key - CWE 321**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Is this a re-tested Finding?** |
| Tushar Sharma | Cybersecurity | Secure Code Review | No |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Risk Rating**  
Impact: Significant  
Likelihood: Unlikely

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

This vulnerability can allow the attackers to access and misuse the encryption key which can further compromise the data.

**Location of vulnerability**

MQTT\_Data\_Frame\_Handler.py:16

**Evidence**

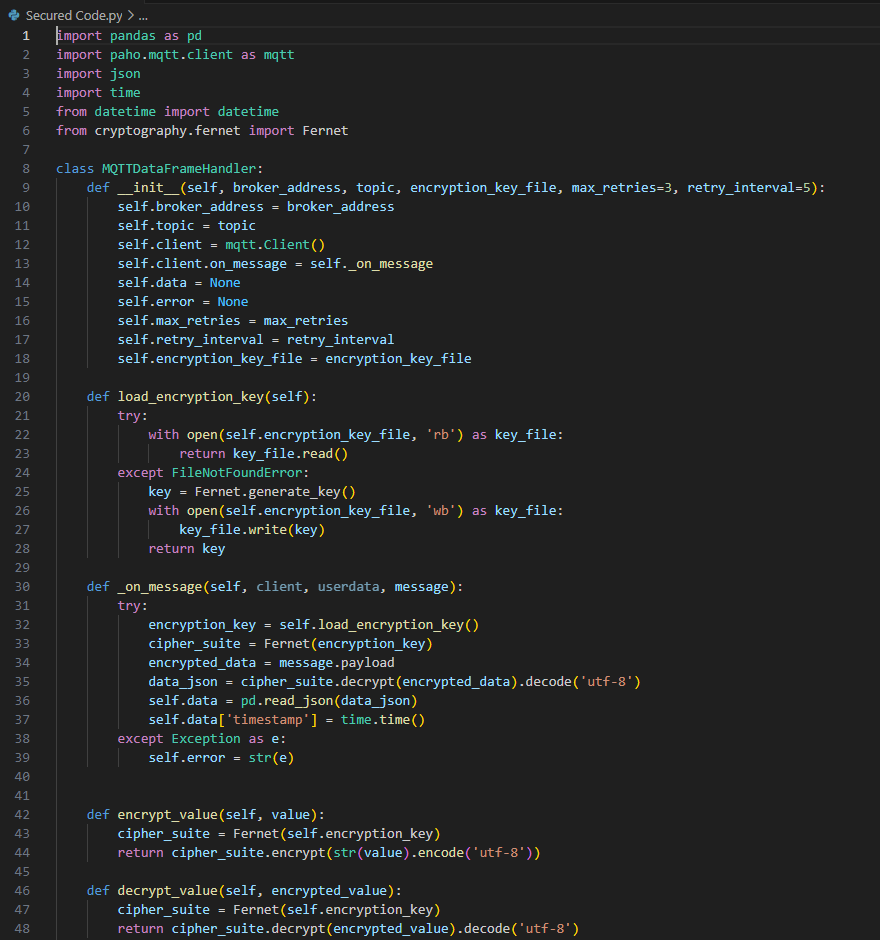
****

**Remediation Advice**

To mitigate the issue of storing sensitive information and hardcoding it within the script, we need to implement secure key management practices. Storing sensitive information like encryption keys directly within the script poses a significant security risk, as it can be easily accessed if the script is compromised. To address this risk, we can consider two approaches:

1. Use of Secure Key Vaults or Hardware Security Modules (HSMs): Secure key vaults provide a centralized and secure location for storing and managing encryption keys. Access to the key vault can be restricted using access controls and authentication mechanisms, ensuring that only authorized users can access the keys. Hardware security modules (HSMs) are designed to securely generate, store, and manage cryptographic keys. HSMs provide tamper-resistant protection for encryption keys and offer features such as key backup and audit logging, enhancing the security of key management.

2. Avoidance of Hardcoding Sensitive Information: Hardcoding sensitive information like encryption keys directly into the script increases the risk of exposure, as the keys can be easily extracted by attackers if they gain access to the script's source code. Instead of hardcoding sensitive information, retrieve the encryption key from secure external sources during runtime. This can be achieved by fetching the key from a secure key vault or using configuration files with restricted access permissions. By retrieving the encryption key dynamically during runtime, you reduce the risk of exposure and enhance the overall security posture of the application.



A screen shot of a computer

Description automatically generated

In this code, I have implemented secure key management: The encryption key is now stored in a separate file (`encryption\_key.key`) instead of being hardcoded which reduced the risk of exposure. Then, I have implemented the `load\_encryption\_key` method to load the encryption key from the file or generate a new key if the file doesn't exist.

**References**

<https://cheatsheetseries.owasp.org/cheatsheets/Key_Management_Cheat_Sheet.html>

<https://cwe.mitre.org/data/definitions/321.html>

<https://cwe.mitre.org/data/definitions/326.html>

**Contact Details**

Student Name: Tushar Sharma

Student Id: 222197136

Finding 2:

**A logo with a person running

Description automatically generatedREDBACK OPERTAIONS**

**VULNERABILITY FOUNDED**

**Improper Handling of Exception Conditions – CWE-755**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Is this a re-tested Finding?** |
| Tushar Sharma | Cybersecurity | Secure Code Review | No |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Risk Rating**  
Impact: Significant  
Likelihood: Moderate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

Insufficient error handling mechanisms present a notable risk to system reliability and availability. Runtime errors and exceptions may trigger unanticipated system failures, leading to service disruptions and impacting user experience. Inadequate error management complicates issue diagnosis and resolution, prolonging system downtime and adversely affecting business operations.

**Location of vulnerbility**

#### **MQTT\_data\_frame\_handler.py.**

**Evidence**

A computer code with white text

Description automatically generated



A screen shot of a computer code

Description automatically generated

**Remediation Advice**

Importance of Proper Exception Handling:

* Implementing robust error handling mechanisms, like try-except blocks, strengthens the code's resilience by effectively capturing and managing exceptions. These mechanisms provide detailed error messages or logs for easier troubleshooting, reducing the likelihood of successful Denial of Service (DoS) attacks that exploit error conditions.
* Integrating retry strategies for network operations enhances the code's ability to withstand transient errors, mitigating the risk of system crashes and resource exhaustion.
* Proper error handling also helps prevent information disclosure and injection attacks by safeguarding sensitive data and validating user inputs.

Exception handling was properly implemented on “\_on\_message” and “receive\_data”. It can also be implemented on further definitions such as:

* Encrypt\_value: Try-except block can be added in this definition, to ensure that no errors were encountered during encryption process.
* Create\_json\_payload: Exception handling can helped to data formatting has been done correctly, and if any error has been occurred, it can be captured and logged.

These updates enhance the code's robustness by providing comprehensive error handling mechanisms, ensuring reliable and resilient operation even in the face of unexpected issues.

**References**

<https://cwe.mitre.org/data/definitions/755.html>

<https://docs.python.org/3/tutorial/errors.html>

**Contact Details**

Student Name: Tushar Sharma

Student Id: 222197136

# Conclusion:

As part of the secure code review process, several critical changes were identified and implemented to enhance the security and reliability of the MQTT\_Data\_Frame\_Handler.py script. These changes address vulnerabilities related to the improper handling of encryption keys and exceptions, significantly reducing the risk of unauthorized access and system failures. By implementing secure key management practices and robust error handling mechanisms, the code now ensures the confidentiality, integrity, and availability of transmitted data. These improvements not only mitigate potential cyber threats but also contribute to the overall resilience and trustworthiness of the system.