Digital Identities for Internet of Things Devices

Tor-Morten Grønli
Mobile Technology Lab
Department of Technology
Kristiania University College
Oslo, Norway
tor-morten.gronli@kristiania.no

Gheorghita Ginea
Department of Technology,
Kristiania University College, Oslo, Norway
College of Engineering, Design and Physical Sciences
Brunel University
Uxbridge, United Kingdom
george.ghinea@brunel.ac.uk

***Abstract*—The IoT industry is still suffering from major security shortcomings, despite an increasing focus on the area. This is due to both a lack of security expertise among developers and a lack of good security solutions for IoT infrastructure. When development starts, it is the functionality that places the greatest emphasis and security is either forgotten or there simply is no time for it. The lack of security is also affected by the resource constraints on many IoT devices. Some IoT devices have the capacity to run an OS, but the devices most widely used today are simple microcontrollers, which often have small, simple, and very specific tasks. To resolve this problem, SSL certificates are a secure solution and a prototype implementation is showcased.**

Keywords—Web Services and Cloud-Based e-Services; Security, Privacy and Trust for e-Service

I. INTRODUCTION

Internet of Things (IoT) and security is an issue that is becoming more and more relevant as the number of IoT devices is increasing worldwide, and more and more are being exposed, hacked and abused as a result of poor security implementations. Thereby increasing the importance to investigate different IoT devices and identify what opportunities there are in securing these. Many IoT devices have few resources available, which limits the methods that can be used. This makes security on IoT devices challenging and there are currently no simple steps developers can take to secure their IoT devices.

Internet of Things is a broad concept with several definitions. In general, an IoT device is a physical thing connected to the Internet to either transmit data it has read through sensors or receive data that it can act on with actuators. In the work by McEwen and Cassimally [1] they define IoT to cover all devices that use the Internet to send and receive data, and are not a computer, phone, or tablet. In the "Internet of Things", "things" are physical devices with sensors and / or actuators. The Institute of Electrical and Electronics Engineers (IEEE) makes it very short by calling IoT a network of things that are connected to the internet and where each device has sensors [2]. The International Telecommunication Union defines IoT as a global infrastructure that offers advanced services by utilizing existing information and communication technologies to connect things [3]. More specifically, they define "things" as something that is capable of being integrated into a communication network. At the same time, one thing is not just something in the physical world, but also something that can be part of the virtual. IoT involves the collection, processing and transport of data.

Interestingly there is less emphasis on the fact that modern IoT solutions must face continuously stricter security and confidentiality requirements.

II. RELATED WORK

Many of the same threats that exist elsewhere in technical solutions and software architectures also apply to IoT [4]. Some IoT devices are more vulnerable than others, depending on which scenarios they are linked to. For some units, it will be important for data to be kept confidential, while for others i.e. a temperature measurement. Real-time device tracking can pose a security risk if data is not sent and kept confidential. Further to this, many of the same attacks we see against IoT are also common in technology. In many cases, IoT can be more vulnerable than other areas standard IT solutions because IoT devices are often located in places with easy physical access to the device. If an attacker has the ability to physically access an IoT device, there are several attacks they can make.

With physical access, it will be possible to install malicious software on the device, i.e. through keystroke injectors that pretend to be keyboards and very quickly type commands to the device they are connected to. This can be used, among other things, to install backdoors, which can allow the attacker to enable hidden functionality remotely on the device at a later date. Unsafe software can also be installed, which enable malicious functionality when certain criteria are met, such as when a specific time is reached, or a sensor reads a certain value. Other examples are software that allows you to perform Man-In-The-Middle attacks. These are attacks that allow you to monitor, modify, and destroy the data your IoT device handles while transporting over the network. The device can also be used to stage the spread of a virus or worm from the IoT device to access the servers it is talking to [5].

With physical access, it's not just installing malicious software on the device the attacker can do. It is also possible for the attacker to connect external equipment to the device to monitor traffic or perform MITM attacks. Arguably it is very important that IoT devices located in the public space are well physically secured so that a potential attacker cannot access them. Examples of such physical security include the device being locked in and the gates on the devices being locked or possibly removed [6]. That said, an IoT device that is well physically secured may still be vulnerable due to its location: Too many IoT devices in remote locations make it easy for an attacker to work undisturbed.

Even though a device is physically protected, there can still be ways to attack it. If the device communicates over Wi- Fi traffic can be intercepted and data compromised. It is also possible to block this Wi-Fi connection with a Wi-Fi Deauther or a computer with a wireless network adapter and Wi-Fi security testing software, e.g. Aircrack-ng. This can also be used for Session hijacking: trying to get access to the Wi-Fi network or get the IoT to connect to another Wi-Fi hotspot that claims to be the original network [5] - in turn, allowing MITM attacks against the IoT device.

A. Mapping of Attacks using OSI model

Based on many of these types of attacks and security holes, we have investigated and surveyed a broader spectre of them. To map attacks that one possibly want to protect against in IoT devices, a compiled overview related to the Open Systems Interconnection (OSI) model is shown in Table 1 with mapping to the corresponding layers of network communication (Table 1).

TABLE I. MAPPING OF ATTACKS USING OSI MODEL

Attack	Area	Goal	Level
SSH-Brute Force	Login	Achieve access to single device	7. Application
MITM (Packet squirrel)	Network	Capture and inject data from network transmission	1. Physical
WiFi Deauther	Network	Disconnect units from WiFi network	1. Physical
Packet Sniffinf	Network	Surveillance and sniffing on network traffic	4. Transport
WiFi mimicking	Network	Disconnect and reconnect devices to fake networks	1. Physical
SQL injection	Web	Inject malicious SQL queries or database manipulation	7. Application
Phishing	Web/E-mail	High level search for usernames and passwords. Potentially social engineering.	7. Application

B. Certificate Management

In the examples from related research, researchers mainly talk about certificates in IoT and not very specifically about Certificate Managers. What the architecture of an IoT infrastructure often looks like today is that the IoT devices are connected to a Gateway that handles the requests from the server and the data stream coming from the IoT devices. In case an IoT device is not connected to a gateway, the device is often a more powerful Single Board Computer (SBC) that can handle both the work of a gateway. This SBC has enough computing power to encrypt data on its own. One challenge is to secure data from non-SBC devices, as many smaller IoT devices do not have the resources needed to handle certificates, both in terms of memory and processing power.

As mentioned in the research by Zhihua Li [7], PKI is well suited for this as the protocol is mature and robust. It has been used in TCP / UDP for a long time. Falk and Fries [8] presents the use of Certificate Whitelists (CWL) as a method of certificate management. A CWL is a data structure that contains references to the certificate's unique serial numbers and issuers, the list can be trusted if it is signed by the whitelist root key of trust. The backend solution acts as a Certificate Manager, which creates lists of approved certificates so that IoTs do not communicate with unauthorized devices. The idea behind whitelisting certificates is that you can add more restrictions and distribute different lists to each unit based on what other units it needs to communicate with. It should also be possible to place restrictions on what kind of protocols are used, so that entities with whitelisted certificates that do not use the current protocol cannot communicate with the rest of the system. By handing over the lists to the devices, and periodically update and verify them, the device does not have to contact the server each time it communicates with another IoT device. This ensures that if the backend server is unavailable, the device can check the list it has stored locally over which devices to accept.
The backend solution keeps the status of the certificates at regular intervals and checks that the certificates are not withdrawn. It can withdraw certificates even by removing them from their lists [8].

While many view certificates as a good solution for the approval and authorization of IoT devices, there are also ongoing research highlighting that certificate authorities will not be able to handle the scope of 50 billion units of IoT

devices within a few years. There are. They argue that the scope is so large that CAs will not be able to handle the task [9].

III. CASE COMPANY

This research was conducted together with an industrial partner that is a leading national provider of solutions for electronic identification and digital payment solutions. They are the only nation-wide company that offers internationally approved Secure Socket Layer (SSL) certificates. They provide services such as Extended Validation (EV) SSL certificates for both digital identification and signing, and electronic ID for retail and corporate customers. They are currently expanding to include even further focus on digital certificates, public key infrastructure and encryption.

IV. DESIGN AND IMPLEMENTATION

A proof of concept solution was implemented, consisting of four parts: a Certificate Manager, which is a server solution, a WebUI that can be used to administer the Certificate Manager, a client running on an IoT device and a Mock certificate authority for generating certificates.

This solution is meant to be so generic that it can be implemented in any production system regardless of the flavour of the system. Among other things, the IoTs must run Linux, and have enough processing power to encrypt / decrypt. To illustrate the communication flow between the different parts of the Certificate Management system, an outline is shown in the figure below.

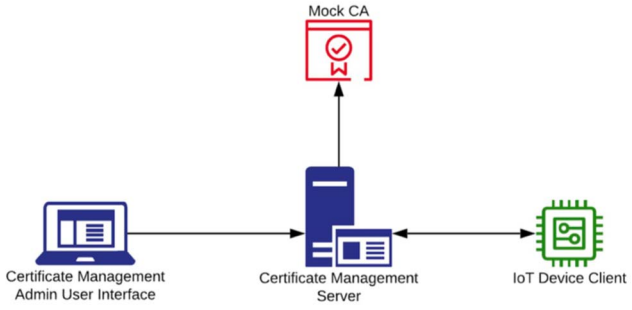

Figure 1 High level component flow

All parts of the solution communicate over APIs, and the arrows in the figure above show which parts call each other. *Certificate Management Admin User Interface* does not offer any API, so all communication goes from this to the server. The *IoT Device Client* has access to one call to *Certificate Management Server* which is the actual registration of the client with the server. The rest of the communication flow goes from the server to both *Iot Device Client* and *Mock CA*.

A. CertificateManagementServer

The server side of the solution is designed as a Representational State Transfer Application Programming Interface (RESTful API). This solution provides functionality on the server over the Internet, which other applications and services can utilize via http API calls. The framework Spring Boot is used to create the APIs and communicate with permanent storage in a database. The Administrator UI and IoT clients use this API to communicate with the server. The API makes it easy for developers to restrict access to the functionality of the server, allowing other developers to create software that uses the server's functionality without being able to access the source code. Further, this also makes the solution modular, since it is easier to replace parts of the solution, such as user interface, and simply call for the same functionality. Therefore, we have also created our own APIs separately for both the administrator user interface and the IoT clients, to have a clear distinction on what functionality should be available for the different parts of the solution. This is something that we have followed in the rest of the code structure, to keep the classes to their purpose, and to keep the responsibilities evenly distributed.

B. IoT Device Client

The client for the certificate management is developed in Java. This can be combined with any configuration file from the server configured through a manual installation of the client. The client provides an API via Spring Boot so that the server can manage the device. The main task of the client is to generate Certificate Signing Request (CSR), send it to the server, receive a certificate, and store it securely.

C. Mock Certificate Authority

For testing the Certificate Manager (CM) a custom Certificate Authority (CA) is implemented. Ultimately the certificates for this solution could become a "subscription" scheme, at the discretion of the industry company. The CM server orders to itself and is stored in a KeyStore on CA and can be used to verify certificate orders for the IoT devices.

Illustrated in Figure 2 below:

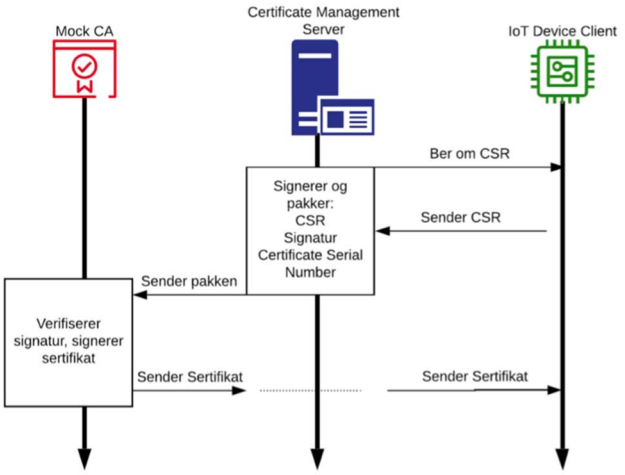

Figure 2 Issue certificate to IoT. Device

When an IoT device orders a certificate, it goes through the CM server. The CM server sends an API call to the device, which responds with a CSR. The CM server signs the CSR with a private key and forwards it to CA who can verify the signature with the public key on the CM server's certificate. It then sends the certificate to the CM who passes it on to the IoT device. The IoT sends back the details that the server needs about the certificate. Of several available technologies to develop a CA solution, the Bouncy Castle framework is chosen due to industrial partner compliance.

V. DISCUSSION

Our software system for Certificate Management is based in Open Source architecture stack from WSO2 IoT-Server and further developed from scratch in accordance with identified characteristics. We decided to only draw inspiration from the WSO2 IoT server and further not implement unnecessary functionality to keep the scope narrow and measurable. The choice of programming language was the next decision to make and all modern approaches from C++, C#, Java and JavaScript was open options. In collaboration with the industry partner and in relation to previous research findings the choice landed on Java with a combination of frontend JavaScript frameworks. The backend of the Certificate Manager was created using Java and the frontend section a RESTful application with a separate application in Node.js using the API implementation. We use the same technology in the client applications as in the backend, a RESTful API that the server can use to control the clients.

As mentioned earlier, less research literature specifically addresses Certificate Managers, although the use of certificates on IoT devices are well investigated. Since certificates are a widespread and robust means of securing communication on the Internet, we got the impression that there is a general consensus that certificates are the way to go. Issues are raised pointing out the fact that many IoT devices use microcontrollers that basically cannot bear the task of managing certificates. From our implementation the same was experienced initially and is why we looked into Hardware Security Module as a possible solution. Others talk about that responsibility for the issuance of certificates should be with the individual manufacturer and not with a CA, the amount of IoT devices will reach a level that the CA is unable to handle [9].

Research articles on using certificates highlight the challenges a CA will face if deploying SSL certificates on IoT devices. The reason for this is the number of units that are estimated to be operational within a few years. One possible solution would have been to allow businesses and individuals to generate their own certificates for their devices so that CAs are not overwhelmed by the number of devices. We believe that if a company generates its own certificates for its own IoT devices, then it can work if the IoT devices only communicate internally within that company, but it does not help for external communication. It also does not help a customer standing and buying an IoT device with the manufacturer's own certificate. It would have been the same result that you could sign your own certificates for your own websites, then all credibility for certificates will disappear. If we move down to the consumer level, we can mostly feel confident that if we buy a Philips smart bulb at an electronic store, we get a credible product. But as also there are many IoT products one gets that certainly do not come with the same degree of security [6]. Here it could be an advantage with a third party who can approve who the manufacturer is. Just because we are dealing with a physical object does not mean that we should go for a lower security and just say that the responsibility is with the consumer. On the contrary, it is perhaps even more important as these are units that become part of our home, society and infrastructure. The typical user often knows less about safety and does little or no safety assessment of the products he / she purchases for his or her own household. So instead of saying that CAs can't handle the task, it is like that a solid software solution from the CAs are needed to perform the task and help improve the security of the IoT.

The certificate manager created should be a link between company and CA. It makes it easier for the business to implement security from the first moment. Using this software solution, you will be able to dynamically expand your IoT network as you can introduce any IoT device into your network, which is automatically handled by the network's security system. If this system is introduced as a kind of standard for IoT networks, CA does not have to deal with requests from billions of devices, but rather each individual IoT network's Certificate Manager.

VI. CONCLUSION

The IoT industry is still suffering from major security shortcomings, despite an increasing focus on the area. This is due to both a lack of security expertise among developers and a lack of good security solutions for IoT infrastructure. When development starts, it is the functionality that places the greatest emphasis and security is either forgotten or there

simply is no time for it. The lack of security is also affected by the resource constraints on many IoT devices. Some IoT devices have the capacity to run an OS, but the devices most widely used today are simple microcontrollers, which often have small, simple, and very specific tasks.

Unfortunately, the overall security level of IoT devices today is low, and some devices are not secured at all. Whether this is necessary depends on the task of the specific IoT device and the data it handles. I.e. the sensor data on the temperature where the data is not confidential, does not need the same kind of encryption. However, data on a fresh-water system that provides drinking water to an entire city it is very critical and should be encrypted.

To resolve this problem, SSL certificates are a secure solution. SSL certificates can be used to encrypt communications between devices to ensure that this communication is private. They can also be used for signing messages sent from the devices to verify where the messages are coming from. With the findings we have presented in this research through a prototype implementation we argue that insight is given into what the IoT industry looks like today. We think this provides a good foundation for companies to move forward with a product strategy should they wish to offer security solutions for IoT. Although our solution is a proof of concept, future work is needed to be able to see the full potential for both making the solution more robust and secure, and expanding with more functionality.

ACKNOWLEDGMENT

We would like to acknowledge K. Abelseth, M. Christiansen, M. Fredriksen and K. Langhaug for originality of ideas, their work and contribution in this project and report.

VII. REFERENCES

[1] McEwen, A., & Cassimally, H. (2013). Designing the internet of things. John Wiley & Sons.

[2] Minerva, R., Biru, A., & Rotondi, D. (2015). Towards a definition of the Internet of Things (IoT). IEEE Internet Initiative, 1, 1-86.

[3] Kafle, V . P ., Fukushima, Y ., & Harai, H. (2016). Internet of things standardization in ITU and prospective networking technologies. IEEE Communications Magazine, 54(9), 43-49.

[4] Russell, B., & Van Duren, D. (2018). Practical Internet of Things Security: Design a security framework for an Internet connected ecosystem. Packt Publishing Ltd.

[5] Goodrich, M., & Tamassia, R. (2018).Introduction to computer security. Pearson.

[6] Russell, B., & Van Duren, D. (2018). Practical Internet of Things Security: Design a security framework for an Internet connected ecosystem. Packt Publishing Ltd.

[7] Li, Z., Yin, X., Geng, Z., Zhang, H., Li, P., Sun, Y., ... & Li, L. (2013, January). Research on PKI-like Protocol for the Internet of Things. In 2013 Fifth International Conference on Measuring Technology and Mechatronics Automation (pp. 915-918). IEEE.

[8] Falk, Rainer, and Carlos Becker Westphall. 2014. SECURWARE 2014 - The Eighth International Conference on Emerging Security Information, Systems and Technologies. IARIA.

[9] Kim, H., Wasicek, A., Mehne, B., & Lee, E. A. (2016, August). A secure network architecture for the internet of things based on local authorization entities. In 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)(pp. 114-122). IEEE.