Secure Bootstrapping of 6LoWPAN Networks to Prevent Vampire Attacks in IoT Environments

Tiago Diogo
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-001 Lisboa, Portugal
tiago.diogo@tecnico.ulisboa.pt

Miguel Pardal
Instituto Superior Técnico
Av. Rovisco Pais, 1
1049-001 Lisboa, Portugal
miguel.pardal@tecnico.ulisboa.pt

ABSTRACT

The Internet of Things (IoT) and its vision of connecting devices to one another and to the Internet presents an opportunity to create large information sharing networks. However, intruders can capture and take advantage of the IoT devices' constrained nature to disrupt these networks and launch a wide range of attacks on its nodes. In our work we propose AutoStrap – a secure bootstrapping scheme capable of securing IoT environments based on 6LoWPAN networks – that aims to prevent "vampire" attacks that drain the energy of the network nodes. To achieve this, we do a thorough analysis of the existing protocols, attacks and mitigation strategies, combining that information into our proposed network management system. Furthermore, we measure the space and energy required for operation in order to assess what kind of physical resources to deploy, based on the network security characteristics.

Keywords

Internet of Things; Power-Aware Security; Secure Bootstrapping; CoAP; 6LoWPAN; RPL; IEEE 802.15.4

1. INTRODUCTION

The Internet of Things can be seen as a web of interconnected devices that range from everyday wearable objects to enterprise grade sensor networks. Despite the huge variety and characteristics of these devices, one characteristic that they all have in common is their constrained nature. In order to enable the massive deployment to be expected in the near future, IoT devices must be accessible and affordable, capable of operating under lossy wireless networks while being battery powered. Section 2 presents an overview of the type of networks and scenarios under consideration. Since IoT environments can range from home to enterprise or even city environments, a breach in security could potentially leak important company activity or provide information about individuals choices and whereabouts constituting a violation of privacy [15]. To this extent a study on existing attacks for constrained devices was conducted in Section 3 and a common mitigation strategy (secure bootstrapping) presented in Section 4. This strategy provided security assurances at the cost of an increased infrastructure complexity, which in its turn created the necessity for a management station capable of storing and writing network credentials onto new nodes. AutoStrap is further analysed in Section 5.3 and the supporting system architecture presented in Section 5.4. In order to accurately define the type and amount of resources needed to create such a network, we deployed our protocol stack and management station on physical hardware and performed space and power measurements as presented in Section 6. Finally, Section 7 states our conclusions and opportunities for future work.

2. ENVIRONMENT OVERVIEW

There are many applicability domains and different methods for creating IoT networks. Some provide direct connectivity of nodes to the Internet while others provide a common gateway for interfacing with external networks. In our work, we focus on scenarios where network nodes are not directly connected to the Internet and require additional network components for proper communication as depicted in Figure 1.

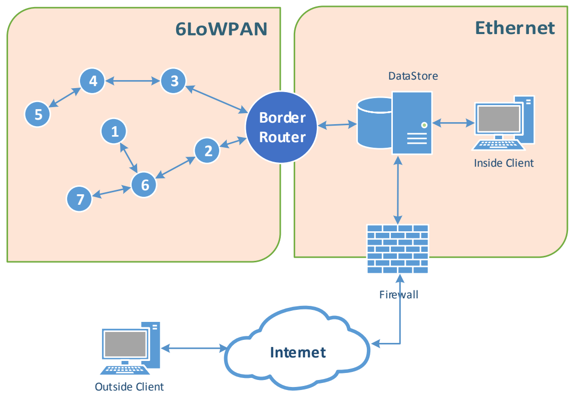

Figure 1: IoT Network Overview.

In this type of architectures, the sensing or actuating nodes belong to a very constrained network with specific protocols and header compression mechanisms, requiring an interface device – the border router – in order to communicate with external networks. After reaching the external network, incoming messages are processed to convert sensor data into useful information which is then stored or used to trigger events. This information can then be accessed by users either on the same network or by making requests through the Internet.

This type of deployment could be used, for example, in a home intrusion system or in a factory monitoring system.

In the home intrusion system scenario, the network nodes would create a sensor network that propagates events in the case of an intrusion and the additional infrastructure would be in charge of receiving these events and notifying the police authorities. In the factory monitoring system scenario, the network nodes would create a sensor network that would be permanently reporting up-to-date readings of machinery control values like: temperature, pressure and power; and the additional infrastructure would in charge of supplying this information to a dashboard for the factory workers. If an attacker could disable these systems, he could then cause an emergency shut-down of the factory machinery due to lack of control over the working conditions. These are real concerns backed by a range of attacks that focuses on disabling IoT networks by placing the nodes offline. These attacks are presented in the following Section.

3. ATTACK ANALYSIS

Exploitation of existing solutions in the form of malicious attacks can be found across the Open Systems Interconnection (OSI) layers. They can range from a physical intruder replacing some node on a sensor field to the well-known Denial of Service (DoS) at the application layer. However, given the characteristics of the devices and networks used in IoT combined with the power consumption focus of this work, a specific kind of attacks performed at the network layer is of special interest and importance: battery depletion attacks, also known as "vampire" attacks.

Battery depletion attacks aim at draining the battery – "life" – of the network devices, working over time to entirely disable a network, hence being called "vampire" attacks. These attacks do not focus on flooding the network with many packets, instead they drain the node's life by delaying the packets transmission. Some of the attacks target specific implementations while others are independent of the protocol [16][12]. In the following paragraphs, we present the most relevant attacks, performed on different routing solutions that demonstrate the draining power of such methods.

3.1 Stateless Protocols

In systems that use this type of routing protocols, the source node specifies the entire route to the destination in the packet header. This means that intermediaries do not make decisions regarding the next hop, they only forward to the next node as specified in the original path therefore reducing the amount of computation performed and the used energy. Using this transmission scheme, a malicious device can specify paths through the network that are far from optimal, wasting energy at the intermediate nodes who follow the included malicious source route. The *Carousel Attack* is an example of such an attack. Its objective is to send a packet along a route composed of a series of loops. This way, a single node may forward the malicious packet several times increasing the total energy consumption by a factor of the number of loops the attacker has introduced on the packet header path. It targets source routing protocols by exploiting the limited verification of the packet headers at the intermediary nodes. Figure 2 shows an example where a "vampire" node created a path composed of circles around the network when it could have exited after the first hop through the D node.

3.2 Stateful Protocols

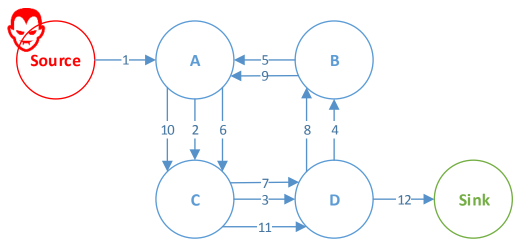

Figure 2: Carousel Attack.

In systems that use this type of routing protocols, network nodes are aware of the network topology and its state, being able to make local decisions on the node to whom they will forward the packet. The effect of the "vampires" in this type of routing is limited since the route is built dynamically from many independent forwarding decisions. However, attackers can still cause damage by forcing packet forwarding through nodes that would not be on the optimal path, for example, by forwarding the packet back to the source. The *Directional Antenna Attack* is an example of such an attack. In it, the attacker takes the role of an intermediary and not the source of a packet. If the attacker has the resources to use a directional antenna, it can deposit a packet on arbitrary parts of the network while also forwarding the packet locally. This causes nodes that were not on the optimal path to also consume energy by forwarding a packet they would not normally receive, therefore increasing the total energy consumption by a factor of the directions the attacker can position the antenna and the distance between the receiver and the sink. Figure 3 shows an example where a "vampire" intermediary deposited a node on a distant location of the network, causing the packet to follow two different routes towards its destination

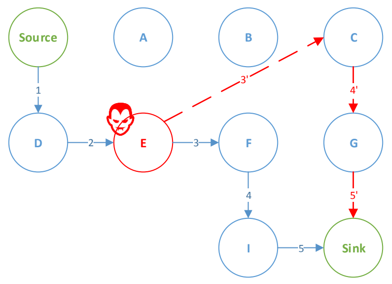

Figure 3: Directional Antenna Attack.

While it is true that there are mitigation strategies available for these attacks [16], those strategies imply that additional checks and validations are performed on the network nodes. For example, nodes could check the packet headers for loops in the message path, and if found, update the route or simply drop the packet, therefore mitigating the carousel attack. If we were to mitigate the directional antenna attack, we could, for instance, analyse the route paths of a given packet that reached the sink more than once. The last node identifier to appear duplicated before the path started to diverge would be the one who directed the packet to multiple regions, therefore revealing the attacker. Unfortunately, these mitigation strategies are placing a burden on the already constrained nodes. For each new attack that we wish to mitigate, additional checks and validations must be performed, eventually reaching a point were the costs involved in validating each packet could be considered an attack on the node resources. To address this issue, we propose that a *secure bootstrapping* is performed for every device that joins the network. The following section details the bootstrapping, how can it be performed and some existing solutions that follow this approach.

4. SECURE BOOTSTRAPPING

The term bootstrapping is applied to the process in which a new device is connected to an existing network. To achieve a secure bootstrapping, a unique identity and security parameters are associated with the device before this phase. There are several ways to carry out the initial setup, either via a wired or wireless interface. In the case of wireless bootstrapping, attention must be given to eavesdropping so that the secure credentials cannot be intercepted. Since the presented "vampire" attacks are to be performed by a malicious intruder capable of interacting with the network, if we could assure a secure bootstrapping, meaning that the new node would be authenticated before becoming an active member of the network, those attacks could no longer be performed.

Secure bootstrapping and network admission solutions have already been proposed in past literature. However, the development and optimization of application layer protocols as well as network layer routing schemes allows for new approaches and solutions that can now fit the nature of IoT devices. Bergman et al.[2] proposed a three-phase secure bootstrapping technique for nodes in a Constrained Application Protocol (CoAP) network. Firstly, the joining node broadcasts a request for a CoAP Service Discovery Server (CSDS). This server, once contacted by a new node, takes the responsibility of key distribution. Then, the system goes under a vulnerable phase where the secret is transmitted from the CSDS onto the new device. The authors propose a short audible or visual feedback to the human installer when the secret is received and assumes that potential eavesdroppers can not intercept this transmission. Finally, this secret is used to setup the Datagram Transport Layer Security (DTLS) connection. This approach has major security drawbacks on the secret transmission phase, so the authors propose limiting the radio power to a low level and disable data forwarding beyond the local network segment, but these techniques cannot assure that an attacker will not be able to intercept the transmission.
Oliveira et al. [10] proposed an admission control solution for 6LoWPAN networks based on administrative approval. Each joining node would broadcast its presence to the network, and that broadcast would be received by the administrator in the management server. Then, the administrator would grant access to that new device based on its address, and that information would be transmitted to all the devices in the network. After this phase, the device would be allowed communication as a regular member of the network by its neighbours. This approach has the advantage of requiring no previous setup on the device before operation, but it is vulnerable to Spoofing Attacks, where as attacker obtains access to the network by bypassing the Access Control List (ACL) with the identifier of a legitimate node. The authors state that some work still needs to be performed in order to validate the sensor identity and leave the pre-instalment of keys on the device as a likely possibility for the deployment of the system.

Besides these related work efforts, there are additional secure bootstrapping techniques [6] that rely on tokens and one-time-passwords for cyphering the first communications and receiving security credentials, or even approaches where the manufacturers deploy the security credentials during the manufacturing process of the device. Still these approaches either require additional hardware to be employed or the need to trust the manufacturer credentials.

In our work, we propose a secure bootstrapping method where:

- No additional hardware needs to be employed during the bootstrapping process;
- No additional security credentials need to be fetched after the deploy on the field;
- No security credentials need to be generated and uploaded to the joining nodes by third parties;
- No initial message exchange is sent unencrypted and/or unauthenticated.

We called our solution AutoStrap because of the need for a simple, automated solution that prevents de "vampire" nodes from entering the network.

In the following Section, our complete infrastructure proposal is presented in terms of objectives and requirements, architecture, used protocols and implementation details.

5. PROPOSED INFRASTRUCTURE

IoT's principle of connecting every device to one another in an automated way can create networks beneficial to a wide range of applications, from home environments to large enterprises or even cities. However these are domains that have different requirements. A home application should be easy to setup without complex configurations. An enterprise solution can benefit from additional administrative configurations as long as the deployment of the network nodes is done quickly due to their potentially large number. In order to demonstrate the capacities and applicability domain of our system, we will use a Smart University Campus scenario since it is our belief that it can effectively demonstrate the needs targeted by our work. In the following sections we will apply the information gathered in terms of attacks and mitigation strategies to find suitable communication protocols and define our objectives and requirements. Then, a model of a university campus with our proposed power-efficient network architecture will be presented and their components' roles explained.

5.1 Objectives and Requirements

A major concern amongst IoT application is the communication model. It is our goal that the entire network is power-aware, using as minimum energy as possible. Also, it is very important that the deployment of new nodes and system maintenance can be performed by regular staff members without knowledge of the inner workings of the system.

This creates usability challenges and requires simple and automated interfaces and bootstrapping processes. Additionally, the following set of requirements is critical in order to allow secure communications to take place:

- Confidentiality: Without confidential message transmission, packets would flow in the network in plain text. Attackers could sniff the packets in order to obtain network activity information constituting a breach in security. Even if there is no critical data being sent, privacy is still compromised;
- Integrity: Assuring message integrity means that the message was not modified between the source and its destination. Without integrity we could not rely on the received data since it could have been, intentionally or not, modified on-the-fly, and be supplying wrong information to the system;
- Authentication: The studied type of networks relies on hop-to-hop communication, meaning several nodes will take place in forwarding a packet. If they are not authenticated they could perform a wide range of attacks and disrupt the network.

5.2 Protocol Stack

There are many alternatives and some proposed standards when it comes to choosing a protocol stack for IoT communications [1] implying that a thorough analysis of the existing solutions is necessary to analyze the strong and weak points of each candidate.

5.2.1 Data Link and Physical Layer

The first requirement for the physical layer of the IoT is the use of wireless radios. These should aim for simplicity, low-power and low-cost communications. While wireless communication is widespread and can be found in deployment scenarios ranging from homes to airports, the type of radio commonly used, known as Wi-Fi [8], uses a high amount of power causing concerns for battery life. IEEE 802.15.4 [9], on the other hand, was created for Low-Rate Wireless Private Area Networks (LR-WPAN) and its specifications focus on low power consumption, low data rate, low cost and high message throughput make it a strong candidate for IoT applications.

5.2.2 Network Layer

IoT's vision and its massive deployment can only be achieved through the use of IPv6 [11] because of the need of many more individual IP addresses. However, physical layers more suitable for communication over constrained networks pose some limitations to the use of the IPv6 messages. For example, the limited packet size in IEEE 802.15.4 based networks. To tackle these issues, the Internet Engineering Task Force (IETF) IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) [13] working group developed a standard based on header compression to reduce the transmission overhead, fragmentation to meet the IPv6 Maximum Transmission Unit (MTU) requirements and forwarding to link-layer to support multi-hop delivery [7]. 6LoWPAN is able to remove a major share of IPv6 overheads, being able to compress its headers to two bytes, therefore allowing small IPv6 datagrams to be sent over IEEE 802.15.4 networks.
With the use of 6LoWPAN, routing protocols can now use the IPv6 addressing scheme. Given the possible frequent topology changes associated with the radio-link instability, successful solutions must take these requirements into account on their specification. Routing Protocol for Low-Power and Lossy Networks (RPL) [17] can support a wide variety of link-layers and is prepared for devices with very limited resources. It is able to build up network routes, distribute routing knowledge among nodes and adapt the topology in a very efficient way. Furthermore, RPL has a built-in topology repair mechanism that acts in the case of a routing topology failure, link failure or node failure.

5.2.3 Application Layer

CoAP [14] is a document transfer protocol based on REpresentational State Transfer (REST) on top of Hypertext Transfer Protocol (HTTP) functionalities. CoAP's objective is to enable tiny constrained devices to use RESTful interactions, where clients and servers expose and consume web services using Universal Resource Identifiers (URIs) together with HTTP Get, Post, Put and Delete methods. Unlike REST, CoAP runs over User Datagram Protocol (UDP) instead of Transmission Control Protocol (TCP) which makes it suitable for full IP networking in small micro-controllers. Retries and reordering are implemented at the application stack using a messaging sub-layer that detects duplicated messages and provides reliable communication using different types of messages. Confirmable messages must be acknowledged by the receiver, nonconfirmable follow the fire-and-forget model. Despite being a lightweight protocol, CoAP still provides important features:

- Resource Observation - CoAP can extend the HTTP request model with the ability to observe a resource, therefore monitoring resources of interest using a publish/subscribe mechanism;
- Resource Discovery - CoAP servers provide a list of resources using well-known URIs that allow clients to discover what resources are provided and their types;
- Interoperability - since CoAP is based on the REST architecture, a simple proxy enables CoAP to easily interoperate with HTTP.

A final overview of the presented stack for IoT communication is presented in Table 1 with the objective of comparing it to the protocol stack commonly used in the Internet.

Table 1: Protocol Stack Comparison Overview

Layer	Web	IoT
Application	HTTP	CoAP
Transport	TCP	UDP
Network	IPv6	6LoWPAN
Data-Link/Physical	802.11	802.15.4

5.3 AutoStrap

Previously presented bootstrapping methods had the drawbacks of either requiring additional hardware during the commissioning phase, the need to fetch additional credentials after starting the operation phase, or even relying on third party entities to provide those security credentials. Given the characteristics of our target infrastructures, in which there is a need for a management station capable of interpreting data coming from the constrained network, the rationale for designing this bootstrapping scheme was to address all these drawbacks at once with an additional infrastructure component, the bootstrapper. By uploading the necessary security credentials through the use of the bootstrapper component, new nodes would start the operation phase with all the security credentials required for secure communications. Besides the bootstrapper, only a key store for the network credentials is required as additional infrastructure components. More in depth, the security credential provided to the joining nodes is a 128bit key, used for the Advanced Encryption Standard (AES) [5] in Counter with CBC-MAC (CCM) star mode [3]. This allows packets to have their payload encrypted and the message authenticated with a Message Integrity Code (MIC) also computed from the AES-128 key. This assures that all the packets flowing through the network, even the joining node first communications, are confidential and authentic. This way, attackers are not able to join the network topology therefore frustrating all attempts to conduct battery depletion attacks. Since the infrastructure administrators are not required and possibly unaware of this process, bootstrapping needs to be performed in an automated way, preferably together with an already existing task. The chosen approach was to upload the security credentials during the firmware uploading of the joining nodes, through the use of an user interface where the operator simply needs to select the type of hardware being commissioned and all the required tasks run in the background. These involve creating an unique identifier for the joining node, fetching the AES-128 key from the key store, storing it in the node firmware and finally upload the new firmware containing the security credential to the new device. A sequence diagram of the full process can be seen in Figure 4.

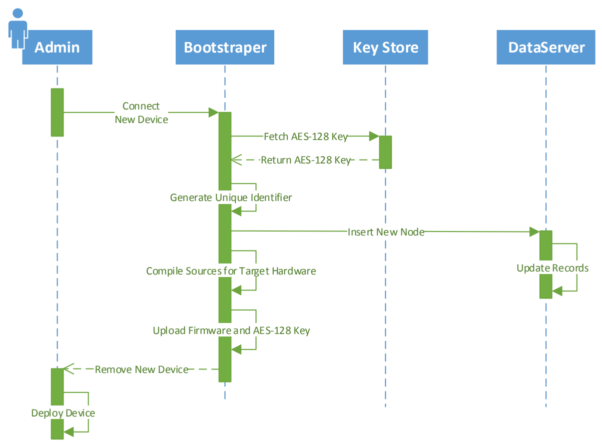

Figure 4: Bootstrapping Process

5.4 System Architecture

As previously stated, we will use a Smart Campus scenario. Being aware of the technological improvements on sensor networks and building management technologies, the campus administration decided to improve the monitoring of the overall conditions of the buildings and inside environments in order to better preserve its assets. To cope with the new requirements, we propose a solution for the monitoring of the campus sections by deploying a wireless sensor network on each building, connected to a central management station operated by the available staff. The scenario will be based on the Instituto Superior Técnico (IST) campus model. An overview of the system and its components over the IST blueprints can be found in Figure 5. Regarding each individual component:

- Numeric Nodes: Represent the network sensor nodes, the most constrained element of the network. They cooperate to build the topology and route messages hop-by-hop until the root is reached. These are fully equipped with the previously presented energy efficient protocol stack;

- Alphabetical Nodes: Represent the root node of each section network topology. They are equipped with the same stack of the numbered nodes but are more powerful, preferentially not battery-powered and act as the bridge between the constrained 6LoWPAN environment and the central management station. These nodes must be more powerful than the numeric ones so that they can process all the requests between a group of sensors and the management station. Also, although the numeric nodes use low-power wireless radios, the alphabetical nodes must be capable of interfacing with more power consuming radios and protocols, therefore requiring more resources. This differentiation allows numeric nodes (the large portion of the network devices) to keep their very constrained nature, consuming less energy, while still being able to communicate with external devices;

- Management Station: A black box model of the core components of the system. Each building reports to the central station and the staff monitors the network status through it; A white box model will be shown in the following sections.

- Client: The system's clients can be any user with access credentials, but mostly the staff members. They can access the management station either from within the local network or from outside through the Internet.

Central Management Station.
The central management station is divided in five main components. A white box schematic of the core components and interactions can be found in Figure 6. Regarding each individual component:

- Key Store: This component is responsible for storing the security credetials used by the network nodes to achieve secure communications.

- Bootstrapper: The bootstrapper acts as the interface between the management station and the network devices. It generates the device identifier and writes it together with the shared network key into the new device;

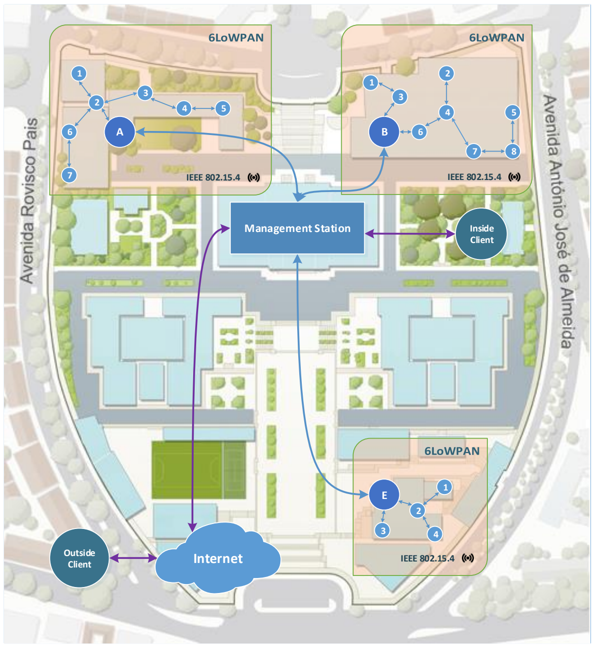

Figure 5: Global System Architecture

- CoAP Client Observer: The one and only client in the network. Instead of having users directly requesting the sensor readings, the client will observe each resource and be notified of the new value. Each time it receives an update, it stores the information on the Data Server for the clients to use;
- Data Server: A database with mappings of each node to the most up to date value reported. It is updated by the client observer and used on demand by the clients;
- Proxy: Responsible for bridging requests coming from the Internet to the Data Server. Responsible for authenticating the external clients and providing access to the Data Server information.

Although each user could access the system through a CoAP terminal and request the most up-to-date readings from the sensor nodes, this approach would cause unnecessary overheads in the system. Since many clients can connect from different locations, many requests would be performed to the sensor nodes for the same information. This would mean additional memory usage in the physical devices, and more requests to the already constrained battery operated network. With the single client approach acting as an observer, only one message needs to go through the network for each new reading.

6. EVALUATION

In this Section we measure and profile the resources required for the system operation in order to discover if our choice of protocols and mitigation strategies is suitable for IoT hardware. This ranges from the memory required for the nodes operating system and protocol stack, to the hardware

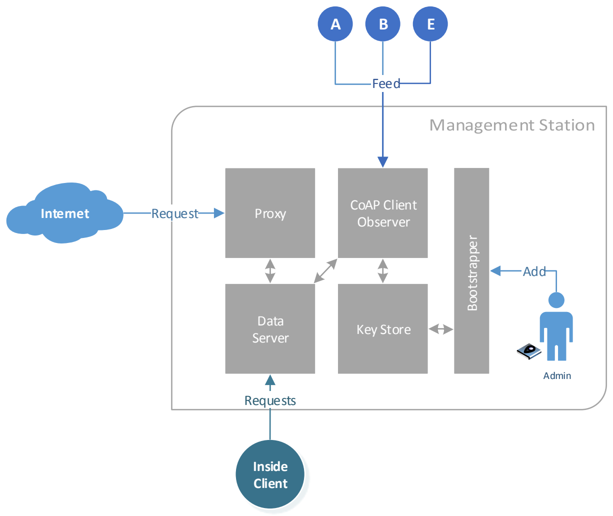

Figure 6: Central Management Station

Table 2: Memory Usage

Security Mechanism	Flash(KB)	RAM(KB)
No-Sec	59.56	13.54
LLSec	61.36	13.80

power consumptions. Furthermore we evaluate our bootstrapping infrastructure by measuring the time required for uploading a firmware to the new device and the ease of use by counting the steps required for the process to take place. Each following subsection both presents the collected data and explains the process and technologies involved in obtaining it.

6.1 Hardware Suitability

In order to evaluate if the IoT hardware is capable of supporting our choice for the stack of protocols security measurements it is necessary to measure the firmware size. For that task, the *msp430-size*[1] tool was used. This tool analyses the firmware file and outputs the amount of *text*, *data* and *bss*. *Text* corresponds to code and constants, *data* is for initialized variables and *bss* is for uninitialized data (which is initialized with zero in the startup code). The total amount of flash memory can be calculated from the sum of the text and data parameters. The total amount of Random-Access Memory (RAM) memory can be calculated from the sum of the data and bss parameters. Both the firmware without security mechanisms and the one with link-layer security were analysed for their Flash and RAM usage. The results can be found in Table 2. The inclusion of link-layer security software represents a 3.02% increase in Flash usage and a 1.02% increase in RAM usage.

In order to be considered an IoT capable device for this work, the selected hardware should possess a 802.15.4 radio, be capable of being battery powered and provide development tools like sensors and actuators that would be mapped to the application layer endpoints. The board also

[1]http://www.ti.com/tool/msp430-gcc-opensource

Table 3: Hardware Memory

Device Name	Flash(KB)	RAM(KB)
Zolertia RE-Mote	512	32
Arago Systems Wismote	128	16
Texas Instruments CC2538DK	512	32

Table 4: Power Consumption

Mode	Voltage(V)	Current(mA)	Power(W)
Radio ON	9.0	17	0.15
Radio OFF	9.0	5.2	0.05

needs to be compatible with the selected operating system and provide low-power modes of operation. With all this in consideration, the Zolertia RE-Mote[2] board was selected since it fulfilled all the requirements and also provided an integrated cryptoprocessor while maintaining a low-power operation. On the other hand, to assure our firmware size is a proper fit for current IoT hardware we did not not only verify that if fits the Zolertia RE-Mote memory but also compared it to two other boards also designed for IoT applications. The boards are the Arago Systems Wismote[3] and the Texas Instruments CC2538DK[4]. The device's memory is shown in Table 3 and allows us to conclude that our solution is practical for the target hardware.

6.2 Power Consumption

The introduction of security mechanisms in low-power networks is necessary and desirable. However, due to the low-power characteristics of IoT sensor and actuator networks, a substantial increase in power consumption can disrupt the network by quickly draining the available resources. To this extent, a power consumption analysis was performed on our selected board in order to determine the most suitable battery powering solution. We performed current measurements during periods of silence as well as during periods of radio activity and calculated the power consumption in Watts(W) from the Power Equation $P = IV$, where I is the current in Amperes(A) and V is the voltage in Volts(V) as presented in Table 4.

Knowing that the ContikiMAC Radio Duty Cycling (RDC) protocol enables the radio to be turned off for roughly 99% of the time[4], the collected data shows that our solution maintains the low-power consumptions required for IoT environments.

6.3 Bootstrapping Process

In order to evaluate our bootstrapping process we conducted an experiment on the amount of time (in seconds) required to bootstrap a new node as an indicator of the process efficiency. The process starts with the operator connecting the new device to the bootstrapper and finishes with the operator removing the device from the bootstrapper. During the process the operator needs to open the user interface, select the appropriate device and press the upload button. In the background the system will compile the source code to match the target platform, then it will erase the previous firmware and finally upload the new one. The experiment results are presented in Table 5. Although the apparent bootstrapping process time is 34 seconds, the process efficiency is especially important when bootstrapping multiple devices. In this case, after the first one, the user interface and target hardware will already be open and selected and the source code will already be compiled, resulting in a real time of 12 seconds. If needed, an operator could perform up to 300 bootstrapping processes in a single hour, allowing us to conclude that our solution is practical for the presented scenario.

[2]http://zolertia.io/product/hardware/re-mote
[3]http://www.wismote.com
[4]http://http://www.ti.com/tool/cc2538dk

Table 5: Bootstrapping Timings

Operation	Time(s)
Insert New Device	2
Open User Interface	5
Select Target Hardware	4
Compile Source Code	13
Erase Previous Firmware	5
Upload New Firmware	3
Remove New Device	2

7. CONCLUSIONS

Due to the limitations of IoT devices, achieving secure communications is not an easy task. In order to allow the deployment of battery powered nodes, their communication model must be very efficient and consume the minimum amount of power required for operation. To achieve those requirements we started by analysing the existing protocols across the OSI layers, trying to find the best suited solutions for this type of environments. After a thorough comparison we achieved a working stack of protocols but soon discovered possible breaches and attacks, especially on the network layer. Those attacks were further investigated and catalogued. Given the common principle on the majority of the attacks, the introduction of rogue nodes to the network, we presented some possible solutions based on secure bootstrapping, the secure authentication of new nodes when joining a network. Once the energy efficient stack, possible attacks and mitigation strategies were defined, we proposed our solution based on a Smart Campus scenario. This solution is focused on providing the joining devices all the secure credentials required for a secure bootstrapping before the deploy on the field, so that when they start the operation phase no additional credentials need to be fetched, implying that no additional energy is spent on configuration. We evaluated our system with regard to hardware applicability, power consumption and bootstrapping efficiency, concluding that it is a good fit for IoT applications. As future work, memory access protection should be addressed in order to prevent the stealing of secure credentials from deployed devices.

8. REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys & Tutorials*, PP(99):1–1, 2015.

[2] O. Bergmann, S. Gerdes, S. Schafer, F. Junge, and C. Bormann. Secure bootstrapping of nodes in a CoAP network. *2012 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 220–225, 2012.

[3] C. Corp. Formal Specification of the CCM * Mode of Operation René Struik Voice : Fax : Re : Abstract as well as some (informational) design rationale . This document is an edited Purpose Notice. pages 1–19, 2005.

[4] A. Dunkels. The ContikiMAC Radio Duty Cycling Protocol. *SICS Technical Report T2011:13 , ISSN 1100-3154*, pages 1–11, 2011.

[5] N. Fips. 197: Announcing the advanced encryption standard (AES). *... Technology Laboratory, National Institute of Standards ...*, 2009:8–12, 2001.

[6] K. Fischer, J. Geßner, and S. Fries. Secure Identifiers and Initial Credential Bootstrapping for IoT@Work. *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 781–786, 2012.

[7] J. Hui and D. Culler. Extending IP to low-power, wireless personal area networks. *IEEE Internet Computing*, 12(4):37–45, 2008.

[8] IEEE. *IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, volume 2012. 2012.

[9] IEEE Computer Society. *Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, volume 2011. 2011.

[10] L. M. Oliveira, J. J. Rodrigues, C. Neto, and A. F. de Sousa. Network Admission Control Solution for 6LoWPAN Networks. *Proceedings of the Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 472–477, 2013.

[11] J. Pickard, A. Y. Patrick, and A. Robinson. Analysis of enterprise IPv6 readiness. *SoutheastCon 2015*, pages 1–7, 2015.

[12] P. Pongle and G. Chavan. A survey: Attacks on RPL and 6LoWPAN in IoT. *2015 International Conference on Pervasive Computing (ICPC)*, 00(c):1–6, 2015.

[13] Z. Shelby, S. Chakrabarti, E. Nordmark, C. Systems, C. Bormann, and Ericsson. Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs). https://tools.ietf.org/html/rfc6775, 2012.

[14] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). https://tools.ietf.org/html/rfc7252, 2014.

[15] A. Ukil, S. Bandyopadhyay, and A. Pal. Privacy for IoT: Involuntary privacy enablement for smart energy systems. *2015 IEEE International Conference on Communications (ICC)*, pages 536–541, 2015.

[16] E. Y. Vasserman and N. Hopper. Vampire attacks: Draining life from wireless ad Hoc sensor networks. *IEEE Transactions on Mobile Computing*, 12(2):318–332, 2013.

[17] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. https://tools.ietf.org/html/rfc6775, 2012.