Investigating and Mitigating Man-in-the-Middle (MITM) Attacks on a Simulated IoT Network Using MQTT

By

MD Kaium Hossen
ID:CSE2203027083

Sani Alam
ID:CSE2203027093

Md. Sazzad Hossen
ID:CSE2203027077

Faiyas Siddike Sabit
ID:CSE2203027014

Samia Arin
ID:CSE2203027147

Supervised by
Zarin Hadika

Submitted in the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY

APPROVAL

The thesis titled "**Investigating and Mitigating Man-in-the-Middle (MITM) Attacks on a Simulated IoT Network Using MQTT** " submitted by **MD Kaium Hossen** (CSE2203027083), **Sani Alam** (CSE2203027093), **Md.Sazzad Hossen** (CSE2203027077), **Faiyas Siddike Sabit** (CSE2203027014), **Samia Arin** (CSE2203027147) to the Department of Computer Science and Engineering. Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Zarin Hadika
Lecturer,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Zarin Hadika**, Lecturer, Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this thesis has been or is being submitted elsewhere for the award of any degree .

Countersigned

(Zarin Hadika)
Supervisor

Signature

MD Kaium Hossen
ID:CSE2203027083

Sani Alam
ID:CSE2203027093

Md.Sazzad Hossen
ID:CSE2203027077

Faiyas Siddike Sabit
ID:CSE220302704

Samia Arin
ID:CSE2203027147

ABSTRACT

As the Internet of Things (IoT) grows very fast, it relies heavily on the Message Queuing Telemetry Transport (MQTT) protocol for efficient communication. However, because it sends data in plain text by default, it is easy for hackers to attack the network. This paper looks at how a Man-in-the-Middle (MITM) attack affects a simulated MQTT-based IoT setup and tests how well different security steps work to block it.We built a controlled virtual environment using VirtualBox with three parts: Ubuntu (acting as the MQTT Broker/Publisher), Windows 11 (acting as the Subscriber using Paho MQTT), and Kali Linux (acting as the Attacker). We intercepted the network traffic using Ettercap for ARP spoofing, along with Wireshark and tcpdump to capture data packets. In the first test without any security, the attacker easily read the plain text data (temperature sensor readings) and changed the messages on the fly without the system noticing.To fix these problems, we added a flexible security setup to the Mosquitto broker. We enabled Transport Layer Security (TLS 1.2) encryption, Certificate Authority (CA) verification, and password-based login for clients.The test results showed that before securing it, normal MQTT had very weak security, even though it was fast. But after turning on TLS on port 8883 with user passwords, all the data was fully encrypted. This completely stopped hackers from reading the packets, changing the messages, or logging into the broker without permission. Our final comparison proves that mixing TLS encryption, digital certificates, and simple passwords gives a very high level of security. It successfully protects the data, keeps it accurate, and stops MITM attacks in IoT systems. Overall, this study offers a simple, practical framework that you can easily use to secure low-power IoT devices.

Keywords: Demilitarized Zone, Cyber-Physical Isolation, Common Vulnerabilities and Exposures, Mutual Transport Layer Security, Cryptographic Algorithms, Packet Sniffing Programs, Irreversible Signatures, Quantum-Vulnerability Analyzer.

ACKNOWLEDGMENT

At the very beginning, we would like to express our deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Zarin Hadika**, Lecturer, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our thesis.

We would like to convey our special gratitude to **Prof. Bulbul Ahamed**, Head, Department of Computer Science and Engineering, Sonargaon University, for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to our parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

ACL	Access Control List
AES	Advanced Encryption Standard
AMQP	Advanced Message Queuing Protocol
ARP	Address Resolution Protocol
BERT	Bidirectional Encoder Representations from Transformer
CA	Certificate Authority
CoAP	Constrained Application Protocol
CPU	Control Processing Unit
DHE	Ephemeral Diffie-Hellman
DNS	Domain Name System
DPWS	Devices Profile for Web Services
ECC	Elliptic Curve Cryptography
ECDH	Ephemeral Curve Diffie-Hellman
ECDHE	Ephemeral Elliptic Curve Diffie-Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GAN	Generative Adversarial Network
HMAC	Hash-based Message Authentication Code
HTTP	Hyper Text Transfer Protocol
ICS	Industrial Control System
IDS	Intrusion Detection System
IoC	Internet of Computers
IoRT	Internet of Robotics
IoT	Internet of Things
IP	Internet Protocol
KHNP	Korean Hydro and Nuclear Power
MAC	Media Access Control
MitM	Man-in-the-Middle
ML	Machine Learning
MQTT	Message Queing Telemetry Transport
M2M	Machine-to-Machine
NIST	National Institute of Standard and Technology
NPP	Nuclear Power Plant
OAuth	Open Authorization
OT	Operational Technology
PKC	Public Key Cryptography
PRF	Pseudo Random Function
REST	Representational State Transfer
RSA	Revest-Shamir-Adleman
SSL	Secure Seckets Layer
SVM	Support Vector Machine
TLS	Transport Layer Security
TCP	Transmission Control Protocol
VPN	Virtual Private Network
XMPP	Extensible Messaging and Presence Protocol

TABLE CONTENTS

Title **Page No.**

DECLARATION iii
ABSTRACT iv
ACKNOWLEDGEMENT v
LIST OF ABBREVIATION..... vi

CHAPTER 1 1-5
INTRODUCTION
1.1.Background of the study 1
1.2. Problem Statement 2
1.3 Objectives of the study 3
1.4 Motivation 3
1.5 Contribution of the work..... 5
1.6 Organization of the thesis 5

CHAPTER 2
LITERATURE REVIEW 6 - 10
2.1. Introduction to IoT and Protocol Landscape 6
2.2 MQTT 7
2.3. Man in the middle(MitM) attack in MQTT environment 7
2.4. MQTT security mechanisms 8
2.5. Transport Layer Security 9
2.6. Research Gap 10

CHAPTER 3 11-24
METHODOLOGY
3.1. System Architecture 11
3.2. Experimental setup..... 11
3.2.1 Hardware Requirements 11
3.2.2 Software Requirements..... 12
3.2.3. Network Configuration..... 12
3.3. Attack and Mitigation Procedure..... 12
3.3.1. MQTT Communication Setup..... 13
3.3.2. MitM Attack Execution 13
3.3.2.1. ARP Spoofing Using Ettercap..... 13
3.3.2.2. Packet Capture Using TCPdump/Wireshark..... 14
3.3.3. Mitigation Implementation..... 15
3.3.3.1. TLS Implementation & Handshake..... 15
3.4. Publisher Source Code 20
3.5. Subscriber Source Code..... 21
3.6. Certificate Varification 22
3.7. Performance Matrix 23
3.7.1. Measuring Latency(ms)..... 23

3.7.2. Measuring Average(Mean) Latency..... 23
3.7.3.Standerd Deviation (SD)..... 24
3.7.4.Variance..... 24
3.7.5.Percentage Increase in Latency..... 25

CHAPTER 4 25-31
RESULT ANALYSIS
4.1. MitM attack result 26
4.2. Mitigation Results 28
4.3. Compare security between TLS and non_TLS..... 29
4.4. Discussion. 29
4.4.1. Analysis of attack effectiveness 29
4.4.2.Analysis of security improvements 30
4.4.3 Comparative Analysis 30
4.5. Performance Measure 31

CHAPTER 5
CONCLUSION 32-33
5.1. Summary of the Research 32
5.2. Key Findings and Performance Evaluation..... 32

CHAPTER 6 34-35
LIMITATIOON
6.1. Limitations of the experiment..... 34
6.2. Simulated Environment..... 34
6.3. Limited Network Scale..... 34
6.4. Controlled Network Conditions..... 34
6.5. Latency Measurement Accuracy..... 34
6.6. Limited Attack Scenarios..... 34
6.7. Security Mechanisms Evaluated..... 35
6.8. Certificate Management..... 35

CHAPTER 7
FUTURE WORK 36
7.1. Hardware based Implementation 36
7.2. Intrusion Detection Systems (IDS) 36
7.3. Advanced Attack Scenarios 36
7.4. Lightweight Payload Encryption 36

REFERENCES 37

LIST OF FIGURES

Figure No.	Title	Page No
Fig. 1	Secure MQTT over TLS	4
Fig. 2	Man in the Middle	13
Fig. 3	ARP Spoofing	14
Fig. 4	MitM using Wireshark	15
Fig. 5	TLS handshake in MQTT communication	16
Fig. 6	use_tls TRUE	18
Fig. 7	use_tls FALSE	18
Fig. 8	After TLS message encrypted	22
Fig. 9	Certificate varification	23
Fig. 10	MQTT without TLS	25
Fig. 11	MQTT without TLS	25
Fig. 12	MitM before mitigation	27
Fig. 13	Publisher message log	27
Fig. 14	Subscriber message log	27
Fig. 15	ARP Spoofing	31

LIST OF TABLE

Table No.	Title	Page No.
Table 1	Network Configuration	12
Table 2	Security comparison	29
Table 3	Comparative Analysis	30
Table 4	Latency comparison	31

CHAPTER 1
INTRODUCTION

1.1 Background of the Study

There are many cyber threats around the globe.The Man-in-the-Middle (MitM) attack is one of them also most dangerous.These attacks can interfere with communication between two trusted parties without either party realizing it. This kind of attack can break the confidentiality and integrity of the data being sent across a network. Since internet-based communication is now a part of banking, business operations, government services and other important functions keeping data safe during transmission has become more crucial, than ever. The serious consequences of these attacks show how important it is to have security measures in place to protect communication systems and the data they handle.

This threat was demonstrated by the famous Iranian nuclear facility cyberattack in 2010, also known as the Stuxnet worm[1]. Stuxnet consisted of various types of malicious software that targeted the centrifuges' speed control system, causing their unexpected acceleration or deceleration. This resulted in the destruction of about 1000 centrifuges and a significant delay in Iran's nuclear program. Moreover, there are reports of other cyber-attacks in the past at nuclear facilities happened at Kudankulam in India (2019)[2], KHNP in South Korea (2014)[3] and the Wolf Creek and Davis Besse plants in the United States (2017)[4].

Nevertheless, with the help of the developed security system, it will be possible to ensure the safe functioning of nuclear facilities and protect them from potential cyber threats. It is designed to provide infrastructure protection and reduce the risks of such incidents, which will minimize the danger to the population and the environment. This security system was developed as a result of our research, aimed at increasing the level of industrial security with minimal data exchange via the Internet.

1.2 Problem Statement

Despite being one of the most used protocols for modern Industrial Internet of Things (IIoT), Message Queuing Telemetry Transport (MQTT) has one major security related design flaw that makes all of its data transmissions insecure by default. With the possibility to intercept and alter crucial data packets, attackers can utilize various Man-in-the-Middle (MitM) techniques to compromise the integrity of MQTT-based systems, which are commonly used in Nuclear Power Plants (NPP).For instance, using Address Resolution Protocol (ARP) poisoning, an attacker can redirect all sensor data packets to his device where he can modify the content before the information is delivered to the destination.In practice, this would allow him to change the temperature of a reactor coolant to a lower value, which would trick the operator that everything is working fine, whereas in reality, the reactor might be overheated.

Firstly, the problem of computational complexity of existing security measures cannot be ignored, particularly when it comes to the IIoT application of the MQTT protocol. Since most of the sensors and controllers are usually operated via low-end devices, deploying a resource-demanding security protocol may result in a significant overhead that negatively impacts the performance of such devices.

Secondly, another critical issue related to the utilization of current security practices in NPPs is the possibility of operational delays caused by the necessity to process and analyze additional data transmitted between MQTT agents. At present, such delays are unacceptable in the majority of critical infrastructures, including NPPs, where a short response time is often the only thing preventing a catastrophe. As a result, even though TLS, mTLS, and similar protocols offer a substantial level of protection, their utilization in delay-sensitive areas may lead to unacceptable levels of system instability.

In conclusion, there is a critical need for an effective low-bandwidth security protocol that will be able to prevent the majority of MitM attacks without adding considerable overhead to the system. The failure to implement such a standard can lead to severe consequences for all MQTT-based systems, particularly those operating in the IIoT environment, as the likelihood of successful attacks will remain exceedingly high.

1.3 Objectives of the study

The purpose of this study is to design and evaluate a novel low-complexity secure system for detecting and preventing Man-in-the-middle (MitM) attacks in a simulated Internet of Things (IoT) network using the Message Queuing Telemetry Transport (MQTT) protocol with the use of low power devices.
The objectives of this research include:

1) To expedite the communication process within existing secure systems while enabling packet transmission with light packet overhead.

2) To identify Man-in-the-middle attacks by experimenting and analyzing various approaches, including Address Resolution Protocol (ARP) poisoning and packet injection, to understand the extent to which attackers could manipulate real-time sensor data and actuation

3) To design a light packet overhead security tool or algorithm that detects and prevents unauthorized data alterations at the network layer during data transmission. The objective also includes demonstrating system integrity during serious scenarios by ensuring that the designed system can prevent physical damage to critical infrastructure, which would have occurred due to cyber-attack, and safeguarding human lives.

4) To ensure low data consumption and utilization to maintain low bandwidth usage while achieving a robust and efficient system that supports low-power

hardware without significantly compromising the speed of the targeted network.

5) To experimentally evaluate performance and compare the results of traditional cyber security approaches such as Transport Layer Security (TLS)/Mutual TLS (mTLS) and the proposed solution based on network latency analysis, packet delivery ratios, CPU utilization, and detection rates.

1.4 Motivation

Even though there are security risks with industrial systems making sure that MQTT communication is secure is still hard because security needs to be put in place without making the communication slower. In industrial IoT setups data has to be sent and received all the time and in a reliable way but many devices have very little processing power and network ability. This means there is a need to look at how MQTT communication can be kept safe from Man-in-the-Middle attacks without making the network performance worse. Because of this problem the reason, for this research is focused on three challenges.The main motivation behind this research comes from three major challenges in today's industrial IoT setup:

1) We notice that MQTT does not encrypt data, MQTT sends everything as text over TCP. That leaves networks very vulnerable to MitM attacks. In a factory or power plant a hacker using a MitM attack can quietly. Alter sensor readings such as temperature or pressure without setting off any alarms. As a result operators see no warning. Remain completely unaware of real dangers.
2) In the past critical facilities kept their networks disconnected from the internet a practice known as air- gapped to keep safe. But today automation requires data streaming so systems can no longer stay truly isolated. The real danger was shown by the Stuxnet attack on Iran's Natanz nuclear plant and by more recent cyberattacks in India, South Korea and the United States. In high risk places such as plants, where cooling systems must run nonstop altering data for even a few minutes can upset the reactor core. Altering data for a few minutes can trigger massive explosions, severe environmental damage and loss of human life.

3) Security methods such as TLS or mTLS can stop MitM attacks. They need too much processing power and consume a lot of internet data or bandwidth. Most industrial IoT devices, such as sensors and controllers have very limited CPU power and memory. Forcing encryption, on these small devices slows communication creating delays that safety critical systems cannot afford.

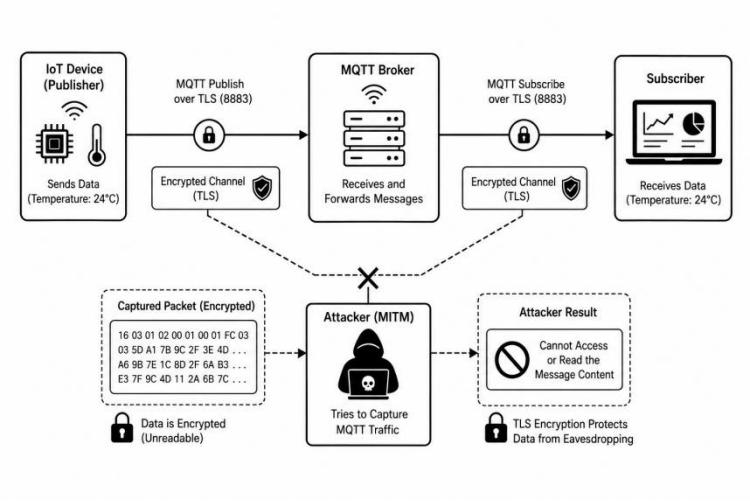


Fig - 1 : Secure MQTT over TLS

Figure 1 shows how MQTT messages are sent securely between an IoT publisher, an MQTT broker and a subscriber using Transport Layer Security. At the first, the IoT device works as a publisher and generates data as temperature reading of 24 degree Celsius. The publisher construct a TLS-secured MQTT connection with the broker on port 8883. This TLS-secured MQTT connection guarantees that the MQTT message is encrypted before it leaves the device. The MQTT broker gets the encrypted message. Sends it to the subscribed client through another TLS-secured MQTT connection. Someone trying to break into the communication between the devices might try to catch the network traffic as part of a Man-in-the-Middle attack.The packets that are caught have encrypted hexadecimal data instead of the original MQTT message. This means the person trying to break in cannot read the temperature value or any other part of the message. The X symbol in the figure shows that unauthorized people cannot get into the protected communication channel. So the figure shows that using TLS with MQTT keeps communication safe from packet interception and eavesdropping by making sure the data stays private.

1.5 Contributions of the Work

The primary contributions of this thesis toward advancing the state-of-the-art in Industrial IoT (IIoT) security and protecting safety critical infrastructures are summarized as follows:

1) We developed a realistic virtual IIoT network that is designed to represent critical industrial control systems. This provides a scalable testbed for evaluating MQTT vulnerabilities under different cyber physical conditions.

2) We developed a lightweight security algorithm for MQTT brokers that can detect and stop unauthorized data tampering in real time. This helps prevent attackers from hiding important changes in operational data.

3) We proposed a security framework which is designed to use minimal packet and computational resources. This helps reduce the trade off between security,

resource usage, and latency in constrained edge devices such as sensors and actuators.

4) We showed that the proposed defense mechanism can maintain reliable operational telemetry for safety critical systems such as nuclear plant cooling fans.This can help reduce the risk of serious hardware failures caused by cyberattacks.

5) We evaluated the proposed system against standard security protocols such as TLS or mTLS using important performance measures, including network latency, CPU usage, and detection accuracy. This comparison helps show the effectiveness and resource efficiency of the proposed approach.

6) We proposed a method that provide a detailed packet level analysis of Man-in-the-Middle (MitM) attacks, especially ARP poisoning and telemetry injection to this research.The analysis shows how these attacks can bypass traditional firewall mechanisms and manipulate sensor data.

1.6 Organization of the Thesis

This thesis is organized into seven chapters. Chapter 1 introduces the background, motivation, problem statement, objectives, scope, and significance of the study. Chapter 2 reviews the existing literature on IoT, MQTT, Man-in-the-Middle attacks, and MQTT security datasets. Chapter 3 describes the methodology. Chapter 4 presents the results and analysis. Chapter 5 concludes the study. Chapter 6 discusses the limitations, and Chapter 7 outlines the future work.

CHAPTER 2
LITERATURE REVIEW

2.1. Introduction to IoT & Protocol Landscape

Choudhary (2024) examines existing simulators and their functionalities, so that researchers can easily select the simulator as per their targeted objectives. In addition, major issues are highlighted here that should be addressed by the scientific community. The significance of this work is to understand fundamentals of IoT architecture as well as a complete review in order to delve deeper into the difficulties and devise appropriate solutions [5].

Muheden et al.(2024) introduces an integrated energy approach, enhances existing standards, and establishes a shared basis for multidisciplinary planning. It also introduces new semantic network ontologies to provide a comprehensive framework for solving resource-related challenges. This new approach aims to fill the gaps in current standards and create an integrated environment for multi-stakeholder collaboration, using a semantic web ontology for communication and improved decision making in energy systems Provides information integrated, including various forms of smart cities With flexibility for flexibility and inclusion in the energy industry, can accommodate the specific characteristics and needs of various smart city applications In this study, computing -physical system (CPS), software-defined network (SDN), internet (IoT).), and analyze how smart cities are connected. CPS combines physical channels with electronic systems to provide increased network management efficiency and flexibility. SDN improves dynamic capacity and flexibility, while IoT is more connected for real-time data exchange and decision-making[6].

Bellman et al. (2019) observes and analyzes that the history of IoT. It has provided a substantial body of topics to look back on: IoT's evolution, products, and major security incidents including the largest botnet ever witnessed. Unique to IoT, its architecture, interaction design, and scale make its many issues distinct from those in the Internet of Computers (IoC). Its perceptions, understandings, and definitions have evolved over time, thus requiring an updated focus from the perspective of security and cyber-physical safety. They take a fresh look at challenges and opportunities in IoT security, the characteristics that uniquely distinguish it from the IoC, and identify security-related questions that they raise. Their aim is to provide an up-to-date view of the IoT security landscape and technical security issues to help guide both existing and especially new researchers looking for challenging open problems that remain largely unaddressed[7].

2.2. MQTT

Hintaw et al. (2023) examines and explains Representational State Transfer (REST) API method, which is based on web communication protocol, Hyper-Text Transfer Protocol (HTTP), and is widely used in IoT messaging. Some of these protocols are DPWS, XMPP, MQTT, COaP, AMQP. Among these protocols, MQTT is the most preferred protocol and is expected to be the de facto messaging IoT standard. MQTT uses a publisher/subscriber model to facilitate messaging between devices making messaging lightweight. Nevertheless, there are a number of security issues due to the design of the protocol itself. Some of the issues are denial of service, identity spoofing, information disclosure, elevation of privileges and data tampering. These issues can be caused by both internal and external perpetrators. Researchers have proposed various security techniques and mechanisms to address these issues. Incorporation of security has added processing overhead to the devices and this will have a bearing on IoT devices that are powered by a battery [8].

Husnain et al. (2022) shows preventing MQTT vulnerabilities by using an unique system .In this paper, they have designed and developed an MQTT parsing engine that can be integrated with network-based IDS as an initial layer for extensive checking against IoT protocol vulnerabilities and improper usage through a rigorous validation of packet fields during the packet-parsing stage. In addition, we evaluate the performance of the proposed solution across different reported vulnerabilities. The experimental results demonstrate the effectiveness of the proposed solution for detecting and preventing the exploitation of vulnerabilities on IoT protocols[9].

Nebbione and Calzarossa (2020) analyze the security and IoT application layers. They offers a comprehensive survey of application layer protocol security by presenting the main challenges and findings. More specifically, the paper focuses on the most popular protocols devised in IoT environments for messaging/data sharing and for service discovery. The main threats of these protocols as well as the Common Vulnerabilities and Exposures (CVE) for their products and services are analyzed and discussed in detail. Good practices and measures that can be adopted to mitigate threats and attacks are also investigated. Their findings indicate that ensuring security at the application layer is very challenging. IoT devices are exposed to numerous security risks due to lack of appropriate security services in the protocols as well as to vulnerabilities or incorrect configuration of the products and services being deployed. Moreover, the constrained capabilities of these devices affect the types of security services that can be implemented[10].

2.3. Man-in-the-Middle (MITM) Attacks in MQTT Environments

Henry Wong and Tie Luo(2020) introduce a new scheme of a Man-in-the-Middle (MITM) attack on IoT devices that utilize the Message Queuing Telemetry Transport (MQTT) protocol for communications. This attack scheme consists of an MQTT Parser that is created to dissect and alter MQTT messages at the bit level, and a novel BERT-based adversarial model that generates malicious messages using an approach inspired by GAN. They present the design of this

attack in order to show how a sophisticated attack could lead to serious damage that is difficult for typical security defense mechanisms to detect. They set up a test-bed using IoT hardware and software including Raspberry Pi, WiFi Pineapple, Mosquitto, etc. to conduct experiments. I this paper they show that their designed attack scheme successfully evades logistic regression, random forest, K-nearest neighbor, and support vector machine (SVM) based anomaly detection models. Multi-Layer Perceptron fares better against our model, but such use of deep neural networks on typical IoT devices is rather restricted due to the computation cost [11].

Simsek and Atilgan (2023) present the possible attacks on availability of middleware layer messaging protocols. In the research, a comprehensive case study was carried out, especially focusing on the MQTT (Message Queuing Telemetry Transport) protocol.They performed Man-in-the-Middle (MitM), Denial of Service (DoS) and Brute-Force attacks in our experimental environment. The effects and results of the attacks made in cases where the connection to the MQTT protocol is made with a user name and password, and when the user name and password are not used are examined. The results of the attacks that emerged in the different scenarios created were evaluated and the precautions to be taken to protect against the attacks were discussed[12].

Sultan et al.(2022) analyzes the different Machine Learning Algorithms for MQTT base IoT device . In this paper they have implemented five different machine learning model on an open-source dataset and evaluated different parameters like accuracy, precision, recall, F1 score and most importantly training time and test time because most of IoT network are hosted on resource constrained devices like Raspberry Pi [13].

2.4. MQTT Security Mechanisms and Countermeasures

Harsha et al. (2018) analyzes and examin a standardized, light-weight, session layer protocol with publish/subscribe architecture widely used for messaging and information exchange among IoT devices is the MQTT (MQ Telemetry Transport) protocol. In this paper they identify various security loopholes in MQTT, using Shodan API and implementing an experimental setup on a Raspberry Pi as an MQTT Broker and python programs as publisher/subscriber clients. The experimental results with respect to the security issues in this protocol at packet and topic levels were studied and the corresponding security measures, consisting of authentication and authorization techniques (ACLs) were implemented. As a result, the Broker was then found to be immune to such attacks. This paper is a concise study of security inconsistencies in MQTT and its countermeasures[14].

Altulaihan et al.(2022) interprets the cybersecurity threats on IoT. We know that Cybersecurity threats are unique to the Internet of Things, which has unique characteristics and limitations. In consideration of this, a variety of threats and attacks are being launched daily against IoT. Therefore, it is important to identify these types of threats and find solutions to mitigate their risks. Therefore, in this paper, they reviewed and identified the most common threats in the IoT environment, and they classified these threats based on three layers of IoT architecture. In addition, they discussed the most common countermeasures to

control the IoT threats and mitigation techniques that can be used to mitigate these threats by reviewing the related publications, as well as analyzing the popular application-layer protocols employed in IoT environments and their security risks and challenges[15].

Mileva et al.(2021) competitively analyze about this matter. Recently its 5.0 version has been introduced and ensuring that it is capable of providing services in a secure manner is of great importance. It must be noted that holistic security analysis should also evaluate protocol's susceptibility to network covert channels. they present a systematic overview of potential data hiding techniques that can be applied to MQTT 5.0. They are especially focusing on network covert channels that, in order to exchange secrets, exploit characteristic features of this MQTT version. Lastly they develop proof-of-concept implementations of the chosen data hiding techniques and conduct their performance evaluation in order to assess their feasibility in practical setups[16].

2.5 Transport Layer Security (TLS/SSL & mTLS)

Delgado (2026) examines and presents a multi-surface framework for post-quantum TLS observability. The framework separates passive session evidence, active probing, certificate-chain evidence, and registry knowledge, and maps them onto measurement planes for session behavior, key establishment, endpoint capability, authentication, lifecycle, observability, and policy. He instantiate it as a reproducible artifact with schema-enforced observations and results, versioned registries, auditable inference rules, stress contracts, and baseline adapters. They evaluate the framework on 29 controlled scenarios spanning TLS 1.2 and TLS 1.3, classical and hybrid key establishment, mutual TLS, resumption, HelloRetryRequest, truncation, fragmentation and coalescing, temporal drift, IPv6, and chain-depth variation. Passive evidence closes session-level planes, active probing establishes capability lower bounds, and multi-surface evidence closes the full measurement object while preserving uncertainty and contradiction when required. Against an inherited TLS quantum-vulnerability analyzer, the baseline detects 2 of 29 runs and 0 of 23 TLS 1.3 runs. In a stratified public campaign over 1000 targets and 2000 fresh probes, the framework completes 1971 handshakes, collects 1368 chain artifacts, confirms hybrid capability for 310 targets, and identifies 310 cases where endpoint capability exceeds what any single classical session view reveals[17].

Fernando (2024) analyzes and examine on it . He decided that it explores widely adopted security mechanisms such as OAuth 2.0, OpenID Connect, API gateways, and token-based authentication, along with encryption protocols like TLS to ensure secure communication. The role of API management platforms and service meshes in enforcing security policies and monitoring API traffic is also discussed. Furthermore, the study highlights the importance of integrating security into the software development lifecycle through DevSecOps practices, enabling continuous testing, vulnerability assessment, and automated threat detection. Emerging technologies such as artificial intelligence and machine learning are examined for their potential to enhance API security through anomaly detection and predictive threat analysis. The findings emphasize that a layered, proactive,

and policy-driven approach is essential to securing APIs in distributed systems, ensuring data integrity, confidentiality, and system resilience[18].

Măriș and Tătar (2025) examine and take decision that the proposed method is designed for implementation within the operational technology (OT) domain, enforcing security protocols that ensure both data encryption and network isolation. The primary objective is to establish a dedicated operational environment encompassing a command and control center where the robotic network server resides, alongside real-time data storage from network clients and remote control of field-deployed mobile robots. Building on this infrastructure, operational strategies are developed to enable an efficient robotic response in critical situations. By leveraging remote robotic networks, significant benefits are achieved in terms of personnel safety and mission efficiency, minimizing response time and reducing the risk of injury to human operators during hazardous interventions. Unlike generic IoT or IoRT systems, this work focuses on secure robotic integration within segmented OT infrastructures. The technologies employed create a synergistic system that ensures data integrity, encryption, and safe user interaction through a web-based interface. Additionally, the system includes mobile robots and a read-only application positioned within a demilitarized zone (DMZ), allowing for secure data monitoring without granting control access to the robotic network, thus enabling cyber-physical isolation and auditability [19].

2.6. Research Gap

Most of the studies focuses on simple theoretical threats in basic smart home setups, ignoring how real-world data attacks affect critical systems like power plants. Additionally, standard security tools like TLS/SSL are widely recommended to block attacks, but previous research ignores how these heavy security processes slow down small, low-power IoT devices. Furthermore, many papers suggest complex theoretical models without testing real-world factors like network speed, delay and data usage.

To fill these gaps, this study tests real-time attacks in a simulated network and measures the exact speed and performance impact of TLS 1.2 security, providing a practical and lightweight protection system for IoT networks.

CHAPTER 3
METDOLOGY

The methodology emphasizes the experimental setup, network configuration, attack implementation procedures and mitigation techniques used throughout the study.

3.1 System Architecture

A simulated virtual environment was established using VirtualBox, where Ubuntu Linux worked as the MQTT broker and publisher, Windows 11 worked as the subscriber and Kali Linux served as the attacking machine. The MQTT communication was accelerated using the Mosquitto broker, along side network monitoring and attack execution was performed using tools such as Wireshark and Ettercap.
The research is divided into two phases. In the first phase, a man-in-the-middle attack is launch through ARP spoofing to intercept and analyze MQTT traffic transmit between the publisher and subscriber. In the second phase, security mechanisms including Transport Layer Security (TLS) and authentication techniques are implemented to mitigate the identified vulnerabilities. The effectiveness of these mitigation measures is evaluate by comparing network behaviour and communication security before and after implementation.

3.2 Experimental setup

The methodology described in this chapter provides a systematic approach for assessing MQTT security and determining the effectiveness of countermeasures against MITM attacks in IoT environments.

3.2.1 Hardware Requirements

The proposed MITM attack on the simulated IoT network requires both hardware and software components to establish and monitor the experimental environment. The host machine performs the MITM attack and ARP spoofing, while virtual machines simulate the MQTT broker, publisher and subscriber. Various security and networking tools, including Kali Linux, Mosquitto, Wireshark, tcpdump, Ettercap and Paho MQTT are used to conduct and analyze the experiment.

3.2.2 Software Requirements

These are the following software that are implemented in this work:

1) Kali Linux (Attacker) - is used to perform the MITM attack, intercept MQTT communication and analyze the captured traffic.

2) Ubuntu (MQTT Broker/Publisher) - hosts the Mosquitto MQTT broker and publishes simulated IoT sensor data, such as temperature values to the MQTT network.

3) Windows 11 (Subscriber) – is acts as the MQTT subscriber, receiving the sensor data from the broker and displaying the transmitted messages and latency.

4) Mosquitto Broker - provides the MQTT communication infrastructure, receiving messages from the publisher and forwarding them to subscribed clients. It's also configured for both normal MQTT and secured MQTT communication using TLS.

5) Wireshark / tcpdump - are used to capture, inspect, and analyze MQTT network packets, helping demonstrate plaintext traffic before mitigation and encrypted traffic after TLS implementation

6) Ettercap - is used from the Kali attacker machine to perform the MITM attack, positioning the attacker between the MQTT communicating devices and attempting to intercept their network traffic.

7) Python MQTT Client (Paho MQTT) - is used to develop the publisher and subscriber programs in Python, generate sensor data, communicate with the Mosquitto broker and measure communication latency.

3.2.3 Network Configuration

We set up the VirtualBox network adapter in Bridged Adapter mode so the virtual machines can stay on the network as our host machine. Using the Bridged Adapter mode means the virtual machines can talk directly to devices, on the network because the virtual machines have their own IP addresses. This setup really helps us to make the network communication feel real while we work on the MITM attack experiment.
Using NAT instead of Bridge adapter have a bit problem to establish connection between broker and subscriber. Shown in the configuration of the table 1, 192.168.0.106 is Kali Linux acts as attacker machine, 192.168.0.102 is Ubuntu worked as broker and iot device like sensor and 192.168.0.101 is windows 11 is a listener or subscriber.

Device	IP Address	Role
Kali linux	192.168.0.106	Attacker
Ubuntu	192.168.0.102	Broker/Publisher
Windows 11	192.168.0.101	Subscriber

Table 1 : Network Configuration

3.3 Attack and Mitigation Procedure

A Man-in-the-Middle (MITM) attack is simulated by intercepting MQTT communication between IoT devices and the broker to demonstrate unauthorised message capture and manipulation. The attack is mitigated using TLS/SSL encryption, certificate based authentication and access control mechanisms, ensuring secure and reliable MQTT communication.

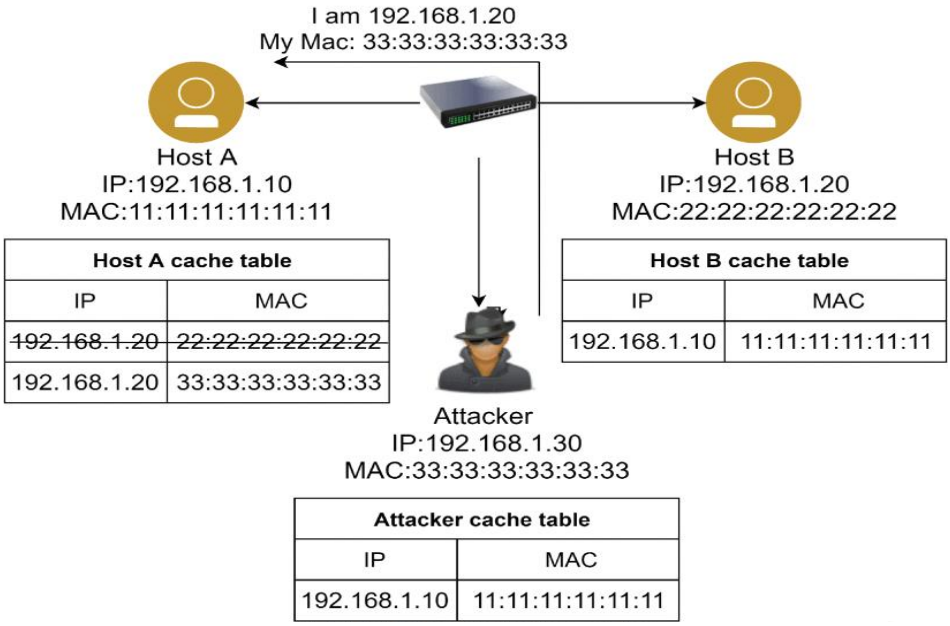


Fig - 2 : Man in the Middle

3.3.1. MQTT Communication Setup

The MQTT communication setup consists of IoT clients (publishers and subscribers) connected to an MQTT broker, where data is exchanged using the publish/subscribe model. This configuration enables efficient, lightweight and real time communication between IoT devices within the simulated network. There are two common python oriented MQTT tools to communicate between publisher and subscriber.

1) paho mqtt : is installed in windows 11 to run subscriber.py
2) mosquitto : create an MQTT broker or communication hub to receive or send the data according to the topic.

3.3.2. MitM Attack Execution

MITM attacks occur when an attacker secretly intercepts and modifies communication between two parties without their knowledge. The assailant situates himself between the transmitter and the recipient, therefore fabricating the appearance of direct communication. This allows them to intercept, obtain confidential information or alter the transmitted content [20],[21],[22].

3.3.2.1 ARP spoofing using Ettercap

ARP has been designed to work effectively under regular circumstances, not to deal with malicious hosts. In a normal case if host A needs to communicate with host B in the same network and the MAC address of host B is unknown to host A. Host A will send an ARP broadcast request to all devices in the network to receive only one ARP unicast reply from host B, and this ARP reply contains the MAC address of host B. Each host has a cache table that recorded all IP and MAC pairs for all hosts' contacts with it before. Fig. 2 and Fif. 3 illustrates an example of three devices cache tables after sending a broadcast ARP request from host A and receiving a unicast ARP reply from host B[23].

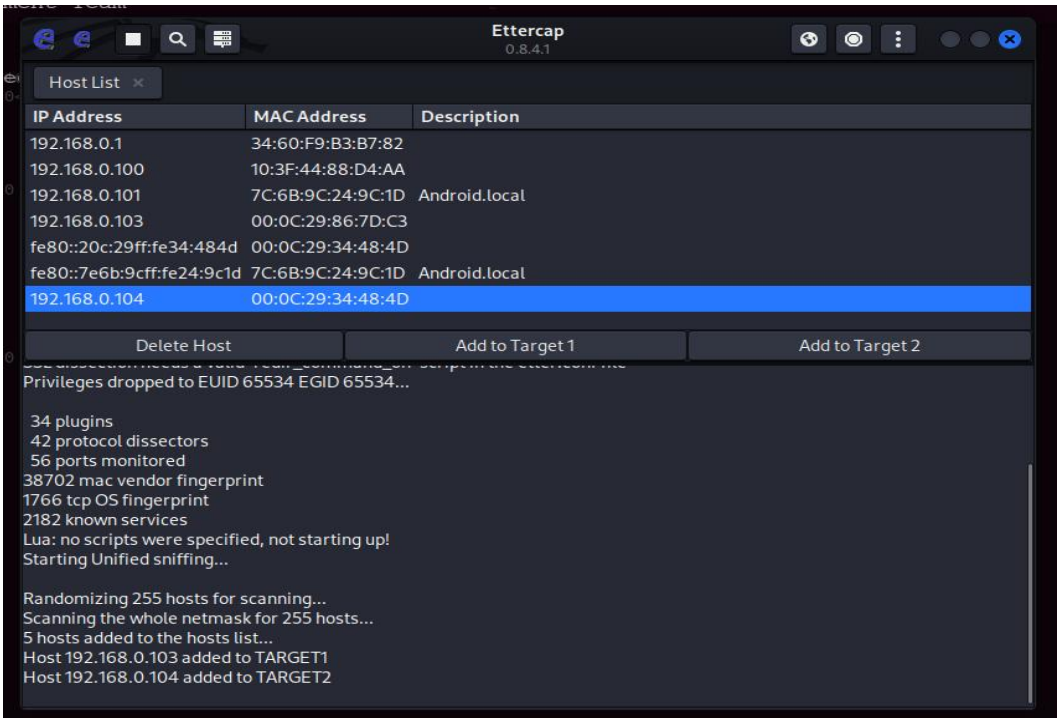


Fig - 3 : ARP Spoofing

3.3.2.2 Packet capture using Wireshark/tcpdump

There are various types of packet sniffing programs, including both free and commercial ones. Each program is designed with different goals in mind. A few popular packet analysis programs are tcpdump, OmniPeek and Wireshark. Tcpdump is a command line program. Wireshark have graphical user interfaces (GUIs). Wireshark is the world's foremost and widely used network protocol analyzer. It lets us see what's happening in our network at a microscopic level. Wireshark offers several benefits that make it appealing for everyday use. It is aimed at both the journeyman and the expert packet analyst and offers a variety of features to entice each [24].

Fig. 4 shows non-TLS MQTT communication between the MQTT broker or publisher at 192.168.0.104 and the subscriber at 192.168.0.101 over TCP port 1883. The captured packets are identified directly as MQTT Publish Message

packets, with the topic iot/temperature clearly visible in the packet details. Before TLS implementation an attacker using wireshark packet sniffing tool can easily intercept MQTT traffic. The attacker can directly read the application-layer data, also read the MQTT topic and the temperature value. We notice that this makes the communication vulnerable. The vulnerability allows packet interception and information disclosure. A MitM or passive sniffing attack can easily steal this data.

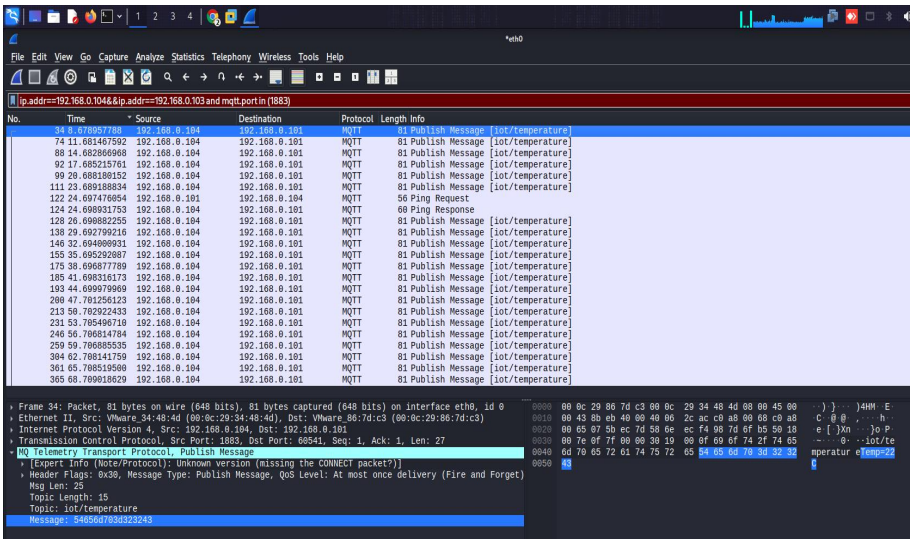


Fig -4 : MitM using Wireshark

3.3.3. Mitigation Implementation

TLS provides endpoint authentication. Ensures message integrity. Implementing TLS over MQTT based iot network is necessary because it directly handles the security weaknesses exposed during the MitM attack. When an MQTT connection establish over an unsecured connection, an attacker may intercepts the traffic and also can read plain text MQTT topic, sensors data, usernames, passwords and others sensitive information. TLS implementation encrypts the communication between the MQTT clients and the broker, so even if the attacker successfully intercepts the packet, captured packet cannot be read as meaningful MQTT data.

3.3.3.1. TLS implementation and handshake

Transport Layer Security (TLS) [25] enables end-to-end secured communication between two machines. TLS uses cryptography for providing data and end-point authenticity, and data confidentiality. TLS is a continuation of the Secure Socket Layer (SSL) protocol and the current version is TLS 1.2. Technically, TLS is a protocol suite operating on top of the TCP protocol. While establishing a connection, the handshake protocol allows the two parties to agree on how they authenticate each other. Also during this phase the cipher spec protocol enables selecting the cryptographic primitives used by the application data protocol for

achieving data authenticity and confidentiality during actual data transmission[26].

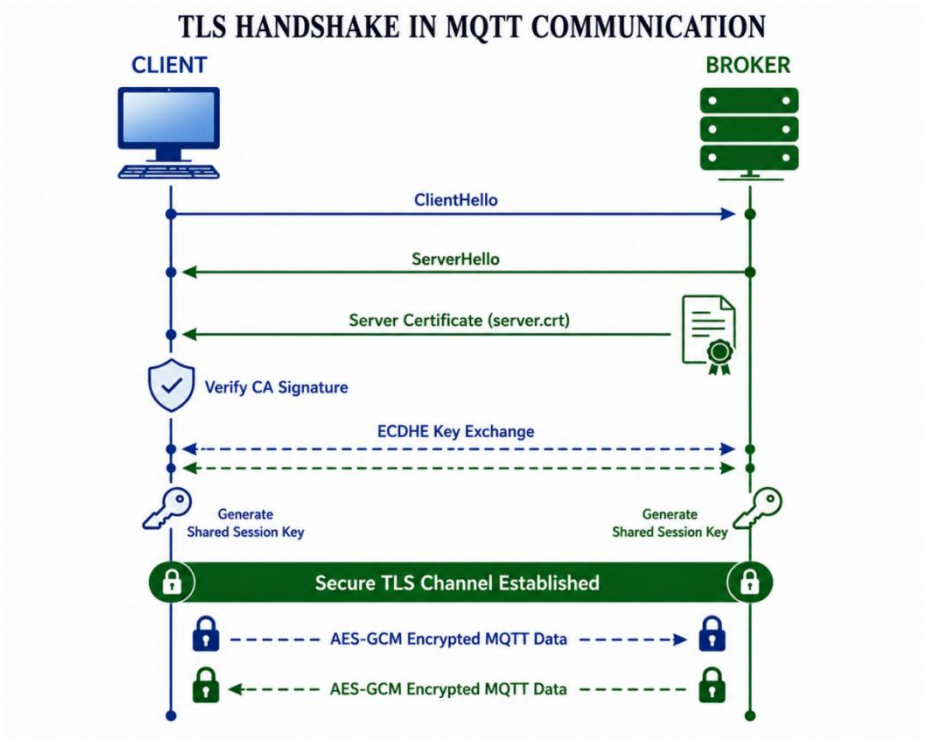


Fig - 5 : TLS handshake in MQTT communication

As discussed earlier, the TLS protocol is designed to establish a secure channel between a client and a server to provide information authenticity, confidentiality, and integrity. Specifically, its handshake layer adopts different PKC algorithms for identity authentication and allows the negotiation of a cipher suite, which consists of a set of cryptographic algorithms. This enables data confidentiality and integrity later in the record layer. Since the detailed understanding of the handshake procedure is essential for analyzing resource consumption, we first summarize the phases of this procedure as follows. The entire process is presented in Fig. 5.

1) Hello Request: This message may be sent by the server at any time as a notification that the client should begin a new negotiation. The client may ignore this message or send Client Hello when convenient.
2) Client Hello: This message is sent when the client first connects to the server or in response to a Hello Request. The record of this message contains the following fields: (i) TLS version, (ii) a random number, (iii) an optional Session ID to quickly resume a previous TLS connection and skip some steps of the TLS handshake, (iv) a list of cipher suites (specifies the key exchange algorithm, bulk encryption algorithm, MAC and a Pseudo-Random Function (PRF)), (v) the compression method, which is often null to avoid CRIME attacks [27].

3) Server Hello: This message is sent by the server in response to Client Hello. A Server Hello message includes: (i) TLS version, (ii) a random number, (iii) session ID: in case Client Hello's session ID is not empty, the server looks into its session cache for a match, and (iv) the cipher suite selection chosen from the client list, which should be the strongest suite supported by both sides.
4) Server Certificate: The server must send a certificate message immediately after Server Hello. The certificate type must be appropriate for the selected cipher suite key exchange algorithm.
5) Server Key Exchange: This message is sent immediately after the Server Certificate message. Only for Ephemeral Diffie-Hellman (DHE), Diffie-Hellman (DH), and RSA key exchange the server uses this message to specify the cryptographic parameters. In the case of Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) the key exchange parameters reside in the Server Certificate message. If the key exchange is based on EC, the server specifies the curve name only. Both the server and the client can derive the curve parameters, such as prime p and generator G, based on the NIST standard [28]. The server also chooses a random private key a, computes a ∗ G as the public key and saves it in server_params. In addition to this, the server also signs the data with its private key.
6) Certificate Request: A server can optionally request a certificate from the client. This message, if sent, will immediately follow Server Key Exchange message. The message Certificate Request specifies the certificate types that the client may offer and a list of supported hashing/signature algorithms pairs that the server is able to verify, according to [29].
7) Server Hello Done: This message is sent by the server to indicate the end of the Server Hello message.
8) Client Certificate: This message is sent only if the server requests a certificate, which must be appropriate for the negotiated cipher suite's key exchange algorithm. The client signs the certificate. In this paper, we use either RSA or ECDSA as the signing algorithms.
9) Client Key Exchange: This message is sent after receiving Server Hello Done. This message contains the client's DH public key, due to our choice of using ECDH or ECDHE as PKC. If the cipher does not indicate ephemeral key like ECDH, the message will be empty.
10) Change Cipher Spec: This message is sent by both the client and server, as the final non-encrypted message, in order to notify the other party that subsequent messages are protected under the most recently negotiated cipher suite.
11) Certificate Verify: This message is sent by the client only when Client Certificate is sent, in order to provide explicit verification for its own certificate. This message contains all the messages sent or received starting at the message Client Hello and up to but not including this message. This requires the client to buffer the messages or compute or buffer the hash of the previous messages.
12) Finished: This message is sent immediately after Change Cipher spec, in order to verify that the key exchange and authentication processes have been successful. The Finished message is the first protected packet with the most recently negotiated algorithms, keys, and secrets. Before

transmitting any encrypted data, both the client and the server generate several keys including the encryption keys, MAC keys and IV (nonce) using the master_secret as the seed for the PRF.

After the handshake phase is completed, the data transmitted between the client and server at the record layer will be secured by applying the symmetric cryptographic algorithms negotiated above in the listed steps. In the next section, we will investigate the algorithms and their individual functions, that construct a cipher suite, in detail.

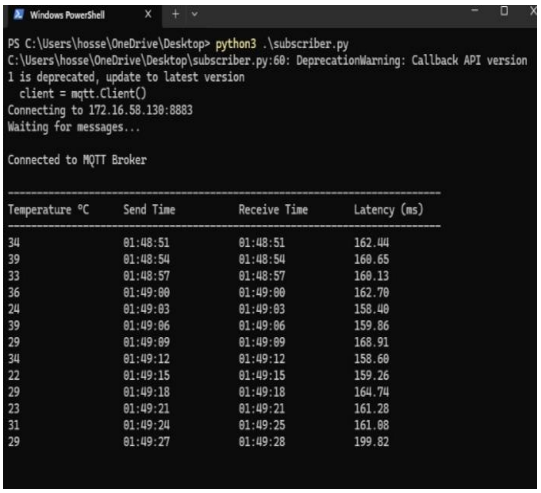


Fig - 6 : use_tls TRUE

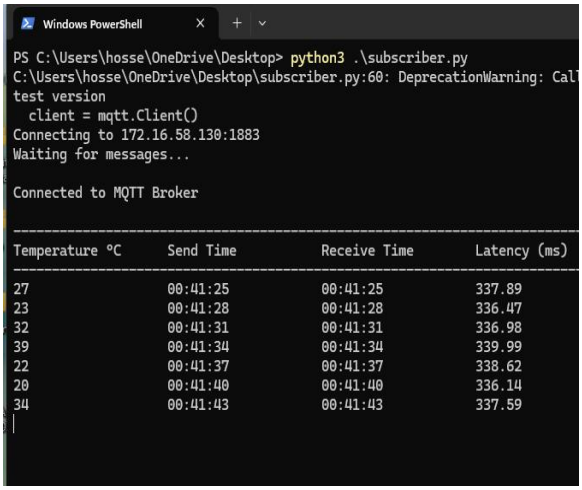


Fig - 7 : use_tls FALSE

To switch between TLS and non-TLS configurations in the broker, subscriber and publisher, the use_tls parameter is utilized. When use_tls is set to True, the broker is configured to listen on port 8883[Fig. 6] and use the specified TLS certificates. The subscriber and publisher are also set to connect to port 8883 and use the appropriate CA certificate for secure communication. Conversely, when use_tls is set to False, the broker listens on port 1883[Fig. 7] without any TLS settings, and the subscriber and publisher connect to this non-secure port. For example, in the broker configuration file (mosquitto.conf), enabling TLS would involve setting the listener to port 8883 and specifying the paths to the CA certificate, server certificate, and server key:

listener 8883
allow_anonymous false
password_file /etc/mosquitto/passwd

cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/mosquitto.crt
keyfile /etc/mosquitto/certs/mosquitto.key

For non-TLS, the configuration would simply include:
listener 1883
allow_anonymous false

In the configuration file allow_anonymous false authenticates the connection between the subscriber and the broker if the username and password match and in

the subscriber and publisher scripts, the use_tls parameter controls the configuration. When use_tls is True, the scripts set the port to 8883 [Fig. 6] and configure TLS settings. When use_tls is False, they connect to port 1883 [Fig. 7] without TLS settings.

```
use_tls = True # or False for non-TLS
port = 8883 if use_tls else 1883
ca_cert = "/path/to/ca.crt" if use_tls else None

client.username_pw_set(
        username="*****",
           password="****"
        )
```

This setup ensures that the system can flexibly switch between secure and non-secure communication modes by adjusting the use_tls parameter and the corresponding configuration settings in the broker, subscriber and publisher [30].

3.4. Publisher source code

```
 import paho.mqtt.client as mqtt
import time
import random
import json
import ssl
from datetime import datetime

broker = "192.168.0.102"
use_tls = True
port = 8883 if use_tls else 1883
topic = "iot/temperature"
client = mqtt.Client()

client.username_pw_set(
   username="kaium",
   password="5524"
)

if use_tls:
   client.tls_set(
      ca_certs="/etc/mosquitto/certs/ca.crt",
      tls_version=ssl.PROTOCOL_TLS_CLIENT
   )
print(f"Connecting to {broker}:{port}")
client.connect(broker, port)
print("Publisher Started...\n")
print("-" * 45)
print(f"{'Temperature':<15}{'Send Time':<30}")
print("-" * 45)
MAX_MESSAGES = 10
count = 0
while True:
   temp = random.randint(20, 40)
   send_time = time.time()
   payload = {
      "temperature": temp,
      "send_time": send_time
   }
   client.publish(topic, json.dumps(payload))
   send_time_str = datetime.fromtimestamp(send_time).strftime("%H:%M:%S.%f")[:-3]
   print(
      f"{str(temp) + '°C':<15}"
      f"{send_time_str:<30}"
   )
   count += 1
   if count >= MAX_MESSAGES:
      break
   time.sleep(3)
client.disconnect()
print("\nPublishing completed.")
```

3.5. Subscriber Source Code

```
import paho.mqtt.client as mqtt
import time
import json
import ssl
import csv
import sys
import statistics
from datetime import datetime

broker = "192.168.0.102"
use_tls = True
port = 8883 if use_tls else 1883
topic = "iot/temperature"
MAX_MESSAGES = 10
message_count = 0
latency_data = []

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        print("Connected to MQTT Broker\n")
        client.subscribe(topic)

        print("-" * 95)
        print(
            f"{'Temperature (°C)':<20}"
            f"{'Send Time':<25}"
            f"{'Receive Time':<25}"
            f"{'Latency (ms)':<15}"
        )
        print("-" * 95)
    else:
        print(f"Connection failed: {rc}")
def on_message(client, userdata, msg):
    global message_count
    receive_time = time.time()
    data = json.loads(msg.payload.decode())
    temperature = data["temperature"]
    send_time = data["send_time"]
    latency = (receive_time - send_time) * 1000
    send_time_str =
datetime.fromtimestamp(send_time).strftime("%H:%M:%S.
%f")[:-3]
    receive_time_str =
datetime.fromtimestamp(receive_time).strftime("%H:%M:%
S.%f")[:-3]
    print(
        f"{temperature:<20}"
        f"{send_time_str:<25}"
        f"{receive_time_str:<25}"
        f"{latency:<15.3f}"
    )
    latency_data.append([
        temperature,
        send_time_str,
        receive_time_str,
        latency
    ])
    message_count += 1
    if message_count >= MAX_MESSAGES:

        with open("latency_results.csv", "w", newline="") as file:
            writer = csv.writer(file)
            writer.writerow([
                "Temperature (°C)",
                "Send Time",
                "Receive Time",
                "Latency (ms)"
            ])
            writer.writerows(latency_data)
        latencies = [row[3] for row in latency_data]
        mean_latency = statistics.mean(latencies)
        std_latency = statistics.stdev(latencies)
        variance_latency = statistics.variance(latencies)
        min_latency = min(latencies)
        max_latency = max(latencies)
        print("\nResults saved to latency_results.csv\n")
        print("-" * 95)
        print(
            f"{'Mean Latency (ms)':<20}"
            f"{'SD (ms)':<15}"
            f"{'Variance (ms²)':<20}"
            f"{'Min Latency (ms)':<20}"
            f"{'Max Latency (ms)':<20}"
        )
        print("-" * 95)
        print(
            f"{mean_latency:<20.3f}"
            f"{std_latency:<15.3f}"
            f"{variance_latency:<20.3f}"
            f"{min_latency:<20.3f}"
            f"{max_latency:<20.3f}"
        )
        client.disconnect()
        print("\n\n")
        sys.exit(0)
client = mqtt.Client()
client.username_pw_set(
    username="kaium",
    password="5524"
)
if use_tls:
    client.tls_set(
        ca_certs="ca.crt",
        tls_version=ssl.PROTOCOL_TLS_CLIENT
    )
client.on_connect = on_connect
client.on_message = on_message
print(f"Connecting to {broker}:{port}")
client.connect(broker, port)
print("Waiting for messages...\n")
client.loop_forever()
```

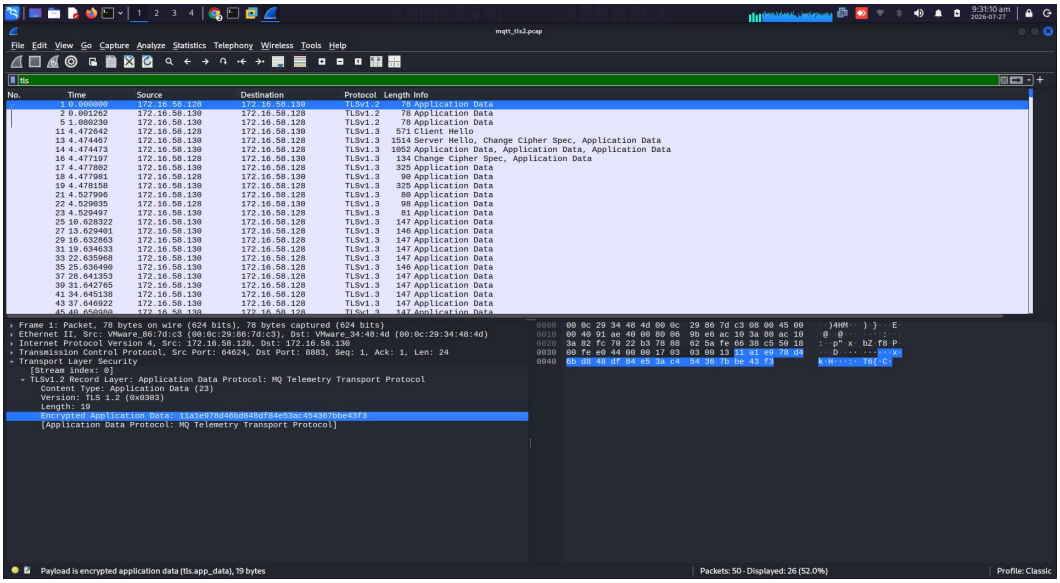


Fig - 8 : After TLS message encrypted

Fig. 8 shows Wireshark captured MQTT communications protected by TLS between two hosts, 172.16.58.128 and 172.16.58.130. The communication is established over TCP port 8883, which is commonly used for MQTT over TLS. In captured network packet list, Wireshark identifies the traffic as TLSv1.2 and TLSv1.3, with multiple packets labeled “Application Data”. The selected packet is identified as TLSv1.2 Application Data and the protocol is MQTT under the TLS record. Also Fig - 8 shows the payload as “Encrypted Application Data”, it means the transmitted data or message content is not readable in plaintext. This demonstrates that, even though network packets can still be captured by an attacker using packet-sniffing tools, TLS protects the MQTT payload from being directly viewed or understood, thereby reducing the effectiveness of a MitM interception attack.

3.6. Certificate verification

A Certificate Authority is a trusted entity that issues and signs digital certificates. These certificates are used to verify the identity of a server, device or user and to establish encrypted communication using TLS/SSL.

Without certificate verification, the publisher doesn’t know if it is a legitimate broker or fake[shown in Fig. 9], it will just send data if the payload topic matches. The publisher can’t identify the differences between a real broker and a fake broker.

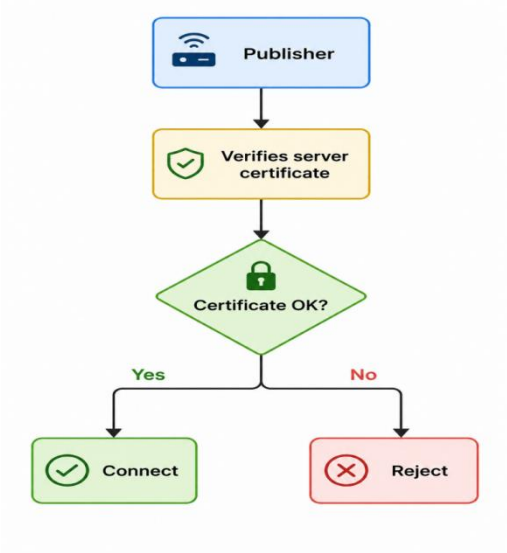


Fig - 9 : Certificate verification

3.7. Performance Matrix

3.7.1. Measuring Latency(ms)

Latency is the amount of time that data transmission takes, from its source (such as an IoT device) to its destination.

Latency can be assess by using a ping test, which measures the round-trip time for a message to travel from a source to its destination and be echoed back to the source.

Latency (ms) = (Receive Time - Send Time) * 100

3.7.2. Measuring Average (Mean) Latency

Average Latency means is the delay incurred as a result of the process. To compare MQTT without TLS and MQTT with TLS, measure multiple messages and compute the average:

$$\overline{x}=\frac{\sum_{i=1}^{n} x_i}{n}$$

where:

i. $\overline{x}$ = average latency

ii. x_i = latency of the iii-th message

iii. n = total number of messages

3.7.3. Standard Deviation (SD)

The standard deviation of a random variable, sample, statistical population, data set or probability distribution is the square root of its variance (the variance being the average of the squared deviations from the mean). A useful property of the standard deviation is that, unlike the variance, it is expressed in the same unit as the data. Standard deviation can also be used to calculate standard error for a finite sample, and to determine statistical significance[31]. The standard deviation represent the variability of latency measurement in milliseconds and indicate the consistency of communication delay.

The standard deviation equation is:

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}$$

where:

i. s = standard deviation

ii. x_i= individual latency value

iii. $\overline{x}$ = average latency

iv. n = number of latency samples

3.7.4. Variance

Variance is the square of the standard deviation and it's expressed in milliseconds squared (ms^2). While variance is useful for statistical analysis, standard deviation is generally easier to interpret because it uses the same unit as the measured latency.

The variance equation is :

$$s^2 = \frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}$$

It is useful for calculations, but standard deviation is easier to interpret because it has the same unit (milliseconds).

3.7.5. Percentage Increase in Latency

To quantify the effect of TLS:

$$\text{Latency Increase}(\%) = \frac{\text{Average TLS Latency - AverageNon_TLSLatency}}{\text{AverageNon_TLSLatency}} \times 100$$

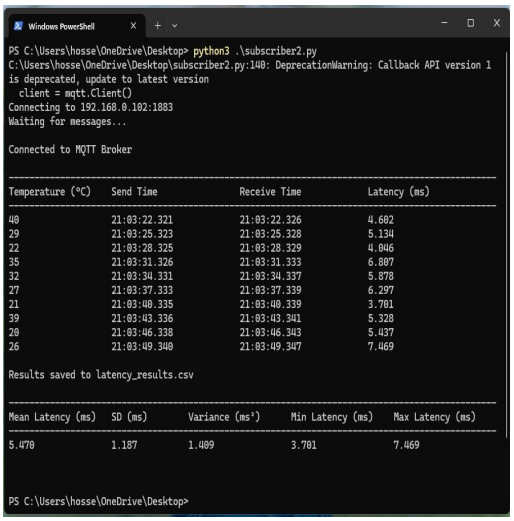


Fig -10 : MQTT without TLS

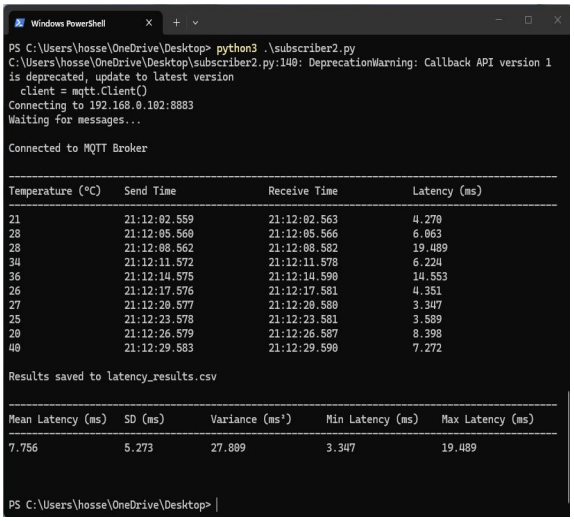


Fig -11 : MQTT with TLS

CHAPTER 4
RESULT AND ANALYSIS

This chapter presents the experimental implementation and evaluation of the proposed MQTT based IoT network under both normal and attack scenarios.The experiments were set up in an environment that contained an MQTT publisher, an MQTT broker, an MQTT subscriber and a Man-in-the-Middle attacker. The goal was to observe how a Man-in-the-Middle attacker can disrupt MQTT communication. The experiments also examined the effectiveness of security mechanisms such, as Transport Layer Security and broker authentication. In this way the experiments aimed to understand the impact of a Man-in-the-Middle attack and to evaluate how strong TLS and broker authentication are. The results appear as outputs MQTT message logs and network packet captures, which illustrate the communication behavior before and after applying the proposed security measures.

4.1 MitM attack result

The MitM attack result represents the results obtained from performing a Man-in-the-Middle (MITM) attack on the simulated MQTT based IoT network. During the attack, the person who was trying to attack was able to catch MQTT messages that were being sent between the person sending the message and the person receiving the message in a network that was not protected. The intercepted messages were analysed and where applicable, modified before being forwarded to the subscriber, demonstrating the potential impact of message tampering on data integrity. The effectiveness of the proposed mitigation techniques, including TLS encryption and broker authentication, is then evaluated by comparing communication behaviour before and after the security mechanisms were implemented.

The observation of the system and the evidence of the system were captured during MitM attack are attached below:

a). **Observations from MitM attack**

1. MQTT messages captured successfully using tcpdump and wireshark.
2. Temperature data visible in plaintext without any encryption.
3. Packet interception achieved.

b). **Evidence from MitM attack**

1. Wireshark/tcpdump packet captures

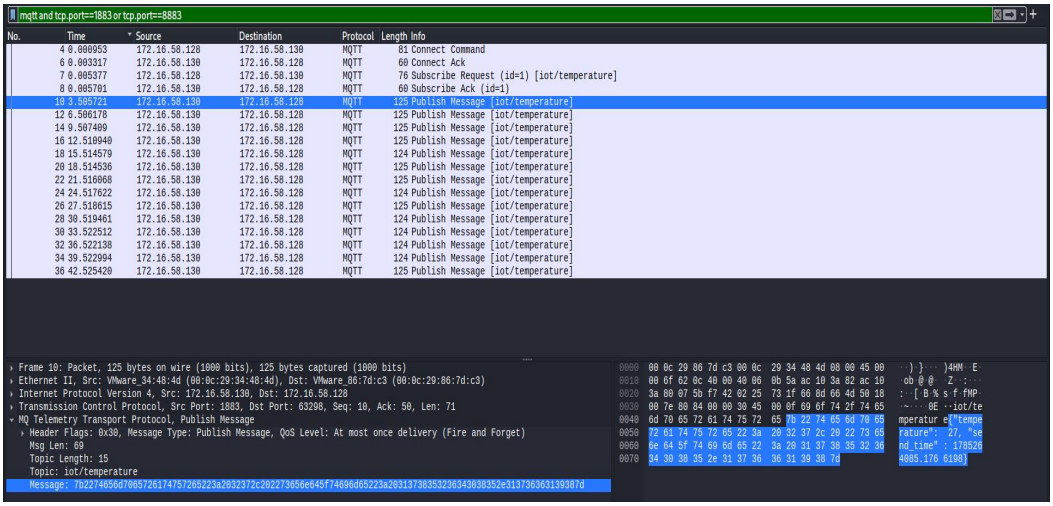


Fig - 12 : MitM before mitigation

2. MQTT message logs

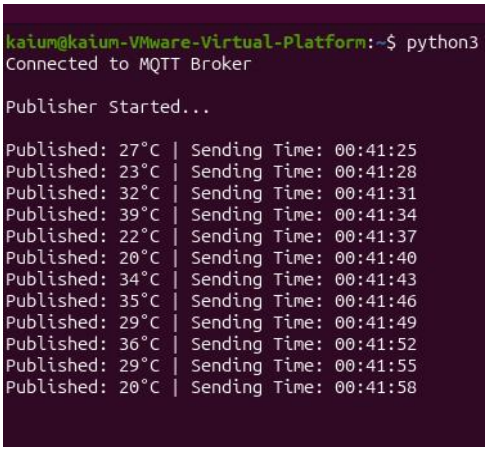


Fig - 13 : Publisher message log

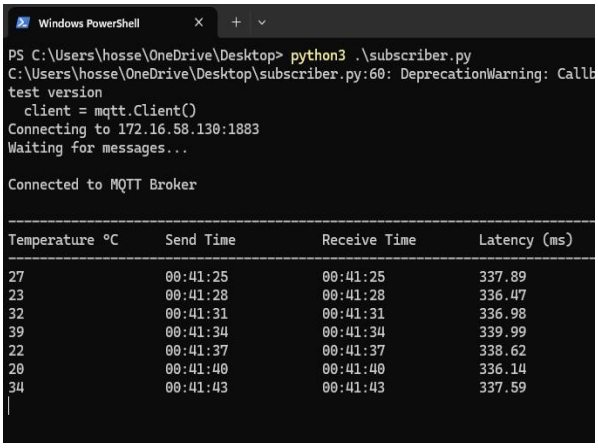


Fig -14 : Subscriber message log

3. ARP spoofing screenshots

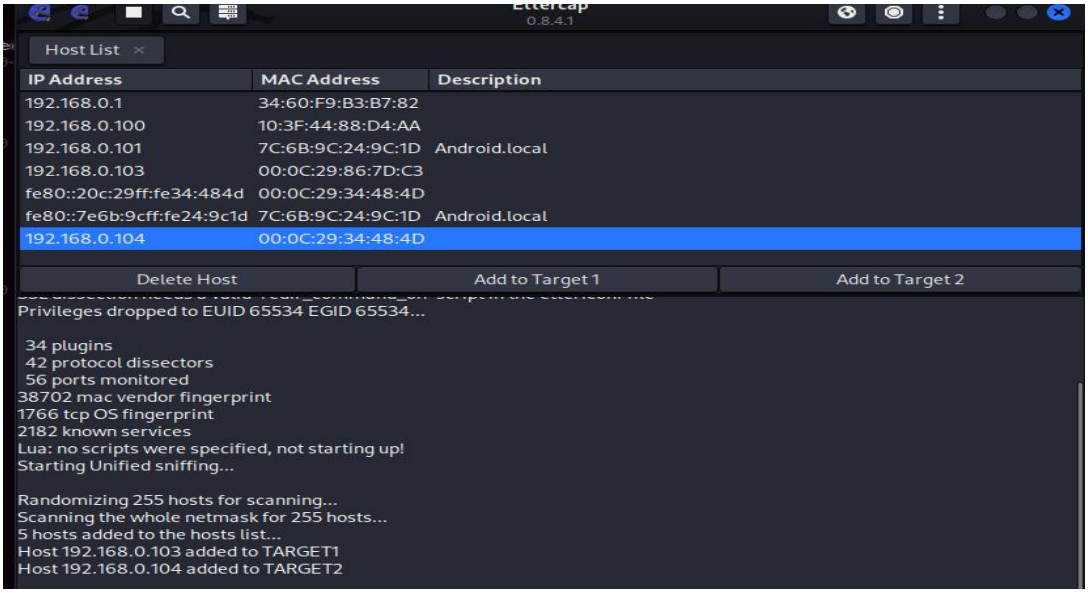


Fig- 15 : ARP Spoofing

Fig. 12 shows MQTT traffic is captured by wireshark during unsecured communication between publisher[Fig. 13] and subscriber[Fig. 14]. Topic, payload and message of the captured traffic are visible in plaintext. In general, Fig. 12 demonstrate unencrypted MQTT communication is vulnerable and payload disclosure during MitM attack. In Fig. 13 generating temperature data and sending data to the subscriber with actual sending time to measure latency. Fig.14 shows unsecured connection between subscriber and broker on port 1883. Subscriber received temperature data, sending time from broker and it's calculating latency according to the received time. Fig. 15 demonstrates the ARP poisoning/spoofing from an attacker's side to intercept the communication between publisher and subscriber.

4.2. Mitigation Results

a). **Before Mitigation**

i. Plaintext communication : MQTT communication is transmitted without encryption, allowing the exchanged data to remain readable during transmission. This makes the communication vulnerable to interception and unauthorized access.

ii. Successful packet interception : attacker successfully intercepts packets exchanged between the MQTT devices by performing a MitM attack. The captured packets can be analyzed to observe the communication between the publisher and subscriber.

iii. Anonymous login : MQTT broker allow clients to connect without requiring authentication credentials. This allows unauthorized clients to access the broker and potentially interact with MQTT topics.

b). **After TLS Implementation**

i. Encrypted MQTT traffic : TLS encrypts the MQTT communication between the client and broker, preventing unauthorized parties from easily reading the transmitted data. This protects the confidentiality of MQTT messages during network transmission.

ii. No readable payloads : MQTT payload is encrypted during transmission and therefore cannot be directly viewed as readable text through packet-capture tools. This significantly reduces the risk of sensitive information being exposed through intercepted network traffic.

iii. User authentication : User authentication requires clients to provide valid credentials before accessing the MQTT broker. This prevents unauthorized users from connecting to the broker and accessing protected MQTT communication.

4.3. Compare security between TLS and non_TLS

Configuration	Encryption	Username/ Password	Certificate Verification	Security Level	Latency
MQTT	No	No	No	Very low	Normal
MQTT + Authentication	No	Yes	No	Medium	Normal
MQTT + TLS	Yes	No	Yes (sever certificate)	High	High
MQTT + TLS + Authentication	Yes	Yes	Yes	Very High	Very High

Table 2 : Security comparison

In MQTT protocol default communication provide very low security and latency, while MQTT protocol with TLS implementation and CA authentication provides highest security but increase latency as highest. After TLS implementation the communication became more resistant to unauthorized access and MitM attack.

4.4. Discussion

4.4.1. Analysis of attack effectiveness

The experiment results show that the simulated Man-in-the-Middle attack was successful, at breaking MQTT communication. When MQTT ran without TLS encryption or broker authentication the attacker could easily grab messages sent between the publisher and the subscriber. We could see the intercepted MQTT traffic clearly in packet capture tools. This allowed the attacker to look at the temperature values and other parts of the messages. On top of that the attacker changed the temperature data before sending the data to the subscriber. This

shows that the integrity of the messages can be ruined without the people talking even knowing. The successful interception and modification of MQTT messages confirm that unsecured MQTT communication is highly vulnerable to MITM attacks and can lead to unauthorized data manipulation and loss of trust in IoT systems.

4.4.2. Analysis of security improvements

To mitigate the identified weakness, TLS encryption and broker authentication were implemented to the MQTT communication. TLS encrypted all MQTT traffic between the publisher, broker and subscriber stopping attackers from seeing the message data when network packets were captured. Packet analysis using Wireshark showed that the application data was encrypted making the temperature values and message contents unreadable. Also broker authentication required clients to provide valid usernames and passwords before establishing a connection. Unauthorized clients with incorrect credentials were rejected by the broker, preventing unauthorized access to the MQTT service. These security mechanisms significantly improved the confidentiality, integrity and access control of the simulated IoT network, thereby reducing the effectiveness of MITM attacks.

4.3.3 Comparative Analysis

Papers	Main Focus	Limitation	Defense Approach
O. Salem et al. (2021) [32]	MitM mitigation in IoMT	RSSI reliability and limited real-world validation	Irreversible signatures + RSSI authentication
Wong H.C. (2022) [33]	MitM attack and message manipulation in MQTT	No TLS evaluation	attack generation/detection analysis
Fereidouni H. et al. (2025)[34]	Comprehensive IoT MitM survey	No original experiments	Reviews existing mitigations
Our Study	MitM mitigation and secure connection in IoT	Latency	TLS implementation + CA authentication

Table 3 : Comparative analysis

The comparative analysis shows that previous studies focused on MitM attacks in IoT, IoMT and MQTT environments using RSSI-based authentication, irreversible signatures and attack detection approach. These studies have some limitations such as lack of real-world testing, lack of TLS-based evaluation and the absence of original experimental implementation. In contrast, our study focuses on mitigating MitM attacks through TLS-secured MQTT communication with CA-based authentication. Additionally, the proposed approach evaluates the

latency impact of TLS implementation, providing both security and performance analysis.

4.5. Performance Measure

Configuration	Min(ms)	Max(ms)	Mean (ms)	SD(ms)	Variance (ms2)	Percent Increase (%)
MQTT	3.701	7.469	5.470	1.187	1.409	NaN
MQTT + TLS	3.347	19.489	7.756	5.273	27.809	41.791

Table 4 : Latency comparison

This performance analysis shows a comparison of latency between the default MQTT and MQTT with TLS. The average latency increased from 5.470 ms to 7.756 ms and the maximum latency increased from 7.469 ms to 19.489 ms after TLS implementation. The higher standard deviation (5.273 ms) and variance (27.809 ms^2) indicate greater latency variability in the TLS configuration. Overall, TLS introduced a 41.791% increase in mean latency and also the results demonstrate that TLS enhances MQTT communication security while introducing a measurable increase in latency and variability.

CHAPTER 5
CONCLUSION

5.1 Summary of the Research

In this study, a virtualized testbed was established using VirtualBox to simulate an IoT architecture and analyze its vulnerability to active network threats. The experimental environment comprised three specialized virtual nodes connected over an isolated NAT Network:

1. **Broker and Publisher Node (Ubuntu 24.04 Desktop):** Hosted the Eclipse Mosquitto MQTT broker alongside a Python script generating real time telemetry .

2. **Subscriber Node (Windows 11):** Ran a Python based Paho MQTT client to receive and process published sensor streams.

3. **Attacker Node (Kali Linux 2024.4):** Utilized offensive security tools including Ettercap, Wireshark, tcpdump and custom Python packet manipulation scripts. The investigation proceeded in two main phases:

1. Vulnerability Assessment of Unsecured MQTT: When operating on standard plain-text transmission over port 1883, the network was subjected to an ARP spoofing attack using Ettercap. By positioning itself between the broker and subscriber, the attacker intercepted network traffic using Wireshark and tcpdump, fully exposing JSON-formatted payloads containing parameters such as device_id, temperature, humidity and timestamp.Furthermore, using IP forwarding and scapy/netfilterqueue scripts, the attacker modified temperature payloads in real time (e.g., escalating valid readings to critical values like 100°C) without interrupting the transmission or triggering connection alerts.

2. Mitigation via Security Enforcement: To address these security vulnerabilities, security features were configured on the Mosquitto broker. Transport Layer Security (TLS 1.2) was enabled on port 8883 using Open SSL generated self signed Certificate Authority (CA) certificates and broker keys. Anonymous connections were disabled and password-hash authentication was enforced.

5.2 Key Findings and Performance Evaluation

The experimental evaluation demonstrated a significant improvement in network security along with a minor impact on transmission performance:

Security Posture Improvement: Enforcing TLS 1.2 and user authentication elevated the overall security rating of the IoT network from Very Low to Very High. Traffic sniffing attempts yielded encrypted, unreadable binary streams. Cryptographic integrity checks completely prevented real time payload

modification and unauthorized command injection. Furthermore, unauthenticated access attempts to the secured broker port (8883) were consistently rejected.

The experimental results showed a difference in latency between the unsecured and secured configurations. In the setup the average latency was 5.470 ms. After adding TLS security the average latency went up to 7.756 ms. This increase mainly came from the time it takes to complete the handshake and the extra work needed for encryption. At the time network bandwidth usage also rose slightly. It moved from 1.2 KB/s in the case to 1.8 KB/s with TLS. This small rise is due, to the data needed for TLS headers and the additional steps involved in validating certificates during the connection setup. With these changes both setups kept a 0% packet loss rate. This means that all messages were delivered successfully throughout the testing period. The results show that the secure configuration didn't hurt reliability. Message delivery stayed stable with the added security layers.

This thesis demonstrates that standard MQTT communication without cryptographic protection leaves IoT networks highly vulnerable to active interception and data tampering. Although implementing TLS 1.2 encryption and user authentication introduces a small increase in average transmission latency and bandwidth overhead, this trade-off is essential to guarantee data confidentiality, payload integrity and node authentication. The implemented defense mechanism offers a practical, reliable and scalable security architecture for protecting MQTT based IoT deployments against MITM exploits.

CHAPTER 6
LIMITATION

6.1. Limitations of the Experiment

The experiment was carried out in a controlled laboratory environment using a simulated IoT network. While the setup was sufficient to demonstrate the impact of Man-in-the-Middle (MitM) attacks and evaluate the TLS-based mitigation approach, the findings should be considered within the following limitations.

6.2. Simulated Environment

The network was implemented using virtual machines instead of physical IoT devices. As a result, certain hardware-related factors, including processing capacity, memory constraints, and device-level communication behaviour, could not be fully represented. Therefore, the observed performance may differ from that of a real-world IoT deployment.

6.3. Limited Network Scale

The experimental network included one MQTT broker, one publisher, one subscriber, and one attacker. This relatively small configuration was adequate for evaluating the proposed attack scenarios but does not represent the complexity of large-scale IoT environments. Networks containing many interconnected devices may introduce additional traffic, resource demands, and security challenges.

6.4. Controlled Network Conditions

Testing was performed under relatively stable network conditions with limited background traffic. Practical IoT networks may experience congestion, packet loss, wireless interference, and variations in available bandwidth. Such factors were not incorporated into the experiment and may influence communication performance and mitigation effectiveness.

6.5. Latency Measurement Accuracy

Latency was estimated using timestamps recorded at the publisher and subscriber. The system clocks were synchronized before each test; however, minor timing differences may still occur in a virtualized environment. Therefore, the reported latency values should be regarded as approximate measurements of end-to-end communication delay rather than exact values.

6.6. Limited Attack Scenarios

The study concentrated on MitM attacks involving packet interception and message modification within MQTT communication. Other potential threats, including Denial-of-Service (DoS) attacks, replay attacks, broker compromise,

and sophisticated authentication bypass techniques, were not examined. Consequently, the results do not represent the security performance of the system against the full range of possible MQTT-related attacks.

6.7. Security Mechanisms Evaluated

The mitigation strategy mainly relied on TLS encryption and broker authentication. Additional security mechanisms, including mutual TLS (client certificates), intrusion detection systems (IDS), device attestation or hardware based security modules were not implemented or evaluated.

6.8. Certificate Management

TLS certificates were manually generated and configured for the experimental environment. In real world IoT deployments, certificate issuance, renewal, revocation and life cycle management present additional operational challenges that were not addressed in this study.

CHAPTER 7
FUTURE WORK

While this study effectively demonstrates MITM exploitation and TLS mitigation in a virtual setup, several avenues remain for future research:

7.1. Hardware based Implementation: Test the proposed TLS configuration on physical microcontrollers (e.g., ESP32, ESP8266, or Raspberry Pi) to measure the actual hardware energy consumption, memory footprint, and CPU overhead under attack conditions.

7.2. Intrusion Detection Systems (IDS): Integrate Machine Learning (ML) based Intrusion Detection Systems (such as Snort or custom Python scripts) to detect dynamic ARP spoofing attempts at the network gateway level in real time.

7.3. Advanced Attack Scenarios: Evaluate MQTT security against additional threat vectors, such as Denial of Service (DoS/DDoS) attacks on the Mosquitto broker, MQTT topic hijacking and DNS spoofing.

7.4. Lightweight Payload Encryption: Explore payload-level lightweight cryptographic algorithms (e.g., ChaCha20-Poly1305) as low overhead alternatives for ultra low power IoT edge devices that cannot support full TLS handshakes.

REFERENCES

1. https://isis-online.org/isis-reports/did-stuxnet-take-out-1000-centrifuges-at-the-natanz-enrichment-plant/
2. https://www.csidb.net/csidb/incidents/0feb0e2f-4287-437d-ab6d-f8b9ff4320f0/
3. https://www.38north.org/2024/03/revisiting-the-2014-korea-hydro-and-nuclear-power-hack-lessons-learned-for-south-korean-cybersecurity/
4. https://blog.ucs.org/dlochbaum/leak-at-the-creek-davis-besse-like-cooling-leak-shuts-down-wolf-creek/
5. Choudhary, A. (2024). Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions.
6. Muheden, K., Mohammed, M., & Bilal, N. (2024). Exploring the Landscape of Smart Cities: A Comprehensive Review of IoT and Cyber-Physical Systems.
7. Bellman, C., & Van Oorschot, P. C. (2019, August). Analysis, implications, and challenges of an evolving consumer iot security landscape.
8. Hintaw, A. J., Manickam, S., Aboalmaaly, M. F., & Karuppayah, S. (2023). MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT). *IETE Journal of Research*, *69*(6), 3368-3397.
9. Husnain, M., Hayat, K., Cambiaso, E., Fayyaz, U. U., Mongelli, M., Akram, H., ... & Shah, G. A. (2022). Preventing MQTT vulnerabilities using IoT-enabled intrusion detection system. *Sensors*
10. Nebbione, G., & Calzarossa, M. C. (2020). Security of IoT application layer protocols: Challenges and findings.
11. Wong, H., & Luo, T. (2020, August). Man-in-the-middle attacks on mqtt-based iot using bert based adversarial message generation.
12. Şimşek, M. M., & Atılgan, E. (2023). Attacks on availability of iot middleware protocols: A case study on mqtt. *Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi*
13. Sultan, A. B. M., Mehmood, S., & Zahid, H. (2022, March). Man in the middle attack detection for MQTT based IoT devices using different machine learning algorithms.
14. Harsha, M. S., Bhavani, B. M., & Kundhavai, K. R. (2018, September). Analysis of vulnerabilities in MQTT security using Shodan API and implementation of its countermeasures via authentication and ACLs.
15. Altulaihan, E., Almaiah, M. A., & Aljughaiman, A. (2022). Cybersecurity threats, countermeasures and mitigation techniques on the IoT: Future research directions. *Electronics.*
16. Mileva, A., Velinov, A., Hartmann, L., Wendzel, S., & Mazurczyk, W. (2021). Comprehensive analysis of MQTT 5.0 susceptibility to network covert channels.
17. Delgado, J. L. (2026). Observability for Post-Quantum TLS Readiness: A Multi-Surface Evidence Framework.

18. Fernando, R. An Analysis of API Security in Distributed Systems.
19. Mărieș, M., & Tătar, M. O. (2025). OT Control and Integration of Mobile Robotic Networks. *Electronics*.
20. Bendovschi, A. (2015). Cyber-attacks–trends, patterns and security countermeasures. *Procedia Economics and Finance*, *28*, 24-31.
21. Chordiya, A. R., Majumder, S., & Javaid, A. Y. (2018, May). Man-in-the-middle (mitm) attack based hijacking of http traffic using open source tools. In *2018 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 0438-0443). IEEE.
22. Al-Shareeda, M. A., Anbar, M., Manickam, S., & Hasbullah, I. H. (2020). Review of prevention schemes for man-in-the-middle (MITM) attack in vehicular ad hoc networks. *International Journal of Engineering and Management Research*, *10*.
23. Morsy, S. M., & Nashat, D. (2022). D-ARP: an efficient scheme to detect and prevent ARP spoofing. *IEEE Access*, *10*, 49142-49153.
24. Dodiya, B., & Singh, U. K. (2022). Malicious Traffic analysis using Wireshark by collection of Indicators of Compromise. *International Journal of Computer Applications*, *183*(53), 1-6.
25. Dierks, T., & Rescorla, E. (2008). *The transport layer security (TLS) protocol version 1.2* (No. rfc5246).
26. Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., ... & Pregartner, G. (2015, July). Securing smart maintenance services: Hardware-security and TLS for MQTT. In *2015 IEEE 13th international conference on industrial informatics (INDIN)* (pp. 1243-1250). IEEE.
27. Sarkar, P. G., & Fitzgerald, S. (2013). Attacks on ssl a comprehensive study of beast, crime, time, breach, lucky 13 & rc4 biases. *Internet: https://www. isecpartners. com/media/106031/ssl_attacks_survey. pdf [June, 2014]*.
28. Lily Chen and Dustin Moody. Elliptic Curve Cryptography, 2017. URL https: //csrc.nist.gov/Projects/Elliptic-Curve-Cryptography.
29. Dierks, T., & Rescorla, E. (2008). *The transport layer security (TLS) protocol version 1.2* (No. rfc5246).
30. Alharbi, S. S., Bell, D., & Awad, W. (2025, April). Application layer security: Mqtt perspective with tls implementation and analysis. In *2024 International Conference on IT Innovation and Knowledge Discovery (ITIKD)* (pp. 1-14). IEEE.
31. https://en.wikipedia.org/wiki/Standard_deviation
32. Salem, O., Alsubhi, K., Shaafi, A., Gheryani, M., Mehaoua, A., & Boutaba, R. (2021). Man-in-the-middle attack mitigation in internet of medical things. *IEEE Transactions on Industrial Informatics*, *18*(3), 2053-2062.
33. Wong, H. C. (2022). *Man-in-the-Middle Attacks on MQTT based IoT networks*. Missouri University of Science and Technology.
34. Fereidouni, H., Fadeitcheva, O., & Zalai, M. (2025). IoT and man-in-the-middle attacks. *Security and Privacy*, *8*(2), e70016.