Security in the Internet of Things: A Survey on Application Layer Protocols

Lavinia Năstase
Faculty of Automatic Control and Computer Science,
University Politehnica of Bucharest, Romania
lavinia.nastase10@yahoo.com

Abstract – The rapid development of technology nowadays led people to a new and revolutionary concept, named the Internet of Things. This model imposes that all "objects", such as personal objects (smartphones, notebooks, smart watches, tablets etc), electronic equipment embed with sensors and other environmental elements are always connected to a common network. Therefore, one can access any resource at any time, by using a device recognized in the network. While the IoT may be economically and socially beneficial, the implementation of such a system poses many difficulties, risks and security issues that must be taken into consideration. Nowadays, the architecture of the Internet must be updated and rethought in order to interconnect trillions of devices and to ensure interoperability between them. Nevertheless, the most important problem is the security requirements of the IoT, which is probably one of the main reasons of the relatively slow development of this field. This paper presents the most important application layer protocols that are currently used in the IoT context: CoAP, MQTT, XMPP. We discuss them both separately and by comparison, with focus on the security provided by these protocols. Finally, we provide some future research opportunities and conclusions.

Keywords: Internet of Things (IoT), IoT Security, Application layer protocols, CoAP, MQTT, XMPP

I. INTRODUCTION

The modern world is leading towards this whole new concept of the Internet of Things, but the exact definition of this concept is still a subject of debate in the research world. The concept was released for the first time in 1999, by the Auto-ID laboratory of the MIT (Massachusetts Institute of Technology) and it was defined as "data and devices continually available through the Internet" [1]. The IoT comprises sensors, smart devices, networks, cloud computing, all connected through common standards. Each layer poses vulnerabilities and security threats. The second section of this paper describe some proposed architectures of the IoT, which strive to provide a flexible layered model that should support a large number of connected heterogeneous objects. Section III describes three of the most common protocols in the application layer and aims to provide a critical approach towards the security of each one of them. We also make a comparison of the discussed protocols. Section IV focuses on vulnerabilities and possible issues in the application layer. The last two sections provide some ideas for future research in this field and conclusions on the paper.

II. IOT GENERAL ARCHITECTURE

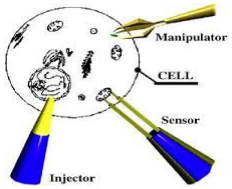
Figure 1 illustrates the main underlying technologies, according to [1], as following:

- *RFID* (Radiofrequency identification), that uses a tag to identify and track the objects to which they are attached,
- *Sensors*, that have the purpose of collecting data and digitizing it for further processing, and actuators, for transforming digital commands into physical actions as light, heat or movement; for example, querying location, gathering environmental parameters, such as temperature or humidity, measuring weight, vibration, speed etc.
- *Smart technologies*, to enhance the information processing capabilities,
- *Nano-technologies*, aiming to connect small "things".

a) RFID

b) Sensors, Actuators

c) Smart Technologies



d) Nanotechnologies

Figure 1. The main underlying IoT Technologies

2379-0482/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCS.2017.101

Given the huge amount of independent researches in the last decade, materialized in a large volume of new products on the market, the interoperability and scalability should be kept in mind, especially in a less standardized architecture compared to OSI. The stages of an IoT system can be summarized in five steps that are useful in defining a scalable architecture, as in figure 2.

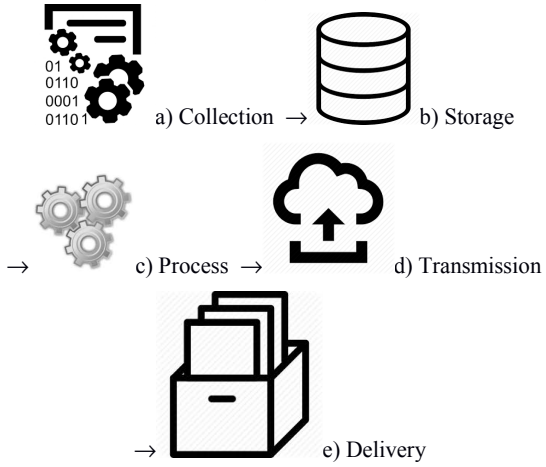


Figure 2. The stages of an IoT System

The first step in any IoT application is the collection of data. Let us take the example of an eHealth application that measures a patient's blood pressure. Firstly, the data collectors located in the device that takes the blood pressure are used; they can be static, such as a RFID tag, or dynamic, as a sensor. Secondly, we are interested in storing this data locally, in the device that contains the collector; usually, these devices have low memory and processing power, therefore the operations done at this level have to be optimized. In case of stateless devices (usually RFID), the data is be stored into the cloud. Thirdly, the IoT offers various services and processing of the gathered data, as well as responding to queries regarding the information that resulted in this process. The fourth step consists in transmitting the data; there are various ways of transmission: from data collectors to data centers/cloud, from data centers to processing units or to end users. In our case, we would be interested to send the blood pressure information to the patient's doctor, along with some reports, such as patient's history or prognosis. The last step, Delivery, sometimes overlaps the previous, especially in less complex applications. However, when delivery is necessary as the last step, it usually involves provisioning and administration tasks in a complex and strictly defined business environment.

Keeping in mind these essential steps, the four proposed architectures described in [2] should be straightforward. We state them as an introduction to the next section.

a. Three layered architecture: application layer, network layer, perception layer.
b. Middleware based architecture: application layer, middleware layer, coordination layer, backbone network layer, existed alone application layer, access layer, edge technology.
c. Service oriented Architecture (SOA based): applications, service composition, service management, object abstraction, objects.
d. Five layered architecture: business layer, application layer, service management, object abstraction, objects.

The details of all these proposed solutions are discussed in [1]. A generic model agreed in many articles is a five-layer architecture, from end-user to real world: business, application, middleware, network and perception, which is a combination of the models enumerated above. The difficulty on agreeing upon a common design emphasizes the lack of standardization and common approval in the IoT, which leads to the necessity of further research in this field. One of the main reasons for relatively poor standardization is the huge number of competitors on the market of "smart devices" aiming to earn the supremacy of their own devices, in a wide range of applications and without a true global authority to enforce a standard.

The focus in this paper is on the application layer, the true service provider for the end user. Since this layer is the "gate" to outside of the network, there are numerous threats that can arise. This level assures the user authentication and access to personal and sensitive data; therefore, the protocol must provide mechanisms for preventing intruders and malicious users to gain access to the system. The traditional attacks can happen: DDoS attacks, spoofing, data modification, eavesdropping etc.

III. Security in the Application Layer

Any object, tangible or not, cannot be 100% secure because in this perfectly secure state it cannot be still be useful in terms of its intrinsic value [3]. However, if it can maintain its maximum intrinsic value under different conditions, it can be considered secure, therefore, ensuring IoT security requires maintaining the highest intrinsic value, including all the IoT objects in the application layer context. In order to build a valuable security context, threats must be known and understood, and the following questions must get precise answers [4]:

1. What are the assets?
2. Who are the principal entities?
3. What are the threats?
4. Who are the threat actors?
5. What capability and resource levels do threat actors have?
6. Which threats can affect what assets?

7. Is the current design protected against threats?
8. What security mechanisms could be used against threats?

According to [5], security and privacy issues are a growing concern for users and suppliers in their shift towards the IoT. The IoT needs to be built in such a way as to ensure easy and safe usage control. Consumers need confidence to fully embrace the IoT in order to enjoy its benefits and avoid security and privacy risks.

The overall security aspects for E2E communications are well stated by [6], which can be taken as a standard for assessing cyber-security. Based on this, specific evaluations of protocols that provides such communication can be done, and [7] is such an example. The recommendation proposes a general and tridimensional model for the security: (1) three security layers (applications, services and infrastructure), (2) three security planes (end user, control and management) which are identified based on the activities performed over the network, and also (3) eight security dimensions to address general system vulnerabilities (Access Control, Authentication, Non-Repudiation, Data Confidentiality, Communication Security, Data Integrity, Availability and Privacy). By restricting the analysis to the application layer, it passes into a bidimensional one, involving three security planes and eight security dimensions. This security assessing model is rather general and can be extended to other layers too, including application layer.

In this section, three common IoT application layer protocols are taken into discussion: MQTT, XMPP and CoAP. As a common point of discussion, any of them is derived from traditional Internet protocols and further adapted to the IoT specifications to make them suitable for application involving constrained devices and networks. However, each of them performs better only in certain and specific conditions. Security characteristics, comments and opportunities are provided for each as well.

A. MQTT

MQTT is a publish-subscribe messaging protocol, based on brokers, created by IBM. MQTT is a lightweight protocol because all messages have a small code footprint. It uses TCP/IP protocol and it is suitable in constrained environments, for example when a device has low memory resources or a limited processor. Also, message payload is limited to a maximum of 256 MB of information, which makes this lightweight protocol suitable in expensive and unreliable networks.

According to the MQTT protocol specification[1], three QoS (quality of service) modes are provided for the message delivery:

- *"Fire and forget"* mode, also known as "at most once": in this case, message loss can happen; therefore, it can be used in environments where an individual measurement is not of great importance, since a next one would be published afterwards.
- *"Acknowledged delivery"* or *"at least once"*: send duplicates can occur, nonetheless all messages arrive.
- *"Assured delivery"* or "*exactly once"*: in this mode, all messages arrive to their destination exactly once. This QoS is useful in applications where missing or duplicate messages lead to unwanted results, as it can happen in a payment service, for example.

Although the last two levels are more reliable, they also impose a certain overhead and bandwidth requirements which are not always feasible, but clients can choose the desired QoS.

Since the MQTT protocol is based on TCP/IP, a connection is initiated by a client to a broker on a certain port (either standard or defined by the broker). For example, TCP port 1883 is reserved for non-encrypted while 8883 is for encrypted communication using SSL/TLS. A session usually consists of four phases: connection, authentication, communication, termination. A client can subscribe to topics of interest, defined by a publisher and can also unsubscribe from them. The connection can be closed either by the client or by the broker.

MQTT is currently used in the Facebook Messenger application, since is allows delivering messages in a timespan of milliseconds, regardless of the Internet connection, but also requires low power, which prevents smartphone batteries from draining.

Security

The interesting part in the MQTT specification is the fact that it has no imposed security mechanisms, because it is designed to operate in secure networks, developed for specific needs. Therefore, it is not a good idea to create a globally MQTT network, because as the size of the topic tree grows, complexity increases.

The IoT environment nowadays requires however a standard for authentication and therefore MQTT relies on SSL/TLS encryption; otherwise, the username and password would be sent in clear text. During the handshake, the client validates the server certificate, which means that it verifies its identity in order to authenticate it.

[1] MQTT V3.1 Protocol Specification, http://public.dhe.ibm.com/software/dw/webservices/ws-mqtt/mqtt-v3r1.html

Optionally, client certificates can also be used, in such way that the broker can authenticate the client that requests the connection[2]. This is not a simple process and there is no way for the message receiver to know who sent the original message unless that information is contained in the actual message (not mandatory). In an extended network, a poor application design based on MQTT could be an invitation to hackers to inject harmful messages into the network in a very easy way. Therefore, security features must be implemented on top of MQTT. However, considering the specificity of a relatively isolated and secured network, this would not be the case.

The drawback of this mechanism is that SSL/TLS is an expensive protocol to use in constrained devices. Nowadays, certain brokers accept anonymous clients and therefore the username and password are no longer required, but it is not desirable in most of the cases.

B. XMPP

XMPP (Extensible Messaging and Presence Protocol) is composed of a set of open technologies mostly used for chat, instant messaging, video and voice calls and is standardized by the IETF[3]. In this protocol, data is exchanged using small pieces of XML structured data, called "XML stanzas", between two or more network entities on a client-server basis. The architecture is, therefore, based on client-server concept; the client requests the connection to a server to be able to exchange messages. Since XMPP runs on top of TCP, a TCP connection is needed before opening an XML stream. Channel encryption is usually assured with TLS (but not mandatory) and the authentication is made via a SASL mechanism. Server to server connection is also possible, after negotiating between themselves and allowing inter-domain communication.

Abstract layering of XMPP is based on (i) Transport Control Protocol (TCP), (ii) Transport Layer Security (TLS) (iii) Simple Authentication and Security Layer (SASL) and finally (iv) Extensible Messaging and Presence Protocol (XMPP).

A XMPP Client is the entity that establishes the XML stream with a server, after authenticating itself through SASL negotiation. A XMPP server can manage the open streams with the connected clients, receiving and delivering XML stanzas as required. It also needs to verify client authentication prior to granting access to the XMPP network. Other responsibilities would be the client data storage (e.g. a list of contacts in a chat-like application) and, for some services such as conferencing, hosting add-on services that use XMPP as communication basis[4].

Finally, since XMPP based solutions are usually deployed in decentralized client-server architectures, there are two possible paths: client-to-server stream and server-to-server stream. If both entities are clients, it is not possible for them to open a communication channel directly between them; instead they need an intermediate entity (a server) with a certain level of trust.

Security

In terms of security, the IETF proposes for the XMPP native support to authentication via SASL (*Simple Authentication and Security Layer*) and transport security with TLS, according to the RFC. The advantage of SASL is that it provides a set of authentication methods from which the client can choose one that best fits the specific needs; however, the drawback is that a weak mechanism still can be chosen. SASL uses Base64 coding which hides easily recognized information; however, it doesn't provide any computational confidentiality. IETF recommends secure mechanisms for peer authentication, such as SCRAM-SHA-1 or SCRAM-SHA-1-PLUS, which protect against man-in-the-middle-attacks, spoofing and unauthorized access.

The stream is secured from tampering and eavesdropping by encrypting using the TLS protocol, specifically a STARTTLS extension that is modeled after similar extensions for the widely used IMAP, POP3 and ACAP protocols. To protect the connection credentials, TLS negotiation should complete before starting SASL.

A strong security strategy could employ security technologies that provide both mutual authentication and integrity checking (e.g. a combination of TLS encryption and SASL authentication). Channel encryption of an XML stream using TLS and in some cases authentication are commonly based on a PKIX certificate presented by the receiving entity or, in the case of mutual authentication, both the receiving and the initiating entity. For preserving the integrity of the stanzas, the signature algorithm should be at least SHA-256.

Unprotected XMPP systems are vulnerable to eavesdropping, sniffing passwords, breaking passwords through dictionary attacks, discovering usernames through directory harvesting attacks, replaying, inserting, deleting, or modifying stanzas, spoofing users, gaining unauthorized entry to a server or account, subverting communication streams through man-in-the-middle attacks and more.

One of the most evident vulnerabilities of XMPP, in spite of its relative stability, emerges from the fact that a stanza can travel along multiple streams, some of them might not be TLS-protected. Only a robust end-to-end

[2] TechTarget, IoT Agenda – "MQTT (MQ Telemetry Transport)", http://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport

[3] XMPP RFC, https://tools.ietf.org/html/rfc6120

[4] XMPP Protocol, https://xmpp.org/

encryption could ensure enough confidentiality and integrity of a stanza that travels all the "hops" along a communication path. However, the XMPP community has so far failed to produce an end-to-end encryption solution that might be suitable for widespread implementation and deployment in IoT.

C. CoAP

Since IoT enables a wide range of application scenarios with potentially critical actuating and sensing tasks in given constrained environments, e.g., in the e-health domain, a simple, though complete protocol is needed. CoAP (Constrained Application Protocol) is a specialized web transfer protocol for use with IoT constrained nodes (i.e. having limited memory and processing power) and constrained networks (i.e. low-power and lossy), being currenly standardized at the IETF[5].

CoAP obeys a simple request/response interaction model between application endpoints, most similar to the client/server model of HTTP, also providing built-in support for services and resources. CoAP makes use of GET, PUT, POST, and DELETE methods in a similar manner to HTTP.

However, unlike HTTP, CoAP deals with these interchanges asynchronously over a datagram-oriented transport, such as UDP, most suitable in constrained environments. Also, having UDP in transport layer, unlike HTTP, CoAP supports the use of multicast IP destination addressing, thus enabling multicast requests.

Moreover, unlike HTTP, requests and responses are not sent over a previously established connection, but are exchanged asynchronously over CoAP messages. All CoAP traffic can be supported through only four types of messages: (i) Confirmable (CON), (ii) Non-confirmable (NON), (iii) Acknowledgement (ACK) and (iv) Reset (RST).

Abstract layering of CoAP is based on (i) UDP, (ii) Requests/Responses and Messages and (iii) Application.

Message reliability is provided by marking as CON, eventually retransmitting it on a default timeout basis until a corresponding ACK is received from the corresponding endpoint. However, when a message does not require reliable transmission (for example, each single measurement out of a stream of sensor data) it can be sent as NON. As CoAP is by default bound to unreliable transports such as UDP, messages may arrive out of order, appear duplicated, or go missing without notice. For this reason, CoAP implements a lightweight reliability mechanism, without trying to re-create the full feature set of a transport like TCP.

[5] CoAP RFC, https://tools.ietf.org/html/rfc7252

Security

Various solutions can be employed for binding a security layer to CoAP. The specificity of using UDP instead of TCP for transport makes inapplicable the known security protocols in the form they are used in association with TCP. Conversely, for some of such protocols substitutes have been developed. For example, the replacement of TLS (specific to TCP) into UDP environments is DTLS (Datagram Transport Layer Security).

Due to the physically limited access, some applications do not even need to employ any security bound to transport layer (UDP). Instead, there are techniques to provide lower-layer security, namely IPSec at the lower (network) layer, when connecting to the outside world. In this mode, the data packets are simply sent over normal UDP over IP. The system security is provided only by routing techniques, by keeping attackers from being able to send/receive packets to/from the specific network with the CoAP nodes.

By using the three DTLS modes described below, for securing UDP transport for CoAP, the new architecture is called CoAPs (secured), just the same as HTTP secured with SSL/TLS is replaced by HTTPS [8]. In this way, the security association can be used to authenticate (within the limits of the security model) and, based on this authentication, authorize the communication partner, since CoAP itself does not provide any protocol primitives for authentication or authorization.

PreSharedKey mode is based on a list of pre-shared keys, each one including a list of nodes it can be used to communicate with. There may be one key for each node, however if more than two entities share the same pre-shared key, this only enables the entities to authenticate as a member of a group and not as a specific peer. When trying to establish a connection to a new node, the system selects an appropriate key based on which nodes it is trying to reach and then forms a DTLS session using PSK (Pre-Shared Key) mode of DTLS.

In *RawPublicKey* mode, the device has an asymmetric key pair without a certificate (a raw public key) that is validated using an out-of-band mechanism (most common, the asymmetric key pair is generated by the manufacturer and installed on the device), an identity calculated from the public key and a list of identities of the nodes it can communicate with. A device may be configured with multiple raw public keys. Implementations in RawPublicKey mode must support cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8. The key used must be ECDSA capable.

In *Certificate* mode, the device has an asymmetric key pair with an X.509 certificate that binds it to its subject and is signed by some common trust root. The device also has

a list of root trust anchors that can be used for validating a certificate. Implementations must support the cipher suite TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8. If the system has a shared key in addition to the certificate, then a cipher suite that includes the shared key such as TLS_ECDHE_PSK_WITH_AES_128_CBC_SHA should be used.

Using full DTLS for securing CoAP may introduce a significant overhead in constrained environments with either limited local memory on nodes, or limited bandwidth on communication, or both, affecting the overall solution efficiency. Therefore, depending on application, DTLS modes may be configured by disabling all the unnecessary ones, thus making the protocol much lighter.

CoAP supports a limited set of HTTP functionality and thus cross-protocol proxying to HTTP is straightforward. For example, when designing a web interface for use over either protocol or when realizing a CoAP-HTTP proxy. Likewise, CoAP could be equally proxyed to other protocols such as XMPP or MQTT, which is a very good thing in terms of application interoperability. Proxying is accomplished by means of an intermediary. However, care should be taken when designing and authorizing such solutions because of vulnerabilities that can be introduced. For example, for any pair of protocols, one of the protocols can very well have been designed in a way that enables an attacker to cause the generation of replies that look like messages of the other protocol.

It is often much harder to ensure or prove the absence of viable attacks than to generate examples that may not yet completely enable an attack but might be further developed by more creative minds.

IV. VULNERABILITIES AND ISSUES

At the application layer, data, applications, and visualization servers are usually operated in large and complex clusters or in the cloud computing infrastructures, therefore can be affected by data tampering, authentication to the servers, authorization of services, provisioning of data.

Based on the specific environment of any IoT based solution, any challenges or possible vulnerabilities at any layer including application layer can be more or less meaningful. For this reason, they must be carefully assessed in the early stages of the technical design.

According to [9], vulnerabilities are weaknesses in a system or its design that allow an intruder to execute commands, access unauthorized data, and/or conduct denial-of-service attacks. More, [10] identifies a threat as an action that takes advantage of security weaknesses in a system and has a negative impact on it, being originated from two primary distinct sources: humans and nature. It is evident that natural threats cannot be forecasted, the good news is that disaster recovery plans are the best and universally applicable choice to ensure business continuity. The human attacker is certainly smart and is likely to destroy privacy in the application layer by a known vulnerability (e.g., buffer overflow, cross site scripting, SQL injection and others), error configuration (simple password for example), or improperly obtained higher permission access. Four security threats at the application layer are identified in this context [11]:

1. *Privacy leak* – given that the application of IoT is executed on common operating systems and hosting services, the attacker can easily steal user data (e.g., user password, historical data, and social relations) by known vulnerabilities
2. *DoS attack* – the attacker can destroy the availability of the application itself
3. *Malicious code* – the attacker can upload malicious codes through the known vulnerabilities, leading to fetcher software infections
4. *Social engineering* – certain relationship exists among IoT users which attackers can easily analyze or obtain additional information that can be used for attacks by social engineering.

[8] concludes that technical vulnerabilities usually happen due to human weaknesses. Results of not understanding the requirements comprise starting the project without a plan, poor communication between developers and users, a lack of resources, skills, and knowledge, and failing to manage and control the system. More, attacks are actions taken in order to harm a system or disrupt normal operations by exploiting vulnerabilities by using various techniques and tools.

Testing applications for security flaws goes well beyond simply preventing attacks. Application vulnerabilities can lead to lost or stolen data, which could potentially result in even more serious consequences, such as stakeholder lawsuits, extensive remediation costs and damage to brand reputation.

V. FINDING NEW RESEARCH DIRECTIONS

To reduce both potential threats and their consequences, more research is needed to fill the gaps in knowledge regarding threats and cybercrime and provide the necessary steps to mitigate probable attacks.

Some IoT application protocols do not even have security features, so they can not actually be used in a Internet based IoT solution, even if, in terms of functionality, they perform better than others prefferred with security in mind. For example, in can be mentioned that more work can be done to empower XMPP or MQTT in E2E secured solutions.

Regarding CoAP, even if DTLS is being considered to support security, it presents some limitations motivating other approaches to security at the application-layer that can be addressed by further research. In [12] the author discusses various issues that may impede the usage of DTLS in constrained sensing devices, for example, the inadequateness of the timers for message retransmission as defined in the protocol, which may require large buffers on the receiver to hold data for retransmission purposes, and the size of the code required to support DTLS in constrained sensing platforms. In response, [13] and [14] propose solutions to some issues identified during massive cyberattacks in the last three years, by making TLS/DTLS more rapid and resilient, however, some issues still remain and still need to be addressed by researchers.

Further research can also address the support of public keys and certificates in the context of CoAP security. Online validation of certificates may be achieved by investigating the applicability of existent Internet approaches such as the Online Certificate Status Protocol (OCSP) [15] or OCSP stapling through the TLS Certificate Status Request extension defined in RFC 6066 [16], considering that such mechanisms could be adapted or simplified.

Other important issue to consider for further research is the computational impact of ECC cryptography on existing sensing devices. In this context, optimizations may be designed at the hardware of sensing platforms to support ECC computations.

In this context, [11] concludes that the overall research on security issues related to the IoT domain remains inadequate. The inherent openness, heterogeneity, and terminal vulnerability of the IoT pose a huge risk, considering two main problems: coupling (different technologies and trust mechanisms are related to each other) and completness (security architecture design should consider future application trends).

Considering the benefits of scalability and interoperability at the application layer, further research must be taken in the field of securing cross-protocol proxies for the application layer IoT solutions.

VI. Conclusions

The Internet of Things is an universal medium for "things" to communicate with each other via Internet, access data on the Internet, store and retrieve data, and interact with users. They will continue to change our lives as the involved technologies are continuously growing. This paper describes briefly three of the IoT application layer protocols from the security point of view. It can be seen that a trade-off is made between lightweight and security, thus none of them is best for any type of solution in terms of both security and functionality.

Security is the biggest challenge facing the IoT today. In developing new solutions for the IoT, organizations must consider the larger context and implications of security and privacy from the very beginning. On the other hand, to prevent cyber attacks, organizations must ensure that they educate their consumers about the correct security procedures to be followed while using an IoT system.

It is evident that successful attackers are smart since their success is based on knowledge. But it is also true that for successful IoT projects, the designers must be smarter, in other words be at least one step in front of any smart attacker. It is a continuous competition between the two parties and will forever be like that, since none is truly wise, meaning know and understand everything. For that, like in any domains, the IoT research has to continue forever, sooner or later any reasonable technological barrier that can be imagined nowdays has to be broken.

In conclusion, this survey may provide a contribution to documenting the status of the dynamic area of research in securing the IoT application layer protocols.

References

[1] Fei Hu – “Security and Privacy in Internet of Things (IoT). Models, Algorithms and Implementarions”, CRC Press, 2016

[2] Stefan Mijovic, Erion Shehu, Chiara Buratti – “Comparing Application Layer Protocols for the Internet of Things via Experimentation”, 2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)

[3] J. M. Kizza – "Guide to Computer Network Security", Springer, 2013

[4] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer et al. – "Internet of Things Strategic Research Roadmap", Internet of Things-Global Technological and Societal Trends, 2011

[5] P. N. Mahalle, N. R. Prasad, R. Prasad – "Object classification based context management for identity management in internet of things", International Journal of Computer Applications, vol. 63, no. 12, 2013

[6] Thamer A. Alghamdi – "Security Analysis of the Constrained Application Protocol in the Internet of Things", IEEE 978-1-4799-2975-7/13 – 2013

[7] ITU-T Telecommunication Standardization Sector of ITU - "X.805 (10/2003) Security architecture for systems providing end-to-end communications", Series X: Data Networks and Open System Communication Security

[8] Shahid Raza, René Hummen – "Lithe: Lightweight Secure CoAP for the Internet of Things", IEEE Sensors Journal, October 2013

[9] D. L. Pipkin – “Information security”, Prentice Hall PTR, 2000

[10] H. G. Brauch – "Concepts of security threats, challenges, vulnerabilities and risks", Coping with Global Environmental Change, Disasters and Security, Springer, 2011

[11] Weizhe Zhang, Baosheng Qu - "Security Architecture of the Internet of Things Oriented to Perceptual Layer", International Journal on Computer, Consumer and Control (IJ3C), Vol. 2, No.2(2013)

[12] K. Hartke – "Practical Issues With Datagram Transport Layer Security in Constrained Environments", DICE Working Group, Internet-Draft, Universität Bremen TZI, April 8, 2014.

[13] Markulf Kohlweiss, Ueli Maurer, Cristina Onete, Björn Tackmann, Daniele Venturi - "*(De-)Constructing TLS 1.3*", Proceedings of Progress in Cryptology – INDOCRYPT 2015: 16th International Conference on Cryptology in India, Bangalore, India, December 6–9 2015

[14] Hugo Krawczyk, Hoeteck Wee – "The OPTLS Protocol and TLS 1.3 (extended abstract)", IEEE European Symposium on Security and Privacy (EuroS&P), March 21–24 2016

[15] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams – "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol-OCSP", RFC 2560, 1999.

[16] D. Eastlake – "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, 2011.

[17] Vasileios Karagiannis, Periklis Chatzimisios, Francisco Vazquez-Gallego, Jesus Alonso-Zarate – "*A Survey on Application Layer Protocols for the Internet of Things*", Transaction on IoT and Cloud Computing 2015

[18] Jorge Granjal, Edmundo Monteiro, Jorge dá Silva – "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues", IEEE Communication Surveys & Tutorials, Vol. 17, No. 3, Third Quarter 2015

[19] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari and Moussa Ayyash – "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", IEEE Communication Surveys & Tutorials, Vol. 17, No. 4, Fourth Quarter 2015

[20] Mohamed Abomhara, Geir M. Køien – "Cyber Security and the Internet of Things: Vulnerabilities, Threats, Intruders and Attacks" – River Publishers 2015, Journal of Cyber Security, Vol. 4

[21] D. Jiang, C. ShiWei – "A study of Information Security for M2M of IoT", Advanced Computer Theory and Engineering (ICACTE), 2010 - 3rd International Conference on, vol. 3. IEEE, 2010

[22] C. Hongsong, F. Zhongchuan, Dongyan – "Security and Trust Research in M2M System", Vehicular Electronics and Safety (ICVES), 2011 IEEE International Conference

[23] I. Cha, Y. Shah, A. U. Schmidt, A. Leicher, M. V. Meyerstein – "Trust in M2M Communication", Vehicular Technology Magazine, IEEE, vol. 4, no. 3, 2009

[24] S. Andreev, Y. Koucheryavy – "Internet of Things, Smart Spaces, and Next Generation Networking", Springer, LNCS, vol. 7469

[25] J. Lopez, R. Roman, C. Alcaraz – "Analysis of Security Threats, Requirements, Technologies and Standards in Wireless Sensor Networks", Foundations of Security Analysis and Design V. Springer, 2009

[26] R. Roman, J. Zhou, J. Lopez – "On the Features and Challenges of Security and Privacy in Distributed Internet of Things", Computer Networks, vol. 57, no. 10, 2013

[27] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac – "Internet of Things: Vision, Applications and Research Challenges", Ad Hoc Networks, vol. 10, no. 7, 2012

[28] M. Abomhara, G. Koien – "Security and privacy in the internet of things: Current status and open issues", PRISMS 2014, the 2nd International Conference on Privacy and Security in Mobile Systems (PRISMS 2014), Aalborg, Denmark, May 2014

[29] E. Bertino, L. D. Martino, F. Paci, A. C. Squicciarini – "Web services threats, vulnerabilities, and countermeasures", Security for Web Services and Service-Oriented Architectures, Springer, 2010

[30] D. G. Padmavathi, M. Shanmugapriya – "A survey of attacks, security mechanisms and challenges in wireless sensor networks", arXiv preprint arXiv:0909.0576, 2009

[31] E. Bertino, L. D. Martino, F. Paci, A. C. Squicciarini – "Web services threats, vulnerabilities, and countermeasures", Security for Web Services and Service-Oriented Architectures, Springer, 2010