International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Securing MQTT Communication in IoT: An Analysis of Protocol Weaknesses and Improvements

G. Umasankar
Research Scholar, Department of Computer Science
Sri Ramakrishna Mission Vidyalaya College of Arts & Science, Coimbatore-20

Dr. K. Soundarraj,
Assistant Professor, Department of Computer Science,
Sri Ramakrishna Mission Vidyalaya College of Arts & Science, Coimbatore-20

***Abstract* - The Message Queuing Telemetry Transport (MQTT) protocol plays a vital role in IoT communications due to its efficient publish/subscribe architecture. However, with over 540,531 MQTT vulnerabilities identified globally, it suffers from critical security flaws, including the absence of mandatory encryption, identity spoofing risks, denial-of-service attacks, data tampering, and unauthorized privilege escalation.**

These vulnerabilities originate from both inherent protocol design limitations and insecure real-world implementations, exposing millions of connected devices to cyber threats when default security practices are neglected. Through systematic vulnerability analysis and security enhancement evaluation, this research demonstrates that robust encryption techniques like TLS/SSL, combined with lightweight cryptographic algorithms such as symmetric key cryptography and elliptic curve cryptography (ECC), can effectively secure resource-constrained IoT devices. Our proposed comprehensive security framework integrates dynamic key management, automated certificate exchanges, and layered defense mechanisms to mitigate man-in-the-middle attacks, unauthorized data access, and message manipulation. The implementation results show significant security improvements while maintaining acceptable performance overhead, thereby establishing a foundation for trustworthy, scalable MQTT-based IoT deployments across industrial and consumer applications.

Keywords - MQTT, IoT Security, Protocol Vulnerabilities, Encryption, Authentication, Message Integrity, TLS, Cybersecurity, Network Security, Publish-Subscribe, Anomaly Detection, Machine Learning, Lightweight Cryptography

I. INTRODUCTION

*T*he Internet of Things (IoT) landscape is expanding exponentially, with projections indicating over 75 billion connected devices by 2025, spanning industries from healthcare and manufacturing to smart cities and autonomous systems [1,2]. At the core of many IoT ecosystems is the Message Queuing Telemetry Transport (MQTT) protocol—a lightweight, efficient messaging technology specifically designed for resource-constrained devices and unreliable networks [3]. With its publish-subscribe model, Quality of Service levels, and minimal bandwidth overhead, MQTT has emerged as the de facto standard for real-time IoT communication [4,5].

However, this rapid IoT growth has amplified security vulnerabilities at an unprecedented scale. Recent security assessments have identified over 540,000 MQTT-related vulnerabilities globally, with critical flaws including authentication bypass, man-in-the-middle attacks, and buffer overflow vulnerabilities [6,7]. MQTT's inherent design, while optimized for performance and efficiency, lacks mandatory encryption in its default configuration and relies primarily on basic username/password authentication mechanisms [8]. This security gap makes MQTT deployments susceptible to threats such as unauthorized device access, message tampering, identity spoofing, and distributed denial-of-service attacks that can compromise entire IoT networks [9,10].

Contemporary trends in IoT cybersecurity research emphasize the integration of traditional cryptographic protections with intelligent monitoring systems [11]. Machine learning-based anomaly detection has emerged as a promising approach to identify suspicious MQTT traffic patterns, unauthorized device behaviours, and potential security breaches in real-time [12,13]. These AI-driven security mechanisms can analyse message frequency, payload characteristics, and communication patterns to detect deviations from normal IoT operations, providing an additional layer of defence against zero-day attacks and sophisticated intrusion attempts [14,15]. The challenge lies in developing comprehensive security frameworks that combine robust encryption, advanced authentication mechanisms, and intelligent threat detection while maintaining the protocol's core advantages for resource-constrained environments [16]

II. LITERATURE REVIEW

The Message Queuing Telemetry Transport (MQTT) protocol has become a widely adopted communication protocol in the Internet of Things (IoT) ecosystem due to its lightweight design and suitability for resource-constrained environments. MQTT operates on a publish-subscribe messaging pattern, enabling efficient and reliable delivery of messages between devices and brokers over unreliable networks.

Anomaly detection in MQTT-based IoT systems is a rapidly evolving research area focused on enhancing the reliability and security of low-cost, real-time communications. Katole and Pattanshetti proposed a supervised anomaly detection model that examines DoS attack patterns in MQTT traffic, employing classifiers such as Naive Bayes, Support Vector Machines, and ensemble methods to improve detection accuracy and reliability. Multiple machine learning (ML) techniques—including decision trees, random forests, neural networks, and ensemble classifiers—have been validated for intrusion and anomaly detection in IoT scenarios, achieving accuracy rates exceeding 99% on benchmark datasets. Comprehensive surveys underline the importance of ML and deep learning (DL) in automating feature extraction, enabling predictive maintenance, and identifying outliers across diverse, non-stationary, and multidimensional IoT data.

Recent studies emphasize the advantages of federated learning and edge analytics for robust distributed detection [34]. However, the integration of such systems faces challenges, including the lack of unified, scalable workflows for the collection of MQTT message streams, feature extraction, and deployment of anomaly detection models in operational IoT environments. Additionally, the necessity for labeled training data, limitations of existing datasets (e.g., MQTT-IoT-IDS2020, CoAP-IoT), and the difficulties in generalizing models to unseen threats constitute ongoing interdisciplinary challenges.[17]

III. BACKGROUND

MQTT is a lightweight publish/subscribe protocol developed specifically for resource-constrained edge environments prevalent in IoT deployments. Its widespread adoption increases exposure to adversarial actions, including denial-of-service, brute-force, malformed packet, and flooding attacks, which are particularly exacerbated by insufficient default authentication mechanisms and a lack of enforced encryption. The criticality of anomaly detection grows concurrently with the rapid proliferation of IoT devices, as attacks exploiting MQTT weaknesses can severely compromise industrial, healthcare, and consumer applications.

Traditional anomaly detection on MQTT traffic involves extracting time-series and categorical features from broker logs, normalization techniques such as min-max scaling, data splitting, and training supervised ML classifiers. Common evaluation metrics include accuracy, precision, recall, F1 score, and confusion matrices. Despite some models reporting high detection rates, many underperform against unknown or novel attack types due to rigid feature sets and static learning pipelines, underscoring the need for adaptive and flexible detection frameworks.[18]

IV. MQTT COMPONENTS

A. MQTT Client:

An MQTT client denotes any device or application that publishes messages to an MQTT broker or subscribes to topics to receive messages. Clients include sensors, actuators, mobile apps, and cloud services. They communicate by exchanging MQTT control packets such as CONNECT, PUBLISH, SUBSCRIBE, and DISCONNECT, which facilitate session establishment, message distribution, and graceful disconnections [20,21]

B. MQTT Broker (Server):

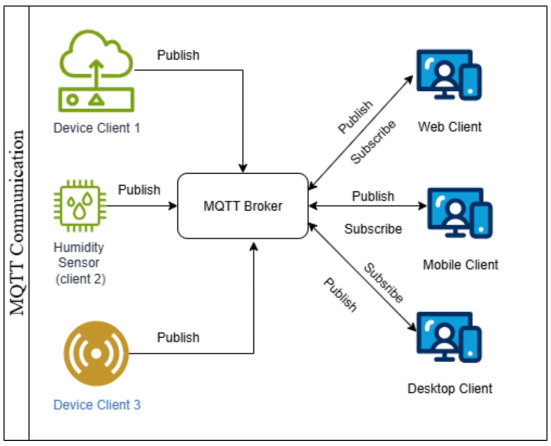
The MQTT broker is the central hub within the MQTT architecture. It receives all messages published by clients, filters messages based on topics, and distributes them only to clients subscribed to those topics. The broker manages client sessions, maintains connection states, queues messages, and enforces Quality of Service (QoS) levels to ensure reliable message delivery. It plays a critical role in maintaining efficient and secure communication between MQTT clients. [20,21]

C. MQTT Messages:

MQTT messages are the discrete packets of data transmitted between clients via the broker. Each message consists of three parts: a fixed header containing the message type and flags, a variable header with message-specific information, and a payload carrying the data. Messages are published to topics, which are structured hierarchical strings that clients can subscribe to or publish on. The payload size can be up to 256 MB, supporting a broad range of IoT use cases [22]

D. Topics:

Topics serve as the addressing scheme in MQTT, functioning as logical channels through which messages are routed. Clients publish data to topics and subscribe to topics of interest. MQTT topics support the use of wildcards, enabling flexible subscription patterns. This decoupling of message producers and consumers enhances scalability and efficiency in the publish-subscribe model, which is fundamental to MQTT's design.[23]



V. MQTT BROKERS AND IT'S TYPES

In Analysis, two types of MQTT Brokers are available in the market [20]

1. Open Source / Self-Managed Brokers
2. Managed Brokers

A. Hosted (Self-Managed/Open-Source) Brokers

Performance & Control:

EMQX, VerneMQ, and Mosquitto are widely recognized for high throughput and protocol-level customization in enterprise and edge deployments. For instance, EMQX achieved the best throughput (28K msg/s) and Mosquitto excelled in lightweight use cases and device-local scenarios.[25]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 03 , March - 2026

Scalability & Reliability:

VerneMQ offers open-source clustering and high availability, while Mosquitto requires paid "Professional" version for these features. HiveMQ's open-source edition does not support clustering, but the paid enterprise edition matches managed broker reliability.[27]

Customizability & Cost:

Hosted brokers permit extensive customization, protocol changes, and fine-tuned security, with lower ongoing costs. However, enterprise features may require paid support or advanced configuration.[26]

B. Managed (Cloud or Commercial Brokers)

Operational Ease:

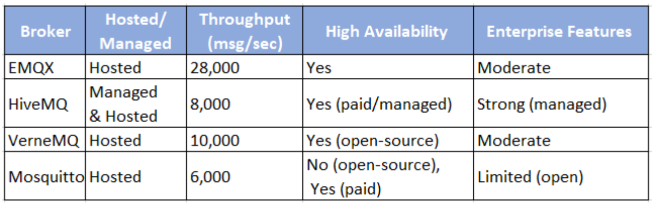
Managed brokers (HiveMQ Cloud, AWS IoT Core, Azure IoT Hub) provide zero-maintenance deployment, automated scaling, integrated monitoring, and advanced security out of the box. They excel in rapid IoT application development and production resilience.[28]

Scalability & Enterprise Features:

Managed solutions handle massive connection volumes and unpredictable traffic with built-in load balancing and multi-tenant support. Benchmarks show managed HiveMQ is the only broker with no message loss at high loads in independent evaluations. [25,26]

Limitations:

Deep protocol changes and extensive custom integration are usually restricted. High usage or advanced features incur ongoing costs



Broker	Hosted/ Managed	Throughput (msg/sec)	High Availability	Enterprise Features
EMQX	Hosted	28,000	Yes	Moderate
HiveMQ	Managed & Hosted	8,000	Yes (paid/managed)	Strong (managed)
VerneMQ	Hosted	10,000	Yes (open-source)	Moderate
Mosquitto	Hosted	6,000	No (open-source), Yes (paid)	Limited (open)

VI. SECURITY LAYERS IN MQTT AND IOT

The security layers applied to MQTT and IoT communication are cross-layer concerns that involve securing multiple layers where MQTT operates or depends upon:

A. Physical/Perception Layer Security:

Device-level security for sensors and actuators, including tamper resistance, secure firmware, lightweight cryptography, and secure boot processes.

B. Network/Transport Layer Security:

Encrypted communication using TLS/DTLS ensures the confidentiality and integrity of MQTT packets in transit, plus network isolation and intrusion detection.[30]

C. Messaging Layer Security (Application Layer):

Securing MQTT's publish/subscribe messaging system with authentication (e.g., username/password, certificates), authorization (ACLs, RBAC), topic-level permissions, payload encryption, and protection against injection or replay.[31]

D. Application Layer Security:

Payload encryption at the application level (end-to-end encryption), secure key management, and access control mechanisms.[29]

VII. MISSING FEATURES AND GAPS

Despite MQTT's success as a lightweight communication protocol optimized for IoT, key security features remain missing or inadequately addressed. These shortcomings, combined with protocol and deployment-specific gaps, increase exposure to attacks and data breaches in practical IoT scenarios.[31]

A. Mandatory Encryption:

MQTT does not enforce encryption at the protocol level, depending on optional TLS/SSL configuration. Many deployments still transmit sensitive data unencrypted.

B. Robust Authentication and Authorization:

Default username/password authentication is simplistic and often weakly managed. Dynamic, federated, or certificate-based authentication mechanisms are lacking in most deployments.

C. Granular Access Control:

Although brokers implement ACLs or RBAC, there are no universally accepted scalable mechanisms for dynamic and context-aware authorization to handle complex IoT setups.

D. Intrinsic Anomaly Detection and Security Logging:

MQTT currently lacks built-in capabilities to monitor traffic patterns, detect anomalies, and provide comprehensive logging for forensic investigations.

E. Integrated Key and Certificate Management:

The protocol does not provide frameworks to securely manage keys or certificates, complicating large-scale or heterogeneous IoT system deployments.

VIII. LAYER-WISE SECURITY FEATURES AND GAPS

The MQTT protocol uses a layered architecture tailored for efficient, lightweight communications in IoT applications. Each layer in this stack has its own responsibilities and associated security gaps that must be addressed for secure deployments. Below, we discuss each layer's security features and the corresponding gaps observed in MQTT implementations.

A. Network Transport Layer

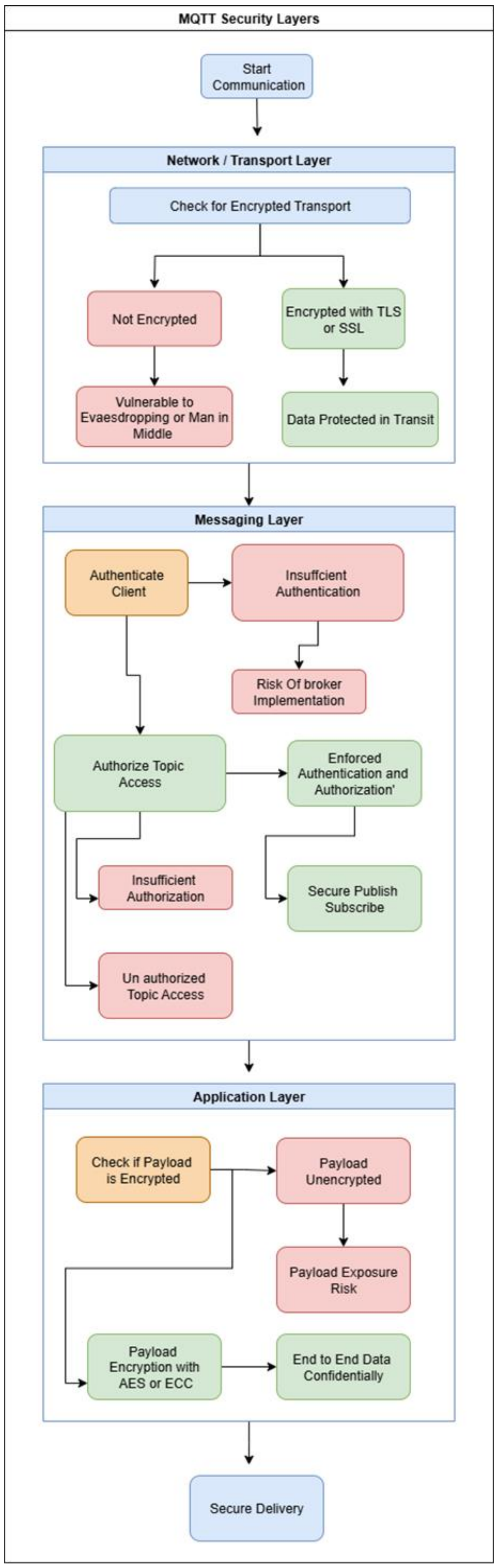
Security Features:

a. Encryption of the communication channel using **TLS (Transport Layer Security)** or **SSL (Secure Sockets Layer)** ensures data confidentiality and protects against eavesdropping and man-in-the-middle (MitM) attacks [35]

a. Use of mutual TLS enables authentication of both client and broker.[37]

Security Gaps:

a. By default, MQTT communication over TCP is **unencrypted**, exposing data to interception if TLS is not configured.
b. Older or misconfigured TLS versions (e.g., without mutual authentication) present vulnerabilities.
c. Lack of transport-layer encryption leads to exposure to replay and MitM attacks.


MQTT Security Layers
Start Communication
Network / Transport Layer
Check for Encrypted Transport
Not Encrypted
Encrypted with TLS or SSL
Vulnerable to Evaesdropping or Man in Middle
Data Protected in Transit
Messaging Layer
Authenticate Client
Insuffcient Authentication
Risk Of broker Implementation
Authorize Topic Access
Enforced Authentication and Authorization'
Insufficient Authorization
Secure Publish Subscribe
Un authorized Topic Access
Application Layer
Check if Payload is Encrypted
Payload Unencrypted
Payload Exposure Risk
Payload Encryption with AES or ECC
End to End Data Confidentially
Secure Delivery

B. Messaging Layer (Broker and Topic Management)

Security Features:

b. Client authentication ensures only valid devices can connect (via username/password or certificates).
c. Topic authorization limits client permissions on publish/subscribe to certain topics, implemented via Access Control Lists (ACLs) or Role-Based Access Control (RBAC).
d. Broker protects against impersonation by verifying identities and monitoring connections.[12]

Security Gaps:

a. Weak or static authentication credentials can be stolen or guessed, enabling unauthorized access.
b. Insufficient authorization controls allow unauthorized topic subscriptions or message injection.
c. Brokers without fine-grained ACLs risk message leaks or malicious control messages passing through.

C. Application Layer (Payload/Data Security)

Security Features:

a. Payload encryption using algorithms like **AES** or
b. **Elliptic Curve Cryptography (ECC)** provides end-to-end data confidentiality independent of transport encryption.
c. Clients encrypt/decrypt payload locally to protect sensitive data from compromised brokers or intermediaries.
d. Payload encryption prevents data leakage even within trusted networks if brokers or infrastructure are hostile.[13]

Security Gaps:

a. MQTT does not mandate payload encryption, so payloads are often sent in plain text.
b. Relying solely on transport security exposes payloads at the broker or intermediary nodes.
c. Implementing payload encryption increases complexity and key management challenges for constrained IoT devices.

IX. MQTT SECURITY IMPROVEMENT

Previous studies have revealed significant weaknesses in the MQTT protocol, particularly the lack of default encryption and authentication. To strengthen MQTT deployments in IoT ecosystems, it is necessary to align defenses with the three-tier IoT architecture—Perception, Network, and Application layers—each requiring specialized controls. [35,38]

A. Network Layer:

MQTT communications are susceptible to man-in-the-middle (MitM), replay, eavesdropping, Denial-of-Service (DoS), and routing attacks. Brokers operating in mixed modes (accepting both secure and insecure connections) exacerbate exposure.

To mitigate these risks:

- Enforce TLS/DTLS-based encryption with no fallback to plaintext.
- Apply mutual certificate authentication and enable certificate revocation mechanisms.
- Deploy Intrusion Detection Systems (IDS) based on distributed or ensemble machine learning to analyze MQTT traffic.
- Use network segmentation (VPNs, VLANs) to isolate critical IoT clusters.

Recent research demonstrated a distributed ML framework using H2O algorithms to detect MQTT network attacks, including DoS, MitM, and message tampering. This approach processed real-time traffic at IoT edge sensors, improving detection scalability.

B. Perception Layer:

At the edge, IoT devices are targeted through physical tampering, device spoofing, insecure firmware, and weak credentials. Limited computational resources restrict traditional cryptographic solutions.

Recommended protections include:

- Employing lightweight cryptography (PRESENT, ASCON, or grain-based ciphers).
- Utilizing secure boot, device attestation, and tamper-resistant hardware modules.
- Enforcing credential management and password rotation policies.

Data confidentiality can also be reinforced via payload encryption at the application layer, ensuring protection even if TLS is compromised.

C. Application Layer:

At the protocol level, MQTT remains vulnerable to packet injection, topic hijacking, unauthorized actions, and malformed payloads.

Security mechanisms include:

- End-to-end encryption using proxy re-encryption or application-layer modules.
- Token-based authentication frameworks such as OAuth or JWT.
- Fine-grained access control through ACLs with topic-based permissions.
- Continuous auditing and payload validation for injection resilience.

This layered approach ensures that risks are mitigated at both device and communication levels, forming a foundation for further enhancements.

X. PROPOSED SECURITY FRAMEWORK

Building upon the above improvements, this research proposes an integrated framework that consolidates cryptographic, authentication, and anomaly detection modules to provide scalable and lightweight MQTT security for IoT environments.[36]

A. Enhanced Network Transport Security

a. Mandatory TLS 1.3 or higher for all communications.
b. Session resumption mechanisms to reduce computational overhead for constrained devices.

B. Adaptive Authentication and Authorization

a. Multi-factor authentication (MFA) combined with lightweight PKI.
b. Dynamic RBAC and context-aware ACLs for granular topic-level control.

C. Application Layer Payload Encryption

a. ECC and AES-GCM for end-to-end message confidentiality.
b. Hierarchical IoT-specific key management scheme for secure, scalable distribution.

D. Intelligent Anomaly Detection

a. Machine learning-based anomaly detection module embedded in MQTT brokers.
b. Combination of supervised and unsupervised models for higher accuracy.
c. Automated alerts and real-time countermeasures against DoS, tampering, and impersonation.

E. Secure Broker Deployment

a. OS-level hardening and firewall enforcement.
b. Network segmentation to isolate IoT clusters.
c. Restricted broker access to authenticated devices and trusted subnets.

F. Continuous Security Assurance

a. Automated vulnerability scans and penetration testing for ongoing risk evaluation.
b. Secure OTA updates for IoT devices and brokers, ensuring timely patch deployment.

XI. CONCLUSION AND FUTURE WORK

Securing MQTT communication in IoT systems remains a critical challenge due to inherent protocol weaknesses such as weak authentication, lack of default encryption, insufficient access control, and vulnerability to denial-of-service attacks. This analysis highlighted these security gaps and the need for layered improvements across the protocol stack. Addressing them requires integrating encryption-based payload protection, strengthening middleware authentication and authorization, and securing perception layer data collection.

Future work directions focus on leveraging advanced anomaly detection techniques using MQTTnet within the .NET ecosystem, combined with ML.NET machine learning capabilities. This approach aims to identify abnormal MQTT message patterns that indicate security threats or attacks in real time. Integrating MQTT communication security enhancements with intelligent anomaly detection mechanisms will substantially improve the robustness of IIoT deployments in critical environments. Overall, this strategy aligns well with evolving IoT security requirements by combining protocol-level improvements with data-driven threat detection and adaptive response capabilities, enhancing protection and resilience.

The fusion of communication protocol hardening and machine learning–powered anomaly detection within the widely used .NET ecosystem promises an effective path forward to securing MQTT-based IoT and IIoT infrastructures for modern applications.

XII. REFERENCES

[1] Statista, "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025," 2024.
[2] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper," Cisco Systems, 2020.
[3] A. Banks and R. Gupta, "MQTT Version 3.1.1," OASIS Standard, Oct. 2014.
[4] N. Naik, "Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP," in Proc. IEEE Int. Syst. Eng. Symp. (ISSE), 2017, pp. 1-7.
[5] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in Proc. 3rd Int. Conf. Commun. Technol. Appl., 2008, pp. 791-798.
[6] Shodan, "MQTT Vulnerability Statistics," Global IoT Security Report, 2024.
[7] T. Kothmayr et al., "DTLS-based security and two-way authentication for the Internet of Things," Ad Hoc Networks, vol. 11, no. 8, pp. 2710-2723, 2013.
[8] S. N. Swamy et al., "Security issues, vulnerabilities, methods, and challenges of the MQTT protocol used with IoT," Int. J. Eng. Res. Technol., vol. 6, no. 6, pp. 876-883, 2017.
[9] A. Niruntasukrat et al., "Security in IoT environment with MQTT protocol," in Proc. 18th Int. Conf. Adv. Commun. Technol. (ICACT), 2016, pp. 28-31.
[10] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT) with proof of possession," in Proc. Int. Conf. Commun. Signal Process., 2015, pp. 2072-2077.
[11] Y. Yang et al., "A survey on security and privacy issues in Internet-of-Things," IEEE Internet Things J., vol. 4, no. 5, pp. 1250-1258, Oct. 2017.
[12] H. H. Pajouh et al., "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks," IEEE Trans. Emerg. Topics Comput., vol. 7, no. 2, pp. 314-323, Apr. 2019.
[13] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," Procedia Comput. Sci., vol. 167, pp. 636-645, 2020.
[14] M. A. Al-Garadi et al., "A survey of machine and deep learning methods for Internet of Things (IoT) security," IEEE Commun. Surv. Tutor., vol. 22, no. 3, pp. 1646-1685, 2020.
[15] L. Xiao et al., "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?" IEEE Signal Process. Mag., vol. 35, no. 5, pp. 41-49, Sep. 2018.
[16] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," IEEE Commun. Surv. Tutor, vol. 17, no. 3, pp. 1294-1312, 2015.
[17] B. H. Katole and T. R. Pattanshetti, "Study on Supervised Anomaly Detection Model for MQTT-Based IoT Data for Dos Attacks," IJFMR, vol. 7, no. 2, 2025.
[18] S. H. Rafique, et al., "Machine Learning and Deep Learning Techniques for Anomaly Detection in IoT," Frontiers in Computer Science, 2024.
[19] MQTTnet Wiki, "Server · dotnet/MQTTnet Wiki," 2022
[20] Amazon Web Services, "What is MQTT? - MQTT Protocol Explained," Sep. 2025

[21] Cavli Wireless, "MQTT Protocol in IoT: A Guide to Reliable," Feb. 2025.

[22] HiveMQ, "MQTT Tutorial: An Easy Guide to Getting Started with MQTT," May 2025.

[23] Paessler AG, "Understanding MQTT Architecture," Sep. 2023.

[24] Cedalo, "MQTT Protocol Explained: The Complete Guide," Jun. 2025

[25] Catchpoint, "The Guide to MQTT Broker," Aug. 2011.

[26] Diyusthad, "Top 5 Open-Source MQTT Brokers in 2025: Features and Comparison," Jan. 2025

[27] HiveMQ, "MQTT Broker Comparison – Which is the Best for Your IoT Application?" Aug. 2022

[28] EMQX, "A Comprehensive Comparison of Open Source MQTT Brokers in 2023," Aug. 2025.

[29] Cirrus Link Solutions, "Securing MQTT: Best Practices for a Robust IoT Ecosystem," Sep. 2025.
[30] K. Oliynyk, "Protecting Your IoT Infrastructure: Essential MQTT Security Practices," Webbylab, Jan. 2025.
[31] HiveMQ, "Securing MQTT Systems - MQTT Security Fundamentals," Mar. 2024

[32] Comprehensive Security assessment of common open source MQTT brokers and clients. (2023). [Online]. Available: https://arxiv.org/pdf/2309.03547

[33] Security and Privacy in the IIoT: Threats, Possible Security Countermeasures, and Future Challenges. (2025). [Online]. Available: https://scifiniti.com/uploads/source/articles/computingai-connect/2025/volume2/2025002025.0011/article.pdf

[34] ETASR. Privacy Policy. Retrieved from https://etasr.com/index.php/ETASR/about/privacy

[35] Secure Enhancement for MQTT Protocol Using Distributed Machine Learning. (2024). [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC10935133/

[36] Securing MQTT: Best Practices for a Robust IoT Ecosystem. (2024). Available https://cirrus-link.com/wp-content/uploads/2024/01/Securing-MQTT-Best-Practices-for-a-Robust-IoT-Ecosystem.pdf

[37] Paris, I.L.B.M., et al. Implementation of SSL/TLS Security with MQTT Protocol in IoT Environment

[38] Authora. Enhancing IoT communication security: Analysis and mitigation of vulnerabilities in MQTT, CoAP, and XMPP protocols. [Online]. Available: https://www.authorea.com/users/878219/articles/1257947-enhancing-iot-communication-security-analysis-and-mitigation-of-vulnerabilities-in-mqtt-coap-and-xmpp-protocols