Security vulnerabilities and cyber threat analysis of the AMQP protocol for the internet of things

Ian Noel McAteer
Edith Cowan University

Muhammad Imran Malik
Edith Cowan University

Zubair Baig
Edith Cowan University

Peter Hannay
Edith Cowan University

DOI: 10.4225/75/5a84f4a695b4c
McAteer, I., Malik, M.I., Baig, Z., & Hannay, P. (2017). Security vulnerabilities and cyber threat analysis of the AMQP protocol for the internet of things. In Valli, C. (Ed.). (2017). The Proceedings of 15th Australian Information Security Management Conference, 5-6 December, 2017, Edith Cowan University, Perth, Western Australia. (pp.70-80).
This Conference Proceeding is posted at Research Online.
https://ro.ecu.edu.au/ism/203

SECURITY VULNERABILITIES AND CYBER THREAT ANALYSIS OF THE AMQP PROTOCOL FOR THE INTERNET OF THINGS

Ian Noel McAteer[1], Muhammad Imran Malik[1], Zubair Baig[1,2], Peter Hannay[1,2]
[1]School of Science, [2]Security Research Institute, Edith Cowan University, Perth, Australia
imcateer@our.ecu.edu.au, mimalik@our.ecu.edu.au, z.baig@ecu.edu.au, p.hannay@ecu.edu.au

Abstract

The Internet of Things (IoT) expands the global Internet-connected network to encompass device-to-device, device-to-server, and server-to-server connectivity for an ever-increasing variety of end-user devices. IoT remains a somewhat amorphous entity, with little in the way of coordinated development, and is undermined largely by a manufacturer-driven scramble to be first-to-market with the latest innovation. Communication between IoT devices/servers relies on underlying protocols, which must be efficient and effective to establish and maintain reliability and integrity of data transfer. However, the lack of coordination during IoT's expansion has resulted in a variety of communications protocols being developed. AMQP (Advanced Message Queuing Protocol) originated from the financial sector's requirement for an improved messaging system that was fast, reliable and independent of end-user platform configurations. AMQP is an open-source server-to-server communications protocol which allows the addition of user-specific extensions. The software coding of such end-user-developed modules can be insufficient regarding threat-mitigation and can make the end product vulnerable to cyber-attack. Through this paper, we present vulnerability and threat analysis for AMQP-based IoT systems.

Keywords: AMQP (Advanced Message Queuing Protocol), AMQP Vulnerabilities, Application Layer Protocols, Internet of Things (IoT), IoT Architecture

INTRODUCTION

For more than a decade, the desire to have an increasing range of devices controllable via wired or wireless communications has seen an exponential rise in the number of Internet-of-Things (IoT) devices being available for the workplace and the home. No sector of our lives appears to be immune from the IoT invasion, whether it be military, medical, industrial, or domestic. The Internet-of-Everything (IoE) is rapidly becoming a reality.

The term IoT is believed to have been originally coined in 1999 (Ashton, 2009), though in hindsight devices such as the wireless telegraph can be considered to be IoT devices dating back to the early to mid-19th century (Foote, 2016). In the early 2000s, the first commercial domestic appliance IoT device, the Internet refrigerator, began being shipped by some manufacturers (Osisanwo, Kuyoro, & Awodele, 2015). During 2008, the number of devices connected to the Internet was considered to exceed the number of people on earth (Evans, 2011). In 2011, IPv6 protocol became available which allowed an address space of 2^{128} unique IP addresses. This vast number prompted Steven Leibson of the Computer History Museum, Mountain View, CA at that time, to say "we could assign an IPv6 address to EVERY ATOM ON THE SURFACE OF THE EARTH, and still have enough addresses left to do another 100+ earths!" (Bhalla, 2012). By 2025, it is estimated that 75 billion IoT devices will make up a market worth between US$3.9 trillion to US$11.1 trillion (Jacobs, 2017).

As with other fields of technological development, the security of such IoT devices has predominantly taken a backseat in a rush to be first to market with a new product (Arias, Wurm, Hoang, & Jin, 2015). For this reason, IoT has been considered as the future of the Internet (Gubbi, Buyya, Marusic, & Palaniswami, 2013) or as the death of the Internet (Vaughan-Nichols, 2016) depending on one's point of view. As part of this research, we present IoT as an imminent reality which is making a positive impact on modern day living and hence the paper will discuss the following:

1. Different protocols involved in enabling IoT concept.
2. The history of the AMQP protocol, along with various features.
3. The design architecture of AMQP.
4. The known vulnerabilities of the protocol.
5. Threats posed by these vulnerabilities.

BACKGROUND

IoT consists of much more than just connected end-user devices. Underlying an IoT operation is a framework to allow data extraction from each device for analysis, and data insertion back to each device for control purposes. Such frameworks apply to all IoT deployments in use today, and will equally apply to future IoT deployments that have not yet been developed, even for so-called independent or autonomous devices.

TechBeacon (2017) describes such an architecture as a four-stage process:

1. Stage 1 – IoT devices consisting of wireless sensors and actuators.
2. Stage 2 – Sensor data aggregation systems and analogue-to-digital conversion.
3. Stage 3 – Edge IT systems for data pre-processing and transmission.
4. Stage 4 – Back-end data systems for analysis, management, and storage.

Bradicich (2015) clarifies these four stages graphically, as shown in Figure 1.

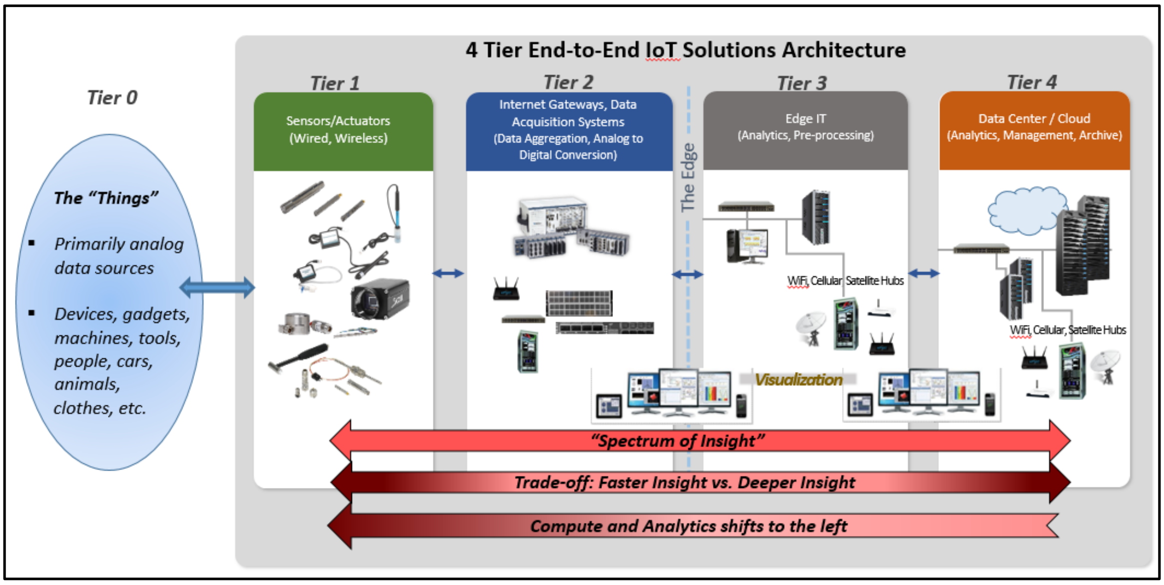
4 Tier End-to-End IoT Solutions Architecture
Tier 0
The "Things"
Primarily analog data sources
Devices, gadgets, machines, tools, people, cars, animals, clothes, etc.
Tier 1
Sensors/Actuators (Wired, Wireless)
Tier 2
Internet Gateways, Data Acquisition Systems (Data Aggregation, Analog to Digital Conversion)
The Edge
Tier 3
Edge IT (Analytics, Pre-processing)
WiFi, Cellular, Satellite Hubs
Tier 4
Data Center / Cloud (Analytics, Management, Archive)
WiFi, Cellular, Satellite Hubs
Visualization
"Spectrum of Insight"
Trade-off: Faster Insight vs. Deeper Insight
Compute and Analytics shifts to the left

Figure 1. IoT Architecture Stages (Bradicich, 2015)

Overview of Application Layer Protocols in IoT

AMQP (AMQP, 2017a) is an open-standard application protocol middleware which enables server-to-server message communication. It is independent of both the platform and language that each of the end-servers may be using. Flow-controlled communication enables delivery options, such as at-most-once, at-least-once, or exactly-once. Delivery authentication is provided by SASL (Simple Authentication and Security Layer). Encryption is provided by TLS (Transport Layer Security), which is the successor to SSL (Secure Sockets Layer).

CoAP (Constrained Application Protocol) (CoAP, 2017) is an M2M (Machine to Machine) web transfer protocol for use with constrained nodes and constrained networks. It is based on the REST (Representational State Transfer) model. Since this is also true of HTTP, seamless connection to a web client interface is possible. CoAP uses minimal resources while providing encryption through DTLS (Datagram Transport Layer Security) equivalent to 3072-bit RSA.

DDS (Data Distribution Service) (DDS, 2017) is a protocol designed to interconnect devices and has its origins in high-performance environments, such as the military, industry and embedded systems. Its rapid delivery of messages simultaneously to an array of receivers make it suitable for real-time analytics and sensor monitoring.

IBM developed MQTT (Message Queuing Telemetry Transport) (MQTT, 2017) in 1999 and since 2013 has become an OASIS (Organization for the Advancement of Structured Information Standards) standard. It is an M2M IoT connection protocol, which uses publish/subscribe messaging transport methodology to connect a device to a server. MQTT's lightweight architecture makes it suitable for situations where communications bandwidth is limited, such as remote locations or via satellite links.

RESTFUL (Service Architecture, 2017) provides a set of standards for communication between computer systems on the Internet. Its underlying HTTP-compliant architecture enables the communication and transfer of data between IoT devices using this protocol. It uses the same HTTP methods of GET, POST, PUT, and DELETE to conduct request/response interactions.

SMQTT (Secure Message Queuing Telemetry Transport) (SMQTT, 2017) is an extension to MQTT which encrypts a message before publication by the broker. The encrypted message is sent to multiple subscriber nodes where the message is decrypted using the same master key.

WebSocket protocol (WebSocket, 2017) provides full-duplex communication between clients and a remote server on a single TCP connection. WebSockets can be implemented in both web browsers, servers and indeed any application making use of the client/server paradigm. Apart from the initial handshake with HTTP servers, WebSocket is an independent protocol that maintains a two-way exchange between client and server using TCP port 80 or TLS port 443 for encrypted traffic.

XMPP (Extensible Messaging and Presence Protocol) (XMPP, 2017) is one of four Instant Message (IM) protocols which has developed to satisfy the rapidly expanding information society's need for short message services. XMPP's use of Extensible Markup Language (XML) overcomes prior difficulties in connecting an IM system with a non-IM system. Several large public IM services, such as LJ Talk, Nimbuzz, and HipChat exclusively use XMPP. Other popular IM applications like WhatsApp, Gtalk and Facebook Chat use XMPP on their back-end servers.

AMQP Protocol History and General Features

AMQP was first proposed in 2003 by John O'Hara of JPMorgan Chase in London. In 2005 JPMorgan Chase commenced taking other firms on board to develop a working group for the project. These firms initially included the likes of Cisco Systems and Red Hat, but within a few years, the AMQP working group had expanded to include 23 partners.

In August 2011 the AMQP working group had attained OASIS membership, and two months later AMQP 1.0 was released at a conference in New York. Successful demonstrations of software running the protocol resulted in an OASIS Technical Committee being formed to develop the protocol into an international open standard.

In April 2014 OASIS AMQP was granted ISO and IEC International Standard approval, under the designation ISO/IEC 19464. Despite its development predominantly within the financial sector, AMQP has seen wide adoption as a server-to-server communications protocol within the IoT environment. AMQP is one of the few highly used application layer protocols with its prominent implementation India's Aadhaar project, considered as one of the world's largest biometric databases (Varma, 2010). AMQP is also being used in the Ocean Observatories Initiative infrastructure that uses sensor nets to bring readings ashore from ocean platforms and a global publisher-subscriber network to disseminate readings (Meisinger, 2010). Other notable implementations of AMQP include The Deutsche Börse, JPMorgan, NASA, AT&T (AMQP, 2017b).

AMQP offers the following protocol features (Ross, 2012):

Session Multiplexing

- Multiple Sessions can be carried over a single connection
- Sessions have independent message sequencing and flow control
- Interleaving of large messages

Full Duplex, Asynchronous Communication

- Within a session, messages can flow independently in both directions

Semantics of Message Hand-off

- AMQP formally defines the semantics of message transfer and settlement/acknowledgement
- Messages can be moved or copied
- Delivery guarantees:
  - Best Effort (Fire and Forget)
  - At Least Once

    - Exactly Once
- Transactional Message Transfer
    - Local Transactions between Endpoints
    - Distributed Transactions

Data Security

- TLS/SSL
    - Encryption only, or
    - Encryption and authentication by x.509 certificates
    - Most APIs provide this option via the text of a URL: amqp://hostname vs amqps://hostname
- Extensible Authentication/Security Mechanisms
    - SASL – Simple Authentication and Security Layer
    - Supports numerous mechanisms:
        - ANONYMOUS
        - PLAIN
        - DIGEST-MD5
        - GSSAPI
        - NTLM

Flow Control

- Limits the number of messages that a producer can transfer at a time
- Greatly simplifies some difficult architectural problems
    - Many data sources sending to a data sink at the same time

Serialisation of Structured Data

- AMQP defines the wire-line format for data types
- The application designer does not need to be concerned with
    - The Endian order of a peer system
    - The Word Size of a peer system
- Rich API support is provided to access this feature
    - A Python program can send a message containing a Python Dictionary
    - A Java program can receive the message as a Hash Map (or a JMS Map Message)
    - A C++ program can receive the same message as an std::map

Message Metadata

- Like HTTP, AMQP allows messages to be annotated with headers
- Any number of application-specific headers may be placed in a message
- Headers are carried separately from the message body (which may be encrypted, encoded, and compressed)

Transport Independence

- Messages can be sent between services in many ways, all hidden from one's business logic

AMQP ARCHITECTURE

AMQP Design

The primary objective of AMQP was to replace existing proprietary and non-interoperable messaging systems that were considered to be hampering communications in the financial sector (Vermesan & Friess, 2014). It aimed to enable messaging at enterprise level between two platform systems of any configuration, provided that each of those platform systems, and the libraries that they are using, was AMQP-compliant (Indrasiri, 2016).

In AMQP, a shared queue space or broker is provided, which all participating applications can access. A message is sent from a client via a publisher to an exchange within the broker. Each message contains a routing key, which will be used by the broker to either assign the message to a particular queue, distribute the message to several queues, duplicate every message to multiple queues or distribute multiple messages to chosen queues as defined by the routing key.

A subscriber then receives its authorised messages from queues for delivery to its end client. Since each message is assigned to one particular subscriber, even though the same message may have been duplicated for another subscriber, only one subscriber can receive a particular message from any of the queues present.

Individual messages are tracked and accounted for to ensure that all messages are delivered as intended. For this to occur, all endpoints must acknowledge receipt of each message.

Richardson (2008) graphically represents the methodology of the AMQP protocol, as shown in Figure 2.

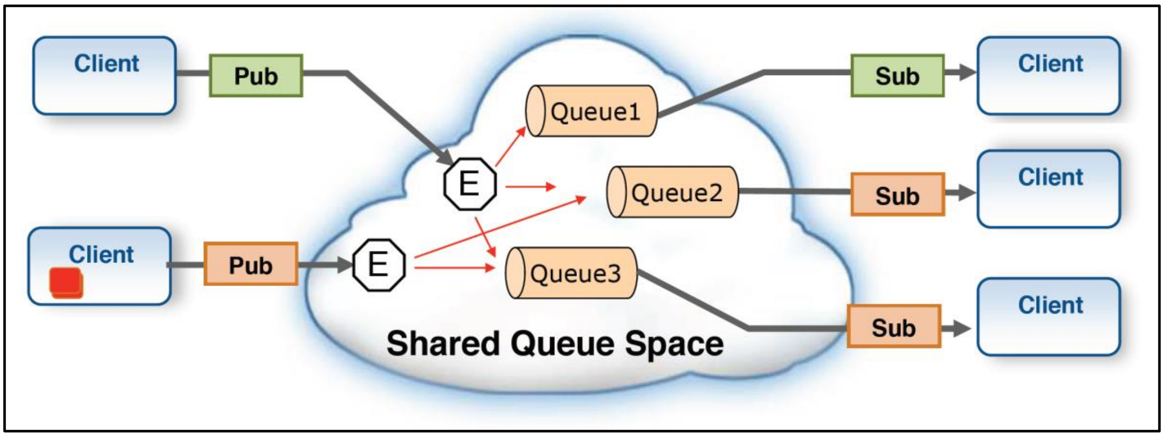

Figure 2. AMQP Methodology (Richardson, 2008)
Key: Pub = Publisher, E = Exchange, Sub = Subscriber

AMQP uses underlying TCP/IP-compatible protocols at the network and transport levels, which provides wire-level interoperability and reliability. It is a messaging solution offering flexibility, which Richardson (2008) summaries as follows:

1. any language (C, C++, C#, Python, Java, Javascript, Erlang, Lisp, Ruby, Tcl, PHP, …)
2. any model (native, .NET WCF, JMS, Mule, can do Caching)
3. any payload (binary, XML, SOAP, JSON, …)
4. any transport (TCP, SCTP, UDP, HTTP, …)
5. any scenario (desktop, router, wan, mobile, mesh, cloud)

Figure 3 shows a comparison between same-platform and cross-platform server integration, the latter being the issue AMQP was developed to resolve.

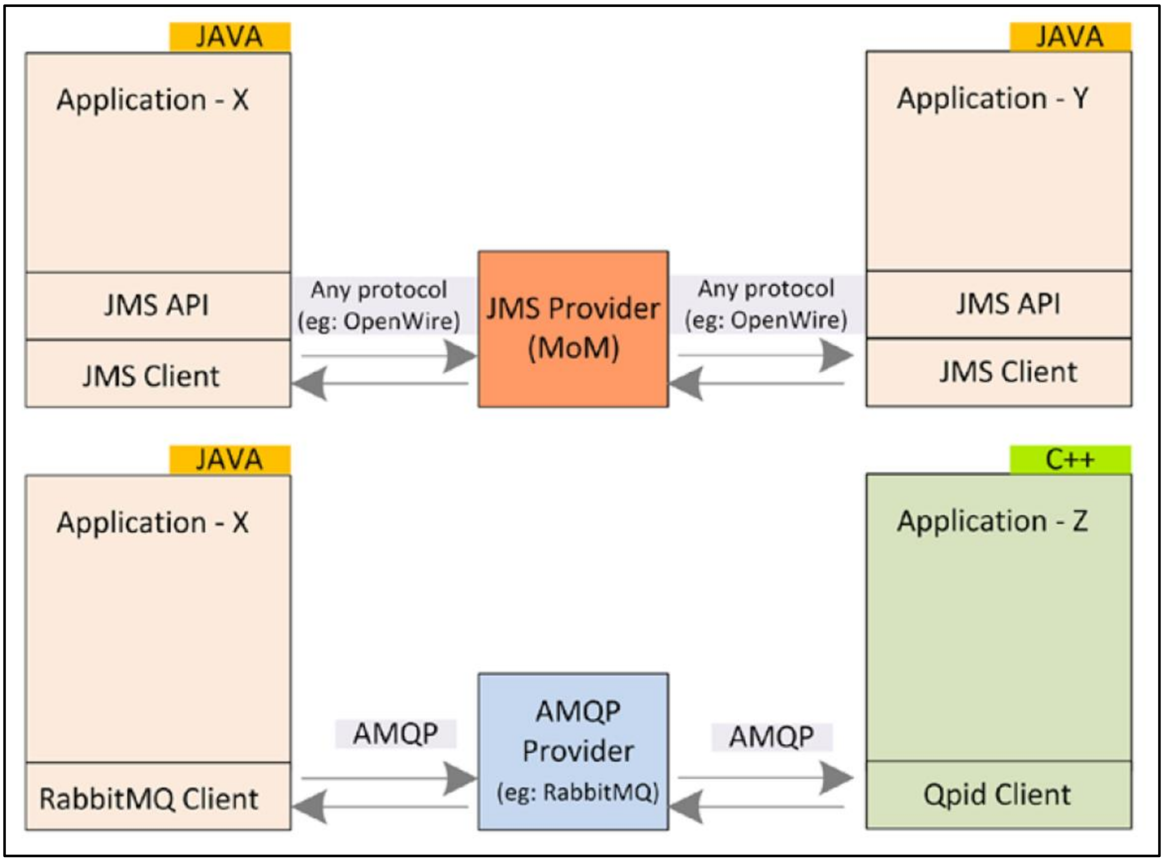

Figure 3. Same-Platform Integration via JMS vs Cross-Platform Integration via AMQP (Indrasiri, 2016)

Known Vulnerabilities

The research explored Common Vulnerabilities and Exposures (CVE) databases for the known vulnerabilities of AMQP. While searching and examining the CVE database for the terms *AMQP* and *IoT*, 17 vulnerabilities were identified with a disclosure date within the past decade:

Table 1. CVE Search Results for "AMQP IoT" (Common Vulnerabilities and Exposures, 2017)

Date Created	CVE ID	Details
18-Apr-2017	CVE-2017-7911	A Code Injection issue was discovered in CyberVision Kaa IoT Platform, Version 0.7.4. An insufficient-encapsulation vulnerability has been identified, which may allow remote code execution.
23-Mar-2017	CVE-2017-7243	Eclipse tinydtls 0.8.2 for Eclipse IoT allows remote attackers to cause a denial of service (DTLS peer crash) by sending a "Change cipher spec" packet without pre-handshake.

Date Created	CVE ID	Details
09-Mar-2017	CVE-2017-6780	A vulnerability in the TCP throttling process for Cisco IoT Field Network Director (IoT-FND) could allow an unauthenticated, remote attacker to cause the system to consume additional memory, eventually forcing the device to restart, aka Memory Exhaustion. The vulnerability is due to insufficient rate-limiting protection. An attacker could exploit this vulnerability by sending a high rate of TCP packets to a specific group of open listening ports on a targeted device. An exploit could allow the attacker to cause the system to consume additional memory. If enough available memory is consumed, the system will restart, creating a temporary denial of service (DoS) condition. The DoS condition will end after the device has finished the restart process. This vulnerability affects the following Cisco products: Connected Grid Network Management System, if running a software release prior to IoT-FND Release 4.0; IoT Field Network Director, if running a software release prior to IoT-FND Release 4.0. Cisco Bug IDs: CSCvc77164.
17-Jul-2017	CVE-2017-11408	In Wireshark 2.2.0 to 2.2.7 and 2.0.0 to 2.0.13, the AMQP dissector could crash. This was addressed in epan/dissectors/packet-amqp.c by checking for successful list dissection.
24-May-2016	CVE-2016-4974	Apache Qpid AMQP 0-x JMS client before 6.0.4 and JMS (AMQP 1.0) before 0.10.0 does not restrict the use of classes available on the classpath, which might allow remote authenticated users with permission to send messages to deserialize arbitrary objects and execute arbitrary code by leveraging a crafted serialized object in a JMS ObjectMessage that is handled by the getObject function.
02-May-2016	CVE-2016-4432	The AMQP 0-8, 0-9, 0-91, and 0-10 connection handling in Apache Qpid Java before 6.0.3 might allow remote attackers to bypass authentication and consequently perform actions via vectors related to connection state logging.
29-Jan-2016	CVE-2016-2173	org.springframework.core.serializer.DefaultDeserializer in Spring AMQP before 1.5.5 allows remote attackers to execute arbitrary code.
01-Jul-2015	CVE-2015-5240	Race condition in OpenStack Neutron before 2014.2.4 and 2015.1 before 2015.1.2, when using the ML2 plugin or the security groups AMQP API, allows remote authenticated users to bypass IP anti-spoofing controls by changing the device owner of a port to start with network: before the security group rules are applied.
25-May-2015	CVE-2015-4080	The Kankun Smart Socket device and mobile application uses a hardcoded AES 256 bit key, which makes it easier for remote attackers to (1) obtain sensitive information by sniffing the network and (2) obtain access to the device by encrypting messages.
09-May-2015	CVE-2015-2247	Unspecified vulnerability in Boosted Boards skateboards allows physically proximate attackers to modify skateboard movement, cause human injury, or cause physical damage via vectors related to an "injection attack" that blocks and hijacks a Bluetooth signal.
07-Jan-2015	CVE-2015-0862	Multiple cross-site scripting (XSS) vulnerabilities in the management web UI in the RabbitMQ management plugin before 3.4.3 allow remote authenticated users to inject arbitrary web script or HTML via (1) message details when a message is unqueued, such as headers or arguments; (2) policy names, which are not properly handled when viewing policies; (3) details for AMQP network clients, such as the version; allow remote authenticated administrators to inject arbitrary web script or HTML via (4) user names, (5) the cluster name; or allow RabbitMQ cluster administrators to (6) modify unspecified content.

Date Created	CVE ID	Details
09-Nov-2014	CVE-2014-8711	Multiple integer overflows in epan/dissectors/packet-amqp.c in the AMQP dissector in Wireshark 1.10.x before 1.10.11 and 1.12.x before 1.12.2 allow remote attackers to cause a denial of service (application crash) via a crafted amqp_0_10 PDU in a packet.
10-Apr-2014	CVE-2014-2814	Microsoft Service Bus 1.1 on Microsoft Windows Server 2008 R2 SP1 and Server 2012 Gold and R2 allows remote authenticated users to cause a denial of service (AMQP messaging outage) via crafted AMQP messages, aka "Service Bus Denial of Service Vulnerability."
12-Oct-2010	CVE-2009-5005	The Cluster::deliveredEvent function in cluster/Cluster.cpp in Apache Qpid, as used in Red Hat Enterprise MRG before 1.3 and other products, allows remote attackers to cause a denial of service (daemon crash and cluster outage) via invalid AMQP data.
21-Aug-2008	CVE-2012-4458	The AMQP type decoder in Apache Qpid 0.20 and earlier allows remote attackers to cause a denial of service (memory consumption and server crash) via a large number of zero width elements in the client-properties map in a connection.start-ok message.
21-Aug-2008	CVE-2012-4446	The default configuration for Apache Qpid 0.20 and earlier, when the federation_tag attribute is enabled, accepts AMQP connections without checking the source user ID, which allows remote attackers to bypass authentication and have other unspecified impact via an AMQP request.
14-Jun-2006	CVE-2012-3467	Apache QPID 0.14, 0.16, and earlier uses a NullAuthenticator mechanism to authenticate catch-up shadow connections to AMQP brokers, which allows remote attackers to bypass authentication.

The way the architecture of IoT has developed in recent years means that there are many protocol options available. The challenge, from the security aspect, is to ensure that the correct protocol is utilised in an appropriate environment. Often this will necessitate a solution which spans many protocols, and not just one (axway, 2015).

As with other IoT protocol software, the security of AMQP often suffers from the code being poorly written by many developers (Braue, 2015). Due to the open-source nature of AMQP, which allows for industry-specific extensions to be added, this exacerbates the variety of vulnerabilities that may be introduced to a particular messaging system based on AMQP.

Threats to AMQP Protocol

As the Internet of Things expands to encompass billions of devices around the world, the cybersecurity CIA triad of Confidentiality, Integrity, and Availability becomes as significant as ever. With an exponential growth in the number of IoT devices, so too is there a corresponding exponential growth in the number of lines of communication and data transfer, be they via wired or wireless connections. Indeed, in a situation where every device is capable of communicating with every other device, the number of communications channels equals n(n-1)/2, where n is the number of devices involved.

Every IoT communication channel is as vulnerable to potential man-in-the-middle cyber-attack as in a simple email communication between two end-users. The four types of such active attacks are:

1. Replay – An attack entity replays data between communication sessions to impersonate a user to obtain information.
2. Masquerade - An attack entity gains access to a system or performs a malicious act by posing as an authorised entity.
3. Modification - An attack entity performs additions or deletions to the network communication content.
4. Denial of Service – An attack that inhibits legitimate users from accessing computer services.

Despite AMQP using TLS/SSL-based encryption on an underlying TCP-based transmission protocol, resolute threat entities will still be able to intercept and decipher IoT communications, given sufficient time. Not only are we seeing IoT devices being introduced on the market with insufficient security measures (Arias et al., 2015), but we are also seeing IoT networks being compromised by the introduction of carefully-crafted botnets. An example of such an attack occurred at an unnamed University in the United States early this year (2017).

Cybercriminals were able to crack default or poorly-configured passwords in one IoT device via a brute force attack taking advantage of the device's inadequate security measures. Once this device was under their control, specially designed malware was able to be installed (Palmer, 2017).

The malware then spread from IoT device to IoT device by a botnet which again brute-forced weak or defaults passwords. As the botnet spread, it locked administrators out and repeatedly changed the password on infected devices with each malware update (Moss, 2017).

Within a short time frame, all 5,000+ devices were infected, and each device was making hundreds of DNS requests for seafood restaurants (Mezzofiore, 2017). The consequence of this DDoS attack was a severe slowing of the University's Internet access resulting in a loss of availability of resources required by students and staff (Palmer, 2017).

What makes this incident particularly interesting is that it is one of the few cases to date which has seen a botnet DDoS attack spread and then directed against the same network on which the infected devices are hosted. If such an attack is to involve the compromise of server-to-server communications hosting AMQP, then the potential would be for multiple IoT networks to be seeded with such internally-spreading infections; causing widespread compromise of AMQP-enabled IoT devices.

Figure 4 is a theoretical graphical interpretation of how the botnet attack on the University above may have been initiated and spread.

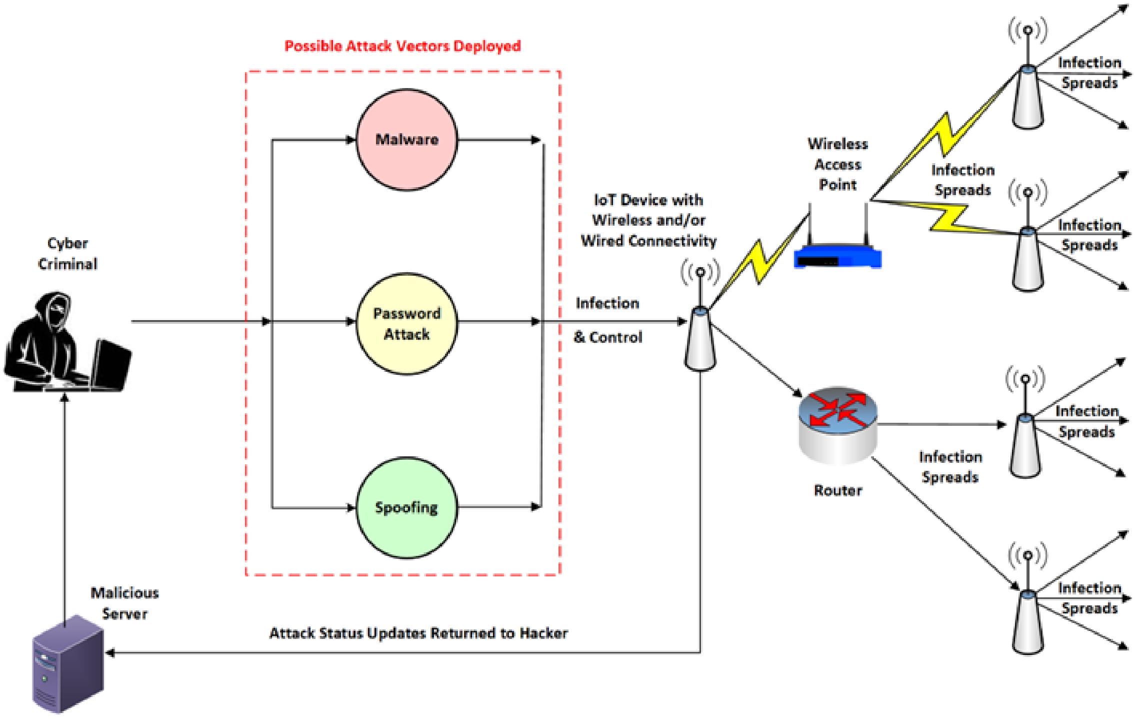

Figure 4. Hypothetical Interpretation of Attack Vector Methodology

Other software developers have created more direct AMQP-specific attack vectors. Enteletaor is a Message Queue & Broker Injection tool that can be used to implement attacks to AMQP-compatible messaging providers, such as RabbitMQ. It is also effective on non-AMQP-compatible message systems such as Redis, and ZeroMQ (GitHub, 2016). Features include:

1. Listing remote tasks.
2. Read remote task content.
3. Disconnect remote clients from Redis server (even the admin!)
4. Inject tasks into remote processes.
5. Make a scan to discover open brokers.
6. Try to discover user/passwords in authentication-protected brokers

CONCLUSIONS

From its origins in the finance sector, AMQP has developed far beyond its original scope to become a widely accepted messaging protocol throughout the field of IT, including the Internet of Things. Through the use of binding/routing keys, AMQP provides a powerful and flexible mechanism for sending messages from publishers to subscribers via exchanges and queues. AMQP's underlying TCP/IP-compatible framework adds to the diversity of its range of applications but also makes it vulnerable to an assortment of known weaknesses and methods of exploitation. Its usage as a server-to-server messaging protocol leads to the potential that its inherent TCP/IP-based vulnerabilities could be leveraged to propagate malware infections between networks. This research has reported various vulnerabilities that exist in AMQP protocol and how various threats can be used to exploit these weaknesses to make AMQP a susceptible protocol in IoTs list which otherwise has a very strong architecture. Therefore, the need for having an IoT device using AMQP protocol in a network must be carefully evaluated before AMQP becomes part of the system.

REFERENCES

AMQP. (2017a). AMQP. Retrieved from https://www.amqp.org/

AMQP. (2017b). Products and success stories. Retrieved from https://www.amqp.org/about/examples

Arias, O., Wurm, J., Hoang, K., & Jin, Y. (2015). Privacy and security in internet of things and wearable devices. *IEEE Transactions on Multi-Scale Computing Systems*, *1*(2), 99-109. doi:https://10.1109/TMSCS.2015.2498605

Ashton, K. (2009). That 'internet of things' thing. Retrieved from http://www.rfidjournal.com/articles/view?4986

axway. (2015). In the Internet of Things, the thing talks back. Retrieved from https://www.axway.com/sites/default/files/resources/tools_and_tips/axway_checklist_internet_of_things_security_checklist_10_critical_issues_you_need_to_address.pdf

Bhalla, A. (2012). Making the transition to IPv6. Retrieved from http://www.wipro.com/blogs/making-the-transition-to-ipv6/

Bradicich, T. (2015). The 7 Principles of the Internet of Things (IoT). Retrieved from http://blog.iiconsortium.org/2015/07/the-7-principles-of-the-internet-of-things-iot.html

Braue, D. (2015). Small, unsophisticated developers perpetuating IoT security lapses: IBM. Retrieved from https://www.cso.com.au/article/560521/small-unsophisticated-developers-perpetuating-iot-security-lapses-ibm/

CoAP. (2017). CoAP. Retrieved from http://coap.technology/

Common Vulnerabilities and Exposures. (2017). Search results. Retrieved from https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=amqp+iot

DDS. (2017). DDS. Retrieved from http://portals.omg.org/dds/

Evans, D. (2011). The internet of things [Infographic]. Retrieved from http://blogs.cisco.com/diversity/the-internet-of-things-infographic

Foote, K. D. (2016). A brief history of the internet of things. Retrieved from http://www.dataversity.net/brief-history-internet-things/

GitHub. (2016). Enteletaor. Retrieved from https://github.com/cr0hn/enteletaor

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst., 29*(7), 1645-1660. doi:http://10.1016/j.future.2013.01.010

Indrasiri, K. (2016). *Beginning WSO2 ESB*. Berkeley, CA: Apress.

Jacobs, J. (2017). What the future might hold for the internet of things. Retrieved from https://www.globalxfunds.com/what-the-future-might-hold-for-the-internet-of-things/

Meisinger, M. (2010). Ocean observatories initiative messaging service. Retrieved from https://confluence.oceanobservatories.org/display/CIDev/Messaging+Service

Mezzofiore, G. (2017). A university was attacked by its lightbulbs, vending machines and lamp posts. Retrieved from http://mashable.com/2017/02/13/internet-of-things-university-network-/#KXJv7vGDqOqj

Moss, S. (2017). University suffers DDoS attack from IoT vending machines. Retrieved from http://www.datacenterdynamics.com/content-tracks/security-risk/university-suffers-ddos-attack-from-iot-vending-machines/97808.fullarticle

MQTT. (2017). MQTT. Retrieved from http://mqtt.org/

Osisanwo, F., Kuyoro, S., & Awodele, O. (2015). *Internet refrigerator – A typical internet of things (IoT)*. Paper presented at the 3rd International Conference on Advances in Engineering Sciences & Applied Mathematics (ICAESAM'2015), London (UK). http://iieng.org/images/proceedings_pdf/2602E0315051.pdf

Palmer, D. (2017). How IoT hackers turned a university's network against itself. Retrieved from http://www.zdnet.com/article/how-iot-hackers-turned-a-universitys-network-against-itself/

Richardson, A. (2008). *AMQP business messaging for predictable, scalable, available SOA*. Paper presented at the Microsoft Architects Insight Conference 2008. http://download.microsoft.com/documents/uk/msdn/events/sol/sol03.pdf

Ross, T. (2012). Integrating the internet of things with AMQP. Retrieved from https://www.scribd.com/document/234021715/AMQP-IoT-pdf

Service Architecture. (2017). Representational State Transfer (REST). Retrieved from http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html

SMQTT. (2017). SMQTT. Retrieved from http://smqtt.com/

TechBeacon. (2017). The 4 stages of an IoT architecture. Retrieved from https://techbeacon.com/4-stages-iot-architecture

Varma, P. K. (2010). Aadhaar: Scalability & data management challenges. Retrieved from https://www.cse.iitb.ac.in/~comad/2010/pdf/Industry%20Sessions/UID_Pramod_Varma.pdf

Vaughan-Nichols, S. J. (2016). Death of the internet: GIF at 11. Retrieved from http://www.zdnet.com/article/death-of-the-internet-gif-at-11/

WebSocket. (2017). About HTML5 WebSocket. Retrieved from https://www.websocket.org/aboutwebsocket.html

XMPP. (2017). XMPP. Retrieved from https://xmpp.org/