IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45SP/2025/4515125/
Vol. 2025 (23), Article ID 4515125

Security Comparison of Messaging Protocols for Internet of Things*

Jerzy KRAWIEC, Piotr GÓRNY and Maciej KIEDROWICZ

Warsaw University of Technology, Warsaw, Poland

Correspondence should be addressed to: Jerzy KRAWIEC, jerzy.krawiec@pw.edu.pl

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

Many articles present comparative analyses of communication protocols used in the Internet of Things. However, most of them are concerned with the efficiency and functionality of these protocols, and the comparative analysis is concerned with a not very broad set of protocols. This article aims to present a pragmatic comparison of several of the most frequently used communication protocols in IoT systems when choosing a protocol regarding security. To compare the protocols in terms of security, we use the Common Vulnerabilities and Exposures (CVE) database and the Common Vulnerability Scoring System (CVSS). We present a list of security services supported by message protocols according to the criteria: Authentication, Authorization, and Confidentiality. We show the number of vulnerabilities detected in recent years regarding the protocols studied.

Keywords: Messaging protocol, IoT, Security, CVE, CVSS, Vulnerability

Introduction

The automation, artificial intelligence, and IoT era force companies to conduct business in digital transformation. Messaging protocols provide reliable communication between applications in IoT. Messaging protocols have become the backbone of communication, connecting people with speed and reliability across different platforms and devices. From instant messaging applications to sophisticated business communication systems, these protocols ensure secure, efficient, and scalable exchanges of messages. Selecting the proper protocol for a specific IoT system is complex and challenging.

A pragmatic comparison of protocols that can be included in IoT systems is of fundamental importance for the designers of these systems. Some articles (Wytrebowicz at al., 2021; Goworko and Wytrebowicz, 2021; Rzepka at al., 2024) compare protocols regarding the efficiency analysis of selected executions, and the comparisons are often related to a very narrow set of protocols.

Nevertheless, these papers do not touch on complex security aspects. A key feature of an IoT system is information security. A secure IoT system should ensure secure communication of devices where the trust model is based on real-world business relationships. Such a model can be implemented using secure communication with an encrypted DTLS protocol, which ensures data privacy and integrity in applications that require fast connections and are sensitive to latency. A significant problem is reducing system performance, such as computing power or energy consumption, with the need to use security mechanisms, e.g., strong encryption. The research compares protocol characteristics such as transport type, communication pattern type, and security attributes. The research evaluates the three most popular communication protocols: MQTT, CoAP, and OPC UA (Silva at al., 2021).

Cite this Article as: Jerzy KRAWIEC, Piotr GÓRNY and Maciej KIEDROWICZ, Vol. 2025 (23) "Security Comparison of Messaging Protocols for Internet of Things " Communications of International Proceedings, Vol. 2025 (23), Article ID 4515125, https://doi.org/10.5171/2025.4515125

Designing device authentication schemes is important due to improve the quality of service in IoT devices. A sufficiently high level of security can be achieved using hybrid encryption algorithms, which simplify the mechanisms of generating keys and ensure an increase in the computation speed (Swamy and Kota, 2020).

Currently, user authentication methods are moving towards using biometric tags (fingerprint, iris, eye retina, hand blood vessel pattern, face geometry, and voice analysis). ML (Machine Learning) and DL (Deep Learning) algorithms are also increasingly used to predict DoS (Denial of Service) or Distributed Denial of Service (DDoS) attacks. The devices in IoT networks can be vulnerable to various attacks due to predictable weak cryptographic keys. Blockchain technology also affects the applications of supply chain systems and intelligent networks.

In order to increase security in IoT networks with the 5G standard, we will design new authentication methods, identity management, trust models, and privacy protection algorithms.

Many articles present comparative analyses of communication protocols used in IoT systems. However, most of them focus on analyzing the performance and functionality of these protocols, and the comparative analysis is limited to a few selected protocols.

This article aims to present a pragmatic comparison of several of the most frequently used communication protocols in IoT systems when choosing a protocol regarding security. The following attributes are analysed as part of the security criterion: authentication, authorization, and confidentiality.

Methodology

To compare the protocols in terms of security, we use the Common Vulnerabilities and Exposures (CVE) program and Common Vulnerability Scoring System (CVSS).

CVE program allows the use of CVE identifiers during the process of exchanging information about a unique vulnerability. However, CVSS v3.x specifications determine the qualitative severity ratings (Low, Medium, High, and Critical) (NIST, 2024b; CAN, 2024). CVSS v2.0 and CVSS v3.x consist of three measures: base, time, and environment. Basic metrics generate a numerical score on a scale of 0 to 10. These metrics can be changed, taking into account time and environmental parameters. The CVSS result is also presented as a vector string. It is a compressed textual representation of the data used to calculate the result (NIST, 2024a). Table 1 shows CVSS v3.x Ratings in terms of Severity Score Range.

Table 1. Severity Score Range in CVSS v3.x Ratings

Severity	Severity Score Range
None	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

The CVSS specification allows vector strings that give an importance score of 0.0. Strings of CVSS vectors that have no impact are not evaluated (CVSS, 2023). According to the CVE vulnerability definition, a CVE record that would not impact confidentiality, integrity, or availability should not be included in the CVE program. CVSS metric values are assigned using a worst-case scenario approach. If a published vulnerability does not give details, a value of 10.0 (the highest rating) is assigned. Depending on the metrics used to determine CVSS score values, they have different meanings. Therefore, The CVSS score's usefulness is determined by the CVSS metrics used to derive that score in the prioritization process. So, the CVSS output should refer to the CVSS nomenclature version. In CVSS v4.0 specification, Base, Threat, and Environmental metric parameter values are always included in the final score calculation. Therefore, the choice of metric significantly impacts the final score. However, if threats are not specified, the final score is calculated based on the default parameters. This rule introduces a nomenclature that identifies the groups of measures included in the CVSS index value.

Characteristic of Protocols

We studied the protocols using the three primary security attributes: authentication, authorization, and confidentiality.

MQTT

The TLS (Transport Layer Security) protocol, as a newer version of the SSL (Secure Sockets Layer) protocol, determines the security of the MQTT protocol (OASIS, 2019). In addition to various authentication mechanisms, MQTT supports TLS-based encryption. However, to protect devices supporting MQTT, these services must be improved. It is especially true for the broker. It should be noted that broker misconfiguration and software vulnerabilities create many security threats that can be used to carry out attacks on IoT systems.

Potentially vulnerable processes include authentication, authorization, message delivery, verification, and encryption (Nebbione and Calzarossa, 2020). During the authentication process, the MQTT broker does not correctly verify the identity of the publisher/subscriber, which means that repeated authentication attempts are not blocked. This vulnerability allows an attacker to obtain access to MQTT devices or overload the broker so that it will fail. On authorization, the MQTT broker incorrectly sets the publish/subscribe permission. As a result of such an attack, the attacker can take control of MQTT devices. When encrypting messages, clients and servers exchange clear text messages. It allows an attacker to eavesdrop and spoof the messages being sent. This vulnerability is often used to conduct Man-in-the-Middle attacks.

DoS attack may disable the broker or cause it to crash. Such an attack involves sending large messages or messages with a high QoS level. In order to damage or turn off IoT devices, unauthorized posting of messages to damage or turn off IoT devices can be done via privileged messages. It allows an attacker to take remote control of these devices. The MQTT standard provides mechanisms for implementation to protect against such threats. These mechanisms include device and user authentication, authorization of server resource access, MQTT control packets, and application data integrity and privacy. The standard provides general recommendations, e.g., repeated authentication of long sessions, preventing all topics from subscribing and using a VPN.

The MQTT standard recommends using TLS for secure communication. However, TLS only solves some security problems. Older versions of the TLS protocol, misconfiguration, and weak encryption algorithms make the MQTT protocols vulnerable to various attacks. In order to ensure the confidentiality and integrity of messages, much of the research published has focused on developing more suitable implementations for IoT devices that support MQTT. A lightweight authentication mechanism based on a chaotic algorithm was developed to improve the mechanism of access control to restricted devices (Bali at al., 2019). Based on a modified version of the OAuth framework, the MQTT architecture can be used as two sets of credentials to access the broker (Hardt, 2020). One possible solution to this problem is to develop an automated software agent based on a state machine model. It will improve the identification of TLS vulnerabilities. Such an agent checks for possible configuration errors and, above all, verifies the certificate.

CoAP

CoAP security depends on DTLS (Datagram Transport Layer Security) (Shelby at al., 2014). DTLS uses TLS to provide protocol security for datagram-based communication. It can also provide similar security aspects. As an implementation of the TLS UDP protocol, DTLS provides equivalent security guarantees (Rescorla and Modadugu, 2020). The application of DTLS to CoAP is implemented in four security modes that differ in their authentication and key negotiation mechanisms, ranging from no security to certificate-based security.

In order to counteract various threats, CoAP provides some general security measures. It recommends using an accepted DTLS to secure CoAP nodes. Specific countermeasures are mainly applied in access control mechanisms and secure communication. For this purpose, service-level access control for low-power devices is used. It is a set of general use cases for authentication and authorization in restricted access environments (Seitz at al., 2020). This approach authenticates CoAP nodes and uses tickets to grant resource access.

Another safeguard is secure node bootstrapping. This process is necessary because its misconfiguration can compromise the entire network. This process allows the node to gather the information necessary to join the CoAP-enabled network as an authenticated node. DTLS libraries, supported by various CoAP implementations,

are part of the secure communication. They are typically used in Industrial Internet of Things systems (Iglesias-Urkia at al., 2019). The DTLS protocol has also been studied for its cryptographic algorithm (Albalas at al., 2018).

DDS

The DDS security model is based on plugins to ensure information security and can work with five SPI (Service Plugin Interfaces) (OMG, 2015). These plugins are authentication, access control, cryptography, data logging, and tagging. The authentication plugin verifies the identity of the application and the user, which starts the operations in DDS. It contains functions for user authentication processes and creates a shared secret (White at al., 2019).
The DDS protocol provides a wide range of security mechanisms. Like other messaging protocols, DDS supports TLS and DTLS. In order to provide security attributes such as confidentiality, integrity, and authenticity, the Object Management Group (OMG) DDS security specification describes an architecture based on a set of embedded plugins.

AMQP

AMQP has two security options. They are TLS and SASL (Simple Authentication and Security Layer). They provide authentication, signing, and data encryption. AMQP protocol provides an SASL framework for client authentication, and TLS provides integrity and confidentiality (ISO/IEC 19464:2014; Melnikov and Zeilenga, 2020). These security services are enabled by default in the communication process. In MQTT and CoAP protocols, it is different. However, many vulnerabilities and AMQP-based services have been discovered over the past few years. They mainly concern the broker component and the impact on processes, e.g., access control, validation, and queue management. These vulnerabilities can lead to privilege escalation, information disclosure, DoS attacks, lack of authentication and authorization, remote code execution, and network traffic interception.

OPC UA

The OPC UA application identification is a measure of reliability. Signing, encrypting/decrypting messages must also establish the issuer of the certificate (OPC, 2023). When two OPC UA-enabled applications connect for the first time, they exchange their public keys (also known as application certificates or simply OPC UA certificates) while keeping the corresponding private key. After going to the application (Client and Server), the user has to trust the public certificate of the other party. Trusting a certificate in an application is easy, meaning that another application is trustworthy and able to communicate. With asymmetric encryption, every application should have two mathematically related certificates: a private certificate used for decrypting and signing messages and a public certificate used for encryption. These keys are mathematically related, meaning a message encrypted with a public key is only decrypted with the corresponding private key.

The mathematical link of a private and public key means that one key (or part of one) was generated using the second or that both keys were generated from the same large, random prime.

The generation method depends on the algorithm used (RSA, elliptic curve), but the important thing is that there is a mathematical relationship between them. Obtaining a public key from the private key alone is practically impossible. Asymmetric encryption is much more secure than symmetric encryption, but its main disadvantage is speed. Mathematically related keys are OK, but there is a significant processing overhead to perform these operations for each message.

XMPP

The XMPP specification provides TLS authentication and encryption; XMPP servers can operate separately. An extension of XMPP that provides encryption is Off-the-Record (OTR). From the perspective of the protocol's capabilities, XMPP significantly exceeds the level of network security, as XMPP provides pluggable authentication via SASL and TLS (Cam-Winget, 2019). XMPP also works better in implementations where most XMPP clients can support SCRAM (Salted Challenge Response Authentication Mechanism). The XMPP database is a good start for implementing DNSSEC (Domain Name System Security Extensions), with available DANE (DNS-based Authentication of Named Entities) records. However, it should be noted that no specific method has yet been chosen to define an end-to-end security model.

WAMP

The WAMP protocol uses the WebSocket, which provides a simultaneous two-way communication channel over a single Transmission Control Protocol (TCP) connection. In order to ensure confidentiality (encryption), the connections are wrapped in the TLS protocol (WAMP, 2024). Therefore, security mechanisms should separate the components. In addition, man-in-the-middle attacks should be avoided. Default implementations of security mechanisms should prevent re-registration of the security procedure. Routers define administrative domains, while clients decide on the domain selection after establishing a connection. Only authenticated clients can join certain domains. For this purpose, authentication methods are used, e.g., a TLS certificate.

STOMP

Connections using the STOMP protocol use TLS, which should be configured in the broker. Since web applications often use REST APIs for user authentication and authorization, not all STOMP clients can benefit from it (STOMP, 2024). The WebSocket protocol does not specify how clients should be authenticated via HTTP. Therefore, standard HTTP headers (e.g., authorization) are used. Therefore, to establish a STOMP connection with TLS enabled, a TLS listener for the STOMP protocol must be added using configuration keys. The plugin can authenticate connections with TLS enabled. Then, it extracts the name from the client's TLS certificate (x509) without requiring a password. The security policy requires that the server is configured with the TLS options *fail_if_no_peer_cert* set to *true* and *verify* set to *verify_peer*. It forces clients to have a verifiable client certificate.

Weave

Weave protocol security is based on the defense-in-depth design principle (Weave, 2017). This principle, independent of the underlying network, includes end-to-end application security and multi-level trust domains with application-specific group keys.
OpenWeave features were based on the requirements identified by Nest when building its product ecosystem. Weave's encryption protocols are designed to fit today's IoT devices' CPU and memory limitations. Communication security is independent of the core network. Every interaction between products, applications, and cloud services is secure. Since Weave has multi-level trust domains, sensitive operations are only available to authorized devices.

HomeKit

HomeKit protocol security services protect information, create trust services, and control access (HomeKit, 2019). The security services support the following purposes:

• establishing the user's identity (authentication) and selectively granting access to resources (authorization);
• ensuring data security, both on disk and in motion over a network connection;
• ensuring the validity of the code to be executed for a specific purpose.
We can also use lower-level cryptographic resources to create new security services. We must rely on the security structure when the application uses cryptographic algorithms.

Sometimes, security mechanisms are not the best choice. The purpose of using an authorization service is to limit access to proprietary tools. These applications call system tools and software installers with access to restricted operating system areas. The Security Server daemon, running in the operating system, provides a trusted implementation of various security functions, primarily authorization. The Security Server uses the Security Agent to communicate with users regarding authorization requirements. Thus, the application verifies credentials without direct access to the system. In addition, such authorization can change authentication methods without modifying the application. Instead of the class-based version of this API, it is better to consider using the SFAuthorization class.

Authorization plugins are used to extend macOS authorization services to perform authorization in a new way. They are also used to implement a new policy that is too complex to implement fully with a database authorization policy. The API can be imported explicitly:

import Security.AuthorizationPlugin

When a plugin needs to interact with a user, we should create a subclass of the SFAuthorizationPluginView class to preserve the appearance and operation of the system authentication dialogues.

Secure data exchange is performed using the SMIME (Secure/Multipurpose Internet Mail Extensions) protocol, defined in RFC 3851. It also ensures data integrity through digital signatures and data confidentiality in the encryption process. To perform such operations, S/MIME uses the Cryptographic Message Syntax (CMS) protocol, which is defined in RFC 3852.

LwM2M

The communication security of the LwM2M protocol is ensured by the DTLS protocol, which supports various credentials (LwM2M, 2019) - a dedicated server boot LwM2M delivery of credentials and access control lists. Authorization of operations performed by the LwM2M client on object instances or resource(s) should include several conditions. First, access rights to the involved object instances must be obtained. In the "Access Control Disabled" mode, full access rights are granted to the LwM2M server, while in the "Access Control Enabled" mode, access rights should be granted to the LwM2M server. Access rights obtained through the Open Mobile Alliance are specified in the document (LwM2M, 2019). Second, it should be checked whether the access rights granted to the server are sufficient for the requested operation. Third, the access rights should be checked for successful verification, assuming the target resources support the requested operation.

Comparision

Most messaging protocols (MQTT, CoAP, XMMP, AMQP) use TLS to ensure a secure connection. Therefore, concerning applications, most attacks can be carried out on TLS. These attacks include unauthorized subscription/publication, spoofing, and unauthorized access to data. However, TLS does not provide such a function; it only provides a secure tunnel for communication. MQTT and AMQP have the Username/Password Authorization Option extension by default. AMQP provides it using SASL, while DDS has a set of authorization rules. One of the commonly used attack methods is spoofing. It involves the attacker impersonating a legitimate client and redirecting all client traffic to the attacker's computer.

We deal with a DoS attack if the attacker receives the traffic but does not forward it to the client. On the other hand, if the attacker receives the traffic and modifies it, we are dealing with a Man-in-the-Middle attack. However, we can defend ourselves against this type of attack if we enable a digital signature. However, this method is insufficient in IoT systems. In the case of unauthorized access to data, the attacker can access data stored in the broker, gateway, or edge device. In this way, by hacking a broker, gateway, or edge device, an attacker can modify messages stored in a queue or data on the edge device before it is published. If the edge device is hacked, it means a data point failure.

In contrast, if the broker or gateway is hacked, the attacker can manipulate the entire system. In this way, TLS provides a secure communication tunnel but does not protect against IoT-focused attacks. DDS provides the most comprehensive security features by default.

The security aspects, integrity and confidentiality solutions, and authentication and authorization mechanisms vary widely. The messaging protocols support standard and custom security services, while the service discovery protocols do not support any built-in security service. Therefore, the developers should implement appropriate security solutions. Encryption mechanisms are available in all messaging protocols. So, confidentiality is provided by standard services such as TLS and DTLS, while the authentication and authorization mechanisms are based on standard (i.e., SASL) or dedicated solutions. Table 2 presents a list of security services supported by messaging protocols according to the criteria: Authentication, Authorization, and Confidentiality.

Table 2. The security services supported by the messaging protocols

Protocol	Authentication	Authorization	Confidentiality
MQTT	Dedicated solution		TLS
CoAP	TLS		DTLS

DDS	Dedicated solution	Dedicated solution	TLS, DTLS
AMQP	SASL		TLS
OPC UA	OPC UA Certificate	OPC UA Certificate	OPC UA Certificate
XMPP	SASL	Dedicated solution	TLS
WAMP			TLS
STOMP			TLS
Weave	CASE, PASE		CASE, PASE
HomeKit	Dedicated solution	Dedicated solution	S/MIME
LwM2M	DTLS/TLS		DTLS/TLS, OSCORE

Developers should provide security services, but they consider these services optional. Therefore, developers approach this issue too lightly when implementing systems. Moreover, implementing security mechanisms such as end-to-end encryption is often too expensive to compete with many IoT devices' limited capabilities (e.g., throughput, computing power). Therefore, IoT devices often face protocol and network environment-specific security threats. In the process of protocol analysis, we deal with two main problems. First, these are potential threats and attacks. Second, these are protection measures, i.e., good practices and countermeasures to mitigate the effects of these attacks. Therefore, the methodological approach is to examine protocol specifications and analyse CVEs. Table 3 shows the number of vulnerabilities detected in recent years in the protocols studied.

Table 3. The number of vulnerabilities related to studied protocols

Protocol	2018 and before	2019	2020	2021	2022	2023	2024	TOTAL
MQTT	12	15	11	26	15	32	11	132
CoAP	4	4	7	1	4	2	1	23
DDS	13	3	2	6	3	10	3	40
AMQP	23	3	4	3	-	5	5	43
OPC UA	3	3	16	8	23	20	1	74
XMPP	58	1	4	3	3	2	-	71
WAMP	7	1	-	-	1	-	-	9
STOMP	3	-	-	-	1	1	-	5
Weave	-	8	3	4	-	1	1	17
HomeKit	-	-	-	2	1	-	-	3
LwM2M	1	1	1	-	1	1	-	5

The table 4 shows the number of vulnerabilities according to level of CVSS Severity.

Table 4. The number of vulnerabilities of studied protocols according to CVSS Severity (CVSS v3.x Ratings)

Protocol	CVSS Severity			
	Critical	High	Medium	Low
MQTT	36	55	32	-
CoAP	9	12	1	-
DDS	5	16	13	-
AMQP	6	12	19	2
OPC UA	7	47	11	-
XMPP	5	22	43	1
WAMP	-	5	5	-
STOMP	2	-	3	-
Weave	2	9	5	-
HomeKit	-	2	1	-
LwM2M	1	3	1	-

The CVE program has several advantages. It shows potential clients mature vulnerability management practices. Secondly, it provides information about vulnerabilities that add value. Thirdly, it audits the CVE publishing

process for vulnerabilities in a given scope and assigns CVE IDs without having to share embargoed information with other CVE Numbering Authorities. Moreover, it improves vulnerability disclosure processes.

Conclusions

Messaging protocols have built-in security services, but devices that use these protocols have some limitations. This makes implementing these protocols quite difficult. In addition, security services are often optional and must be explicitly enabled and configured by developers, leading to security risks of being misconfigured. The other serious risk comes from the lack of security services. Quite often, the sources of threats concern the lack of appropriate security services or their incorrect configuration. Although the messaging protocols offer various security services, the system is negatively affected by the incorrect configuration of these services. The lack of built-in authentication/authorization mechanisms or the use of weak security mechanisms makes IoT devices vulnerable to unauthorized access.

CVE analysis of IoT devices and services shows that improper TLS configuration or use of weak encryption algorithms creates a vulnerability that exposes sensitive data. Many of these vulnerabilities are the result of improper data validation. Our research show that vulnerabilities occur at different frequencies depending on the type of protocol used. It is important to note that security threats and vulnerabilities increase the level of risk of IoT devices, which can result in many serious consequences.

Most of the research work to date has focused on the MQTT and CoAP protocols, primarily on developing encryption algorithms that would address the limitations of IoT devices. Furthermore, few studies have addressed ways to mitigate the impact of potential attacks, given the serious security threats affecting service discovery protocols.

Summarizing the messaging protocols, we can say that the MQTT protocol supports many security services. However, these services do not cope with the protocol's possible security threats. Improvements to the DTLS protocol, studied from the perspective of the cryptographic algorithm, indicate the integration of DTLS by CoAP based on elliptic curve cryptography. It reduces the computational overhead and ROM footprint. Thus, DTLS provides confidentiality in CoAP environments. However, lightweight solutions cope pretty well with devices with limited capabilities. A typical incorrect but straightforward configuration is to use default credentials. In such a situation, the attacker uses them to control the publicly exposed broker admin interface and the entire AMQP environment. Since every DDS product and service should comply with the security specification, which is not always the case, every protocol implementation is exposed to security vulnerabilities.

The CVE program should be used to receive and manage proposals from the public that focus on improving the CVE program to serve the broader community better. Proposals should include programming principles/guidelines and new automation features that would support more effective dissemination of the CVE program.

Referencies

- Albalas F., Al-Soud M., Almomani, O., Almomani A. (2018), ‘Security-aware CoAP Application Layer Protocol for the Internet of Things using Elliptic-Curve Cryptography’. *The International Arab Journal of Information Technology*, Vol. 15, No. 3A, Special Issue 2018.
- Bali, R.S.; Jaafar, F.; Zavarasky, P. (2019), ‘Lightweight Authentication for MQTT to Improve the Security of IoT Communication’. In: Proceedings of the 3rd International Conference on Cryptography, Security and Privacy (ICCSP ’19), Kuala Lumpur, Malaysia, 19–21 January 2019.
- Cam-Winget N., Appala S., Pope S., Saint-Andre P. (2019), ‘Using Extensible Messaging and Presence Protocol (XMPP) for Security Information Exchange’. *Internet Engineering Task Force (IETF)*, https://datatracker.ietf.org/doc/rfc8600/, Updated: 2019-06-21.
- CAN (2024), CVE® Numbering Authority Operational Rules Version 4.0, https://www.cve.org/Resources/Roles/Cnas/CNA_Rules_v4.0.pdf, Approved by CVE Board on May 8, 2024.
- CVSS (2023), ‘Common Vulnerability Scoring System version 4.0’ Specification Document Version: 1.1, https://www.first.org/cvss/, Updated: 2023-11-01.
- Goworko M., Wytrebowicz J. (2021), ‘A Secure Communication System for Constrained IoT Devices—Experiences and Recommendations’, *Sensors* 2021, 21, 6906. https://doi.org/10.3390/s21206906.

- Hardt. D. (2020), 'The OAuth 2.0 Authorization Framework'. Available online: https://tools.ietf.org/html/rfc6749, Accessed: 15 March 2020.
- HomeKit, (2019), 'Accessory Protocol Specification, Non-Commercial, Version Release R2'. https://forum.iobroker.net/assets/uploads/files/1634848447889-apple-spezifikation-homekit.pdf, Updated: 2019-07-26.
- Iglesias-Urkia M.; Orive A., Urbieta, A., Casado-Mansilla D. (2019), 'Analysis of CoAP implementations for industrial Internet of Things: A survey'. *Journal of Ambient Intelligence and Humanized Computing*. 2019, Volume 10, pages 2505–2518.
- ISO/IEC 19464:2014 'Information technology — Advanced Message Queuing Protocol (AMQP) v1.0 specification', Updated: 2014-05.
- LwM2M (2019), 'Lightweight Machine to Machine Technical Specification: Core. Approved Version: 1.1.1'. https://www.openmobilealliance.org/release/LightweightM2M/V1_1_1-20190617-A/OMA-TS-LightweightM2M_Core-V1_1_1-20190617-A.pdf, Updated: 2019-06-17.
- Melnikov A., Zeilenga K. (2020), 'Simple Authentication and Security Layer (SASL)'. Available online: https://tools.ietf.org/html/rfc4422, Accessed: 13 March 2020.
- Nebbione G. and Calzarossa M.C. (2020), 'Security of IoT Application Layer Protocols: Challenges and Findings'. *Future Internet* 12(3):55, March 2020DOI: 10.3390/fi12030055.
- NIST (2024a), Vulnerability Metrics, https://nvd.nist.gov/vuln-metrics, Updated: March 19, 2024.
- NIST (2024b), 'National Vulnerability Database', Information Technology Laboratory, National Institute of Standards and Technology, https://nvd.nist.gov/, Updated: May, 29, 2024.
- OASIS (2019), 'MQTT Version 5.0', OASIS Standard, https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html, Updated: 07 March 2019.
- OMG (2015), 'Data Distribution Service (DDS) Version 1.4'. Object Management Group. Available online: http://www.omg.org/spec/DDS/1.4, Updated: April 2015.
- OPC (2023), *OPC 10000-2: UA Part 2: Security. Released 1.05.03*. https://reference.opcfoundation.org/Core/Part2/v105/docs/, Updated: 2023-12-13.
- Rescorla E., Modadugu N. (2020), 'Datagram Transport Layer Security Version 1.2'. Available online: https://tools.ietf.org/html/rfc6347, Accessed: 15 March 2020.
- Rzepka K., Szary P., Cabaj K., Mazurczyk W. (2024), 'Performance evaluation of Raspberry Pi 4 and STM32 Nucleo boards for security-related operations in IoT environments'. *Computer Networks* 242 (2024) 110252, https://doi.org/10.1016/j.comnet.2024.110252.
- Seitz, L., G. Selander., M. Mani., S. Kumar. (2020), 'Use Cases for Authentication and Authorizationin Constrained Environments'. Available online: https://tools.ietf.org/html/rfc7744, Accessed: 15 March 2020.
- Shelby Z., Hartke K., Bormann C. (2014), 'The Constrained Application Protocol (CoAP)'. *Internet Engineering Task Force (IETF),* https://datatracker.ietf.org/doc/html/rfc7252, Updated: 2014-06-26.
- Silva D., Carvalho L. I., Soares J., Sofia R.C. (2021), 'Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA'. *Applied Sciences* 2021, 11, 4879, https://doi.org/10.3390/app11114879.
- STOMP (2024), 'STOMP Protocol Specification, Version 1.2'. https://stomp.github.io/stomp-specification-1.2.html, Accessed: 2024-06-13.
- Swamy S N., Kota S. R. (2020), 'An Empirical Study on System Level Aspects of Internet of Things (IoT)'. *Computer Science*, IEEE Access DOI: 10.1109/ACCESS.2020.3029847.
- WAMP (2024), 'The Web Application Messaging Protocol'. https://wamp-proto.org/spec.html, Updated: 11 May 2024.
- Weave, (2017), 'Weave Message Layer, Protocol Specification. Revision 1.1.1'. https://github.com/openweave/openweave-core/blob/master/doc/specs/protocol-specification-weave-message-layer.pdf, Updated: 2017-09-03.
- White R., Caiazza G.; Jiang C., Ou X., Yang Z., Cortesi A., Christensen, H. (2019), 'Network Reconnaissance and Vulnerability Excavation of Secure DDS Systems'. In Proceedings of the *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Stockholm, Sweden, 17–19 June 2019.
- Wytrebowicz J., Cabaj K., Krawiec J. (2021), 'Messaging protocols for IoT Systems – A pragmatic comparison'. *Sensors* 2021, 21, 6904, https://doi.org/10.3390/s21206904.