IEEE International Symposium on Sensing and Instrumentation in IoT Era (ISSI), Shanghai, China, September 2018.

# Securing Low-Resource Edge Devices for IoT Systems

Shams Shapsough
*Computer Science and Engineering*
*American University of Sharjah*
Sharjah, UAE
b00046058@aus.edu

Fadi Aloul
*Computer Science and Engineering*
*American University of Sharjah*
Sharjah, UAE
faloul@aus.edu

Imran A. Zualkernan
*Computer Science and Engineering*
*American University of Sharjah*
Sharjah, UAE
izualkernan@aus.edu

***Abstract*— Security aspects of IoT systems are not well-understood. Therefore, the rapid adoption of IoT technologies may create many exposed computer systems with new security vulnerabilities and IoT applications from a variety of domains may face severe security holes. Edge-devices contribute significantly to security risks for IoT systems. Edge-devices are resource-constrained, wireless-enabled microcontrollers typically running primitive operating systems. The resource-constrained nature of edge devices in tandem with IoT network protocols creates many unique security challenges. This paper examines key security issues in an IoT systems with a special emphasis on edge devices. A commercial IoT edge-device using MQTT (+TLS) and CoAP (+DTLS) protocols was used to analyze the impact of these security concerns. This chosen edge device was found to be susceptible to sync attacks, data injection, passive reconnaissance, and malicious nodes. Securing nodes using TLS/DTLS resulted in only 4.7% overhead for MQTT with the varying QoS levels, and 5% for CoAP.**

***Index Terms*— Security, IoT, Edge Devices, MQTT, CoAP**

## I. Introduction

Emergence of Internet of Things (IoT) has enabled rapid adoption of applications that utilize smart sensors and heterogeneous networks in a variety of domains. Security holes in edge nodes of IoT systems are not well understood. This lack of understanding is reflected in a recent increase of cyber-attacks that compromised and exploited these edge devices. Edge devices in most IoT contexts are severely resource constrained microcontroller-based systems that have limited memory and computing power. Security concerns for edge devices are receiving attention recently because until now researcher have dedicated most time and effort into the development and deployment of novel and experimental IoT systems rather than securing them [1]. A typical edge device collects data using sensors and transmits this data to the IoT network. Edge devices need to optimize power consumption because they are often remotely located and rely on small batteries for power. Finally, the specialized communication protocols used to communicate with these edge devices in many IoT applications present unique security vulnerabilities that must be addressed.

This paper attempts to evaluate the overall security of typical edge devices in IoT systems. This is done through finding possible exploits and vulnerabilities, measuring their severity and impact on various systems, and using the acquired data to improve and reinforce security measure that ensure security while not drastically affecting operations. The evaluation is conducted according to the CIA principles of security, confidentiality, integrity and availability.

The rest of the paper is organized as follows. A summary of the various IoT communication protocols with respect to edge devices and security is discussed first. This is followed by an analysis of the security issues for edge devices. A set of experiments for one commercial edge device and results are presented next. The paper ends with a conclusion.

## II. IoT Communication Protocols

Message Queue Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), Hyper Text Terminal Protocol (HTTP) [2] and Extensible Messaging and Presence Protocol [3] (XMPP) are popular communication protocols used in many IoT systems. However, XMPP and HTTP require computational resources not available in many IoT edge devices [4]. Consequently, primarily due to resource constraints, MQTT and CoAP are more typical protocols of choice in IoT systems. CoAP implements the lighter version of request-response paradigm typified by HTTP while MQTT implements a publish-subscribe architecture. Each of these protocols are briefly described next.

### *A. MQTT*

MQTT is a low-power, low-memory messaging protocol that has been widely adopted in low-resource messaging applications [5]. The smaller packet size and lower power footprint of MQTT make this protocol suitable for communicating with resource constrained IoT edge devices. Unlike other protocols, an MQTT message is received by clients based on specific interest or topic, and not the IP address. MQTT implements a publish/subscribe architecture which makes it easy to send a message from a publisher node to numerous subscriber nodes and hence supporting one-to-many and many-to-many messaging. Messaging is based on the concept of a topic that allow a publisher or a subscriber to specify a hierarchical addressing scheme. The specific format of an MQTT message is, however, not defined and provides the developer with the flexibility of defining their own message format. MQTT operates on the TCP layer and supports the option of running on top of WebSockets. WebSockts are used in projects like Paho [6] and Hive [7]. The lightweight advantage of MQTT is, however, somewhat compromised due to the overhead of WebSockets [8-10]. MQTT supports three levels of quality of service for sending and receiving messages. The three QoS levels are:

- QoS0: message delivered at most once
- QoS1: message delivered at least once
- QoS2: message delivered exactly once

### B. CoAP

Constrained Application Protocol (CoAP) [11] is a request-response messaging protocol like HTTP developed for constrained IoT devices. This protocol implements a Representational State Transfer (RESTful) architecture [12]. CoAP enables constrained devices to use web services, combining the benefits of HTTP and MQTT. CoAP uses UDP as opposed to TCP and thus should be more power efficient than MQTT. To reduce message loss not supported by UDP, CoAP introduces a "message layer" [13] that handles packet sequencing and retransmission in case of an error. A potential challenge for CoAP is that this protocol does not provide native support for a publish-subscribe architecture often used in IoT architectures. Therefore, the nodes themselves need to keep track of the exchanged messages which adds additional workload for each node. The protocol does support a primitive publish-subscribe architecture using the observe mode. This protocol also restricts users to a fixed packet size (typically of 1 KB). To circumvent this issue, data compression or segmentation can be used. However, this workaround can increase complexity and reduce the performance of this protocol.

### C. Security in CoAP and MQTT

By default, CoAP and MQTT protocols do not use any security layer. However, these protocols do offer the option of extra security layers based on TLS [7, 14].

MQTT supports TLS as an optional security layer [7]. However, using secure communication with TLS requires significant additional resources terms of CPU and bandwidth usage. In a secure communication, a TLS handshake is required to initiate a session. Both client and server agree on the cipher suite and the TLS version to be used. This process is slightly resource heavy because the client only needs to establish the handshake once per session making this protocol better than other competing protocols like HTTPS. In addition to handshaking, additional buffers need to be allocated for TLS. This increases memory requirements for the edge node. The choice of cipher suite (decided during the handshake) is an essential concern when using TLS. Depending on the suite selected, the TLS overhead varies. Therefore, it is critical to select the cipher suite that is suitable for specific requirements of the edge node. MQTT also provides a pre-shared key over TLS as well [15]. This is a much lighter approach than traditional TLS, but not as commonly used. Finally, MQTT also supports X509-certification [7]. This feature allows clients to authenticate and verify the identity of the message broker to avoid server spoofing.

As opposed to MQTT, CoAP optionally offers a lighter version of TLS called Datagram Transport Layer Security (DTLS) [14]. For CoAP, both DTLS and IPSec are available through third-party layers [16]. In addition to DTLS and IPSec, CoAP also offers a native CoAP-security layer that provides similar authentication and data security options as DTLS and IPSec but with significantly lower resource requirements [17].

## III. SECURING EDGE DEVICES

Edge devices directly interact with the physical environment using tags, sensors, actuators and embedded devices. As a critical component of any IoT application, the edge layer provides an exposed target to attackers where they can gain access and compromise or take down the entire system. Attacks targeting edge devices can be categorized into denial of services, information gathering or eavesdropping and planting malicious nodes. Each type is briefly described below.

### A. Denial of Service (DoS)

Denial of Service (DoS) attacks aim to interrupt a systems operation and block access to its services by overwhelming it in various ways. In the case of edge devices, this is done in three ways: battery draining, sleep deprivation, and outage attack.

*Battery draining* aims to exploit the fact that most edge devices rely on small power units for operation due to size constraints. Attackers try to deplete the battery of an edge device by any means possible. For example, this might involve forcing the edge device to execute power-consuming subroutines. If a node is integral to the system and is difficult to physically access (like the ones used to monitor natural resources) this could take down the entire system and render it useless [18]. This makes battery draining a very serious issue.

The second method under DoS for edge devices is *sleep deprivation*. In this type of attack, the attacker sends numerous requests that appear to be legitimate. This forces the node to reply or address those requests. This stops the node from sleeping, and hence not conserving energy [19].

A third method for DoS for edge devices is *outage attack*. Here the attacker targets an administrative device or a master node with code-injection, physical tampering, sleep deprivation or battery draining. Once the administrative node stops functioning, the other nodes relying on it are rendered useless.

### B. Information Gathering and Eavesdropping

The second group of attacks is concerned with reconnaissance. Here, the attacker tries to collect as much information as possible on the edge device. This information is critical to the device and provides insight to its status. In the least, this type of attack may lead to serious privacy issues. An example of this type of attack is the side-channel attack in which a node reveals information about its operation through an electromagnetic signature or power consumption and can be used to break or compromise the cryptography [20]. Moreover, attackers can capitalize on the limitations of the technologies used in these systems. For example, by default MQTT and CoAP are not encrypted and are widely used. Eavesdropping and sniffing the plain text communication compromises data and might lead to fatal attacks to the system like battery draining.

### C. Planting Malicious Nodes

The third category of attacks is closely connected to reconnaissance. In this type of attack the information known about a systems operation allows the attacker to falsify data,

plant malicious nodes, replicated existing nodes and compromise the entire system.

## IV. EXPERIMENTS

This section describes a series of experiments to evaluate vulnerabilities of edge devices. The purpose of these experiments was to evaluate security aspects of MQTT and CoAP protocols when implemented on one resource limited IoT edge device available commercially.

### *A. The Edge Device*

The edge device used for these experiments was the Particle Photon board which is an emerging commercial development board for implementing edge devices in IoT systems. The Photon board supports ease of use, built-in WIFI capabilities, cloud-based development and platform, and over the air update (OTA). These features make this board a good edge-device candidate for implementing a wide range of IoT applications. The hardware specification for the Particle Photon hardware used in these experiments are shown in Table I.

TABLE I. HARDWARE SPECIFICATION FOR THE EDGE DEVICE

| ***Specification*** | ***Particle Photon*** |
|---|---|
| Processor | 32-bit ARM Cortex-M3 120MHz |
| SRAM | 128kB |
| Networking | 802.11b/g/n, soft-AP |
| Storage | 1MB flash |
| I/O | 24pins (GPIO/ PWM/ USB/ CAN/ SPI/ I2C/ I2S/ ADC/ DAC) |
| On-board peripherals | RGB LED |

### *B. Experimental Setup*

Fig. 1 shows the experimental testbed created for conducting the experiments. It is important to note that the attacks tested on the system were generic and applicable to other IoT development boards besides the Photon board considered here.

As shown in Fig. 1, the setup consisted of an edge node connected to a message broker using WIFI. The edge devices were all Photon devices connected to sensors and actuators and using the following options: MQTT (using all three QoS options), MQTT with TLS, CoAP, and CoAP with DTLS. A YoctoAmp [21] device was connected to each of the edge nodes to measure the power consumption for each node to compare the resource requirements for each configuration. A messaging broker was needed to implement the reference networking architectures (i.e., MQTT). An attacker node was used to perform penetration testing and gather data to/from the connected edge nodes.

A generic IoT edge node functionality was implemented for each edge device. This type of functionality is typical to smart grid and resource monitoring systems and is a reasonable simulation of actual IoT systems. The operation of each edge device was as follows:

- Send data to a consumer/control node and sleep.
- Based on the received data, the control device issues commands to each node.
- The commands were: to continue normal operation or execute subroutine to enable actuators and complex operations, or to kill code and cease operation.

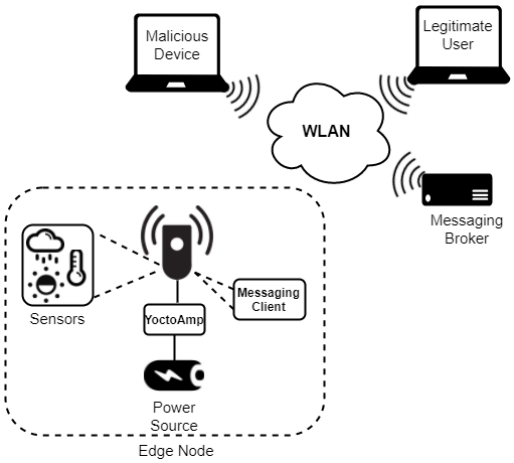


Fig. 1. Experimental setup

### *C. Results*

The list of attacks the system was subjected to and the tools used to simulate the attacks and their outcome are shown in Table II. Results in Table II can be summarized as follows:

- Neither of the options (e.g., MQTT+TLS) were subservient to attacks like ping of death or malware.
- Introducing TLS or DTLS resulted in mediating attacks like sleep deprivation, packet sniffing, and node replication only.
- All options were still susceptible to sync attacks, data injection, passive reconnaissance and malicious nodes.
- Code injection was not possible on this edge node.

Table II also lists the various potential countermeasures that could be taken for each of the attacks.

Fig. 2 shows the average power consumption of an edge node over a full day of continuous use. Surprisingly, for the Photon board, MQTT's implementation was more energy-efficient than CoAP. Adding security layer, however, only added an average of 4.7% overhead for MQTT with the varying QoS levels, and 5% for CoAP which is quite reasonable. In addition, as expected, the power requirements for MQTT, with and without TLS increase as well when the QoS level was increased.

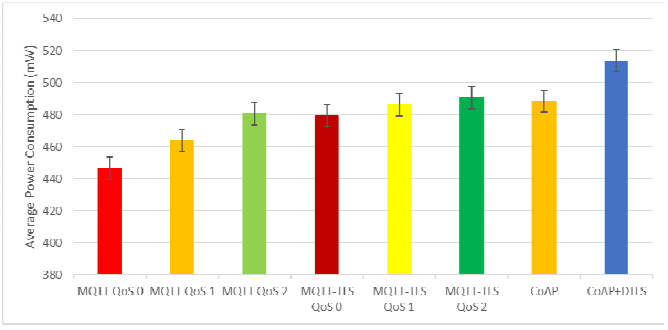


Fig. 2. Energy consumption of various architectural variants

TABLE II. RESULT OF VARIOUS ATTACKS ON FOUR TYPES OF NODES

| Vulnerability | Types of Edge-Nodes | | | | Tools | Countermeasure |
|---|---|---|---|---|---|---|
| | *MQTT* | *MQTT +TLS* | *CoAP* | *CoAP+ DTLS* | | |
| Battery Draining | S* | F | S | F | Python Script | Cryptographic Scheme, IDS, Authentication, Role-Based authorization [23,25] |
| Sleep Deprivation | S | F | S | F | Python Script | Cryptographic Scheme, IDS, Authentication, Role-Based authorization [23,25] |
| Sync | S | S | S | S | Metasploit | IDS, Firewall [23,25] |
| Ping of Death | F | F | F | F | Hyenae | IDS, Firewall [23,25] |
| Packet Sniffing | S | F | S | F | Wireshark, Metasploit | Cryptographic Scheme [23,25] |
| Code Injection [22] | F | F | F | F | Not possible on Photon | N/A |
| Data Injection | S | S | S | S | Python Script | Cryptographic Scheme AND Authentication. Role-Based authorization [22,24] |
| Basic Reconnaissance | S | S | S | S | IoTSeeker, Nmap | Firewall, IPS [23,25] |
| Node Replication | S | F | S | F | Python Script | Cryptographic Scheme AND Authentication [23,25] |
| Camouflage | S | S | S | S | Python Script | Cryptographic Scheme AND Authentication [23,25] |
| Corrupted Node | S | S | S | S | Python Script | Cryptographic Scheme AND Authentication [23,25] |
| Malware | F | F | F | F | IoTroops | Antivirus [22-25] |

* S means attack was successful, and F means that the attack failed.

## V. CONCLUSION

As the use of IoT enabled devices increases, so does the attack surface area and the severity of attacks causing new vulnerabilities to come to light and in some cases exploited routinely. The low-cost, low-power nature of many IoT edge-devices offers a challenge for both developers and security researchers and this research is a first step in addressing these challenges.

## REFERENCES


[1] C. (Defta) Costinela-Luminiţa and C. (Iacob) Nicoleta-Magdalena, "E-learning Security Vulnerabilities," Procedia - Soc. Behav. Sci., vol. 46, pp. 2297–2301, Jan. 2012.

[2] IETF, "HTTP - Hypertext Transfer Protocol," 1999.

[3] "XMPP | XMPP Main." [Online]. Available: https://xmpp.org/. [Accessed: 09-Sep-2017].

[4] A. Talaminos-Barroso, M. A. Estudillo-Valderrama, L. M. Roa, J. Reina-Tosina, and F. Ortega-Ruiz, "A Machine-to-Machine protocol benchmark for eHealth applications – Use case: Respiratory rehabilitation," Comput. Methods Programs Biomed., vol. 129, pp. 1–11, Jun. 2016.

[5] L. Zhang, "Building Facebook Messenger," ed: Facebook, 2011

[6] Paho. Available: https://eclipse.org/paho/

[7] HiveMQ: Enterprise MQTT Broker. Available: http://www.hivemq.com/

[8] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S - A publish/subscribe protocol for Wireless Sensor Networks," in 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), pp. 791-798.

[9] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014.

[10] I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, "Choices for interaction with things on Internet and underlying issues," Ad Hoc Networks, vol. 28, pp. 68-90, 2015.

[11] C. Bormann, "CoAP - Constrained Application Protocol," T.-Z. I. u. Informationstechnik, Ed., ed, 2014.

[12] M. Castro, A. J. Jara, and A. F. Skarmeta, "Enabling end-to-end CoAP-based communications for the Web of Things," Journal of network and computer applications, vol. 59, pp. 230-236, 01/2016 2016.

[13] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An Application Protocol for Billions of Tiny Internet Nodes," IEEE Internet Computing, vol. 16, pp. 62-67, 2012.

[14] N. Modadugu and E. Rescorla, "Datagram Transport Layer Security." [Online]. Available: https://tools.ietf.org/html/rfc4347. [Accessed: 19-Apr-2017].

[15] H. Tschofenig and P. Eronen, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)." [Online]. Available: https://tools.ietf.org/html/rfc4279. [Accessed: 19-Apr-2017].

[16] R. Atkinson and S. Kent, "IP Encapsulating Security Payload (ESP)." [Online]. Available: https://tools.ietf.org/html/draft-ietf-ipsec-esp-v2-05. [Accessed: 19-Apr-2017].

[17] A. Yegin and Z. Shelby, "CoAP Security Options." [Online]. Available: https://tools.ietf.org/html/draft-yegin-coap-security-options-00. [Accessed: 19-Apr-2017].

[18] A. M. Nia and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," IEEE Trans. Emerg. Top. Comput., vol. PP, no. 99, pp. 1–1, 2016.

[19] T. Martin, M. Hsiao, D. Ha, and J. Krishnaswami, "Denial-of-service attacks on battery powered mobile computers," in Proc. IEEE 2nd Conf. Pervasive Computing and Communications, 2004, pp. 309–318.

[20] J. Ambrose, A. Ignjatovic, and S. Parameswaran, Power Analysis Side Channel Attacks: The Processor Design-level Context. Omniscriptum Gmbh & Company Kg., 2010.

[21] "Yocto-Amp - Tiny isolated USB ammeter (AC/DC)." [Online]. Available: http://www.yoctopuce.com/EN/products/usb-electrical-sensors/yocto-amp. [Accessed: 11-Sep-2017].

[22] A. Francillon and C. Castelluccia. Code injection attacks on harvard-architecture devices. In ACM CCS 2008, pages 15–26. ACM, 2008.

[23] H. Suo, J. Wan, C. Zou and J. Liu, "Security in the Internet of Things: A Review," 2012 International Conference on Computer Science and Electronics Engineering, Hangzhou, 2012, pp. 648-651.

[24] J. Habibi, A. Gupta, S. Carlsony, A. Panicker, and E. Bertino. Mavr: Code reuse stealthy attacks and mitigation on unmanned aerial vehicles. In IEEE ICDCS 2015, pages 642–652. IEEE, 2015.

[25] A. Mosenia and N. K. Jha, "A Comprehensive Study of Security of Internet-of-Things," in IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 4, pp. 586-602, Oct.-Dec. 1 2017.