Security and AAA Architectures in an IoT Marketplace

Alberto Blázquez Rodríguez
Abstract

Security and AAA Architectures in an IoT Marketplace

Alberto Blázquez Rodríguez

Nowadays, the concept of Internet of Things has revolutionized the industry, making possible the existence of billions of devices worldwide that constantly produce vast amounts of data. This valuable information may only be processed and filtered by complex applications that will eventually sell the information as products. As a consequence, protection of this data and secure, authorized access turn out to be key priorities in this field. A study about authentication protocols has been carried out, according to certain security factors, giving as outcome that OpenID Connect might be the best solution to the problem. Therefore, an authentication / authorisation module has been developed for an IoT platform to protect its data. Evaluations made for the module stated that the protocol and its implementation assure enough levels of security. Nonetheless, accounting features and a better user privacy management are still open issues that avoid fully protection of data in an open technology as the Internet.
Acknowledgement

First off, I would like to thank Konstantinos Vandikas for the opportunity he gave me, letting me carry out my master thesis at such a great corporation as Ericsson Research.

I’d like to thank Sebastien Pierrel and Maxim Teslenko as well for their patience, helping me when I was completely lost.

Special mentions also to my friends Iakovosk, George, Quentin, Sookie and Andreas for sharing with me such a great, sometimes painful, but definitely exciting experience at the company.
# Contents

Glossary ................................................................. 1

1 Introduction .......................................................... 3

2 Background ............................................................. 7
   2.1 Internet of Things .................................................. 8
   2.2 New Value Chains .................................................. 9
   2.3 Smart Cities ....................................................... 11

3 Problem Description .................................................. 13
   3.1 Problem Statement ................................................ 13
   3.2 Access Control and AAA Architectures ....................... 14
   3.3 Limitations and Scope .......................................... 16

4 Methods ............................................................... 17
   4.1 Factors Studied in AA Protocols ............................... 17

5 State of the Art ....................................................... 21
   5.1 IoT Framework .................................................... 22
   5.2 AAA Protocols .................................................... 23
      5.2.1 Alternative 1: OpenID Connect .......................... 23
      5.2.2 Alternative 2: The SENSEI Project ..................... 30
      5.2.3 Final Comparison ........................................... 36
   5.3 API Protection Designs ......................................... 37

6 Implementation of the AA Module ................................ 39
   6.1 IoT Framework’s Architecture Design ....................... 40
   6.2 Trust Model ....................................................... 41
   6.3 User Creation and Identity Management ..................... 42
      6.3.1 Server-Based Flow ......................................... 42
      6.3.2 Client-Based Flow ......................................... 44
      6.3.3 Non-OpenID Flow .......................................... 45
   6.4 API Protection ................................................... 46
List of Figures

2.1 The architecture of the Internet of Things [27] ....................................... 8
2.2 From devices to information products, the new value chains in the market [37] ........................................................................................................... 9
2.3 Potential applications and use cases in a future Smart City [20] .... 11
3.1 Diagram of a possible IoT architecture featuring AAA [22] ............ 15
5.1 TCP/IP architecture with the identity layer on top [31] ................. 23
5.2 OpenID authentication in prior versions explained by Nat Sakimura [30] 24
5.3 Pseudo-authentication using OAuth explained also by Nat Sakimura [30] 25
5.4 OpenID Connect authentication using OAuth as a base protocol [30] 27
5.5 OpenID Connect’s claims aggregation and distributed claims [30] ... 28
5.6 SWOT analysis for the OpenID Connect protocol ......................... 29
5.7 The SENSEI philosophy consists of the abstraction of physical devices [1] ............................................................................................................. 30
5.8 Lightweight SENSEI architecture with full AAA support [3] ......... 31
5.9 Baseline version of the SENSEI architecture [4] .......................... 32
5.10 SWOT analysis for SENSEI, based on the SAML protocol ......... 35
6.2 User creation using the server-based flow .................................. 43
6.3 User creation using the client-based flow .................................. 44
6.4 Creation of non-OpenID users as a fallback mechanism .............. 45
6.5 Step 1: Authenticate the user who requested a resource .............. 47
6.6 Step 2: Authorise and grant access if the request matches certain policies ................................................................................................. 48
6.7 Access token renewal flow ......................................................... 49
7.1 How a “Man-in-the-Middle” attack works between two end users [24] 53
List of Tables

2.1 Common use cases in a future Smart City .......................... 12
5.1 Comparison between the SENSEI project and OpenID Connect .... 36
6.1 Access policies used by the AA Module to grant access to the API .. 50
7.1 Prevention for security attacks and exploits .......................... 56
Glossary

This following list defines the acronyms and abbreviations of technical terms used in the thesis, in order to make easier the reading of this report.

**AAA** Authentication Authorisation and Accounting

**API** Application Programming Interface

**FIM** Federated Identity Management

**HTTP** HyperText Transfer Protocol

**HTTPA** HTTPA with Accounting

**HTTPS** Hypertext Transfer Protocol Secure

**IoT** Internet of Things

**IdP** Identity Provider

**JSON** JavaScript Object Notation

**REST** Representational State Transfer

**RP** Resource Provider

**SAML** Security Assertion Markup Language

**SOAP** Simple Object Access Protocol

**SP** Service Provider

**SQL** Structured Query Language

**SSO** Single Sign On

**TLS** Transport Layer Security
Security in IoT Marketplaces

**WS&AN** Wireless Sensor and Actuator Network

**WS** Web Services

**XML** eXtensible Markup Language

**URL** Uniform Resource Location
Chapter 1

Introduction

Nowadays, technology has made possible the existence of billions of devices worldwide that constantly produce data. This vast amount of data could be analysed and processed by computer systems in order to about temperature, humidity, speed, location, air pressure or pollution, among others. This is the core idea behind the concept of “Internet of Things”, which essentially consists of a huge heterogeneous network of data and devices from different kinds that provide, as a result, lots of meaningful and useful information, making human life easier.

Although all this information is important by itself, it could be also combined in order to offer value-added products and services. Definitely, these outcomes might become, or even create, new business opportunities in the current market if companies are smart enough to take advantage of them.

Therefore, the aim of this thesis lies on considering potential ways of preserving and provisioning access to such valuable information. To fulfil this, we are going to create an Internet of Things Marketplace that will store and organize all these important resources in a simple way.

In consequence, the thesis will cover the access control issue, offering valuable resources but preventing malicious uses and attacks. The security system is achieved by an authentication/authorisation module that will protect such information. The purpose of the project may be divided into the following small, specific goals:
1. The study of the current protocols, mechanisms and open standards to provide secure authentication and authorisation (AA), using concepts such as Federated Identity Management [36] that allow users to provide their public identity as credential for getting access to certain information resources. As prior art here we will consider protocols such as OpenID [29], OAuth [16] and SENSEI [1] (available in Section 5.2).

2. The study of mechanisms to protect the API [40] of service providers, letting only authenticated and valid users to manipulate data and resources, preventing the system from malicious users. This study will take into account the selected protocols for the AA issue, discussing their suitability in the Internet of Things area. Two alternatives: the Amazon design and the design we propose are fully explained in Section 5.3.

3. The implementation of a module featuring “authentication” [5] capabilities for users, using the mechanisms and protocols eventually selected in the study. A detailed explanation of how the module works, with flow diagrams, may be seen in Section 6.4.1.

4. The implementation of a module featuring “authorisation” [5], according to the policies (Section 6.5) and rules that determine who might or not access the resources and services our system provides.

5. The extension of existing open source software, the IoT framework [2] developed in a course context in the Uppsala University. This extension will transform the framework into a secure system that might evolve in the future to a marketplace with accounting [5] features. Further information may be found in Section 5.1 and in the Appendix A.
Contents of the Report

The content of this thesis report has been organized and divided into 9 chapters plus the bibliography. The following list explains briefly what is each chapter about:

Chapter 2 explains the Background of the project and the reasons that motivate this work.

Chapter 3 defines the Problem addressed by the thesis, among the expected requirements, threats and scope.

Chapter 4 gives an overview of the Methodology used in the thesis, following a scientific approach, to study the problem and implement a solution.

Chapter 5 includes the details about the study of the State of the Art, specially establishing a comparison between the two main technologies: OpenID Connect and the SENSEI project.

Chapter 6 was written to explain the developed Solution.

Chapter 7 describes the Evaluation plan that has been carried out, based on a list of well-known techniques and attacks, to prove the security and robustness of the implemented system.

Chapter 8 includes the Conclusions reached after the work was performed.

Chapter 9 contains the Future Work expected and further research that potentially may arise from this thesis.

In addition, after the chapters listed previously and the bibliography, the reader may find some additional information about the thesis in the following 2 appendices, in case he/she might feel interested:

Appendix A shows a couple of screen shots about the interface of the IoT Marketplace.

Appendix B displays the interaction of the SENSEI’s enhanced architecture.
Chapter 2

Background

The purpose of this chapter is the explanation of the main contextual ideas around this project, in order to make the reader familiar with the topic of the thesis.

First off, Section 2.1 defines the Internet of Things, the root concept that sustains this project and its relevance. It basically states that human life will be surrounded by “living devices” permanently connected to a common network that will provide us lots of meaningful information.

Moreover, Section 2.2 is fully dedicated to the importance of these data streams and how can we exploit these changes in the current market, producing new value chains that lead to new business opportunities, and definitely, to an expanded and diversified market.

Lastly, the combination of these two previous ideas may give, as a result, in a revolution in society and urban life. Section 2.3 describes the concept of a Smart City, a new viewpoint that involves a full change in the current cities transforming the infrastructures, making efficient use of energy and resources.
2.1 Internet of Things

“Internet of Things” (IoT) is a concept proposed by Kevin Ashton in 1999 that refers to a network of uniquely identifiable and interconnected objects. These objects are devices, generally sensors, which are constantly gathering information of any kind to provide it as data sources.

These devices may produce a vast amount of information, measuring lots of values in different timestamps. This collection of data values might be seen as a continuous flow thereby creating a stream of data. Streams consist of information from different types (e.g. temperature, water pressure, humidity or air pollution. They always also contain self-describing meta-data).

In the future, it is expected to create networks of objects with millions of devices, each one producing a huge flow of data. The complexity of its processing makes requires machines and applications that may merge and analyse the data, reacting in a clever way taking the best decision.

Thus, IoT applications seem to be one of the key fields to research in the future, among the concept of Smart City (which is extensively described in Section 2.3).
2.2 New Value Chains

As far as we know, the Internet of Things area is not only changing the current industry, but also giving, as a result, in technological progress that may affect the human life in a highly positive way. Moreover, it turns out to be also a revolution in other areas such as business and marketing.

Catherine Mulligan pointed out a new form of value chain is being developed around this recent platform, built on top of mobile connectivity and computational capacity. She suggests that the combination of both concepts may take raw data from multiple sources (smart phones, sensors, social networks) and process it, resulting in a new kind of product. This outcome would be an information product with surplus value, which might generate new ways of business and employment.

Figure 2.2: From devices to information products, the new value chains in the market [37]

Another important factor that must be taken into account, as Catherine argues, is one of the unique properties that information has. In contrast to traditional and physical materials, raw data is something that can not be consumed during the production process. Its value lies not only in the exchange and use, but also in the reuse.
This property makes possible to resell an information product numerous times in the market or even to be aggregated with many other elements in order to create new products or extend the current ones. Therefore, a lot of new business opportunities are expected to play an important role in the market during the next years. An hypothetical situation that may demonstrate this fact is the next example:

“A person who arrives in an unknown town might be interested in buying some food. Then, he/she could use the smart phone, search for lunch places near his/her current geographic location and choose a road to reach the place and eat”.

In consequence, food retailers could be able to improve their business following these actions:

- Collecting such a flood of data using different sources and service providers.
- Analysing the data, finding information patterns (e.g. aggregating the GPS, the road chosen and user parameters such as age, position, education level...).
- Developing new resources and increasing the investment following such patterns, (e.g. building a restaurant in a specific place where the most of queries were made, offering certain food at a concrete price and with a specific decoration).

This clearly motivates the industry to build software marketplaces. The existence of such systems turn out to be quite relevant in the next years, since marketplaces have a high potential for the business of multiple organizations.
2.3 Smart Cities

Once we have understood the high business potential that the Internet of Things and data processing has, it is also important to remark on the great impact these ideas could have in the society. Long-term, *Smart cities* are probably the most important and ambitious consequence of IoT and this thesis topic.

Several studies, like the ones made by The Climate Group, support the idea that “The 21st century will be the *century of urbanisation*” [19]. In 2050, is expected to be more than 9 billion people around the world. Therefore, taking into account the last years and the progression expected, humans will live in larger cities whose budget and GDP could be comparable to small nations.

Accordingly, the current ways of living are challenged by this new trend, facing different but related problems such as sustainable growth, infrastructures, logistics, and specially healthy environments. Large cities usually provide a wide variety of opportunities and comfort, but these changes also could produce environmental and social issues. In consequence, it would be harder to meet the needs of people, since population will demand gradually higher standards of living.

![Figure 2.3: Potential applications and use cases in a future Smart City [20]](image-url)
The consequences and future implications that the “Smart City” concept will provoke in our societies and environments are unpredictable, but it is expected to modify human life in a much better and comfortable way. Table 2.1 represents a subset of the most common use cases people may find in a smart city context:

<table>
<thead>
<tr>
<th>Activity</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban Transports</td>
<td>Real-Time reaction to traffic bottlenecks</td>
</tr>
<tr>
<td></td>
<td>Alternative public transports based on demand</td>
</tr>
<tr>
<td>Urban Resources</td>
<td>Adjustment of light intensity depending on the climate</td>
</tr>
<tr>
<td></td>
<td>Collection of garbage when full trash cans are detected</td>
</tr>
<tr>
<td>Urban Transports</td>
<td>Discovery of places when needed (parking, gas stations, . . . )</td>
</tr>
<tr>
<td></td>
<td>Adaptation of temperature conditions before reaching them</td>
</tr>
<tr>
<td>Domestic Devices</td>
<td>Temperature and humidity (heaters, air conditioning)</td>
</tr>
<tr>
<td></td>
<td>Small automatic processes (start the dryer after the washer)</td>
</tr>
<tr>
<td></td>
<td>Inform how many food is left and which one will expire soon</td>
</tr>
<tr>
<td></td>
<td>Establish a shopping list based on the food kept on the fridge</td>
</tr>
<tr>
<td>Shopping</td>
<td>Suggestion of recipes based on your food and shopping list</td>
</tr>
<tr>
<td></td>
<td>Suggestion of discounts and offers in near shops</td>
</tr>
</tbody>
</table>

Table 2.1: Common use cases in a future Smart City

The question now is if we can economically afford these changes. The Climate Group suggests in their SMART2020 assessment [13] that making use of smart technologies in multiple areas such as transport, logistics, buildings and electricity grids might decrease the global emissions by 15% in 2020, which means around $900 billions a year in energy savings to the global industry.

However, they clearly state that it is necessary to encourage cities to think about a new urban model that becomes a *Clean Revolution*: a change in the human societies, motivated in part by technology, that enables us to adapt to the future.
Chapter 3

Problem Description

This chapter defines the main problem to address within the thesis, that is the Access Control to valuable information, and also which limitations and scope the thesis has.

3.1 Problem Statement

The problem this thesis faces is the solution to the technical issues and implications in the implementation of “Internet of Things Marketplace”. In the state of the art there has been a lot of work done regarding the future trends in our economy and society, as The Climate Group [19] pointed out, but also regarding the business models we could develop, as Catherine Mulligan did [37].

However, there is not much work focused on the association between such business models and the actual features a marketplace usually provides in other kinds of electronic marketplaces, such as Google Play or the App Store, specially regarding secure Access Control.
3.2 Access Control and AAA Architectures

“Authentication, Authorisation, and Accounting” (AAA) is a term that refers to the management of the access to specific resources through the use of strict policies. This access control is carried out also auditing resources usage, providing billing capabilities for services.

Although there are some architectures that are only concerned with authentication and authorisation (AA), the 3 processes are often combined. Moreover, they are considered fundamental for an effective network management and security.

It is important to define the 3 concepts that compound the AAA acronym.

Authentication

The first step to provide security in access control is the authentication step. This process is based on identifying each user with a unique set of credentials, before access is granted. For this purpose, the credentials that are often used consists in a valid user name and also a valid password, which usually follows a set of basic rules (e.g. minimum 6 alphanumerical characters with at least one number and one special character).

The AAA server compares the credentials with the rest of the user’s information previously stored in a database. If the credentials match, the user is granted access to the network. Otherwise, the authentication fails and network access is denied [28].

To summarize, authentication answers the question “Who or what are you?”

Authorisation

After a user has been authenticated successfully and logged into the system, he must gain permission for doing specific tasks. The authorisation process determines whether the user has the ability to perform actions by enforcing him to fulfill certain policies. These policies determine what types of services or resources a user is able to use.

In summary, the purpose of authorisation is to know “What are you allowed to do?”

Accounting

Finally, the last concept within AAA is accounting. This process measures the consumed resources as well as the activity a user has performed during the access. Different approaches may be used to measure this activity, such as the amount of
time spent on the system or the amount of data transmitted.

The accounting module works auditing the usage of resources accessed by users. Potential applications of these measures might include billing, trend analysis and resource utilization.

To sum up, accounting wants to know “What did you do?”

How AAA integrates into the Internet of Things area

An AAA module may fit into a whole IoT Engine, giving as a result secure access and accounting capabilities. As seen in Figure 3.1, the IoT Engine refers only to the data processing and storage, although some authors might consider the Engine as the whole IoT Hub, even including the Local IoT Index. These integration allows the transformation of the whole system into an actual marketplace:

![Diagram of a possible IoT architecture featuring AAA](image)

Figure 3.1: Diagram of a possible IoT architecture featuring AAA [22]
3.3 Limitations and Scope

Since the length of the thesis should not be greater than a semester, the following paragraphs explain the elements that are considered beyond scope because they are already implemented, such as the catalogue of streams, or because they are not planned to be performed.

**Data Streams Catalogue**

The implementation of the marketplaces will be based upon an existing version of a software tool. Concretely, the marketplace will be an extension of a tool developed in previous work, the IoT Framework. This tool was built by students of the Uppsala University, in the Project CS 2013 course, where both the author and supervisor of this thesis were involved.

Therefore, basic marketplace functionalities, such as indexing or cataloguing of streams, are out of scope since these objectives were achieved before. In addition, the data visualization of these streams was already performed, so the marketplace is intended to be as an extension of the current IoT framework supporting billing and financial capabilities in order to charge customers.

**Study of Business Models**

One of the final goals of an IoT marketplace is to sell valuable information to customers, that they then may use to improve their business opportunities. Therefore, it would be quite interesting to carry out a financial study regarding potential business models that might be used and applied for the marketplace.

Since neither the student nor the supervisor of this thesis have a strong background in economics, it is difficult to study and decide what business model could be the most appropriate for this case study.

**Advanced Interconnection with Other Parties**

The connection with multiple resource directories [33] simultaneously will be kept out of the scope of the project. In addition, the AAA module will not include advanced features such as the creation of several security contexts.
Chapter 4

Methods

This brief chapter explains which methods have been used in the study of AA protocols (carried out in Chapter 5), in order to implement the AA module.

Concretely, there are 7 fundamental factors such as if the protocol is an open standard, if it has been supported and used in the industry, or if it is a recent technology that has not suffered from security exploits.

4.1 Factors Studied in AA Protocols

Due to the lack of experience or background in authentication / authorisation protocols, a list of key factors was required to establish a high quality analysis of the current technologies in the state of the art. The next subsections explain which factors have been studied and what needs have been addressed.

As a result of this research process, the main alternatives found were OpenID Connect and SENSEI. A few protocols were found but discarded for not meeting the outlined criteria. These protocols are Kerberos, CAD, Radius, Diameter, ADFS or OATH.

Open Standard Initiatives

It is preferable an open technology for several reasons: it gives developers more freedom to use the protocol in different ways and use cases, it could also be extended if advanced requirements arise in the future with further work, and last but not least, it will not include extra license costs or royalties.
Single Sign On and Federated Identity

The protocols should support Single Sign-On [14] (SSO). The purpose of this idea is letting a user to authenticate only once, using the needed resources of the Service Provider as many times as necessary. There are two different kinds of SSO solutions: delegated and federated.

A delegated solution means that one site is simply outsourcing its authentication needs to another pre-selected site, like Facebook does with its private protocol: Facebook Connect. In contrast, a federated solution means that visitors to your site can use any account they have, as long as it is compatible. Therefore, Federated Identity Management [36] (FIM) is a much better approach of SSO, due to its higher compatibility. For instance, OpenID matches this criteria.

Security Attacks

If none, or only a few, security attacks have been reported, the robustness and reliability of the protocol will probably be higher (however, this criteria is difficult to measure properly because a protocol might suffer from several unknown vulnerabilities).

Recent Technology

It is important to find recent and modern technologies, in order to make a contribution with enough support in the future, but stable too. A quite recent protocol, not enough tested, will be discarded.

Commercial Sponsors and Support

Related to the previous point, the study takes into account the market share and adoption in the industry of the current stable protocols, as well as the future expectations in the new solutions (e.g. SAML was highly adopted, but OpenID Connect has more potential in the next years).

Degree of Development Comfort

Authentication and authorisation are known to be complex operations that may require non-trivial implementations and/or integrations. Thus, a high development required to use these protocols with current techniques and architectures will be quite important (e.g. a protocol that is not compatible with a RESTful implementation could be tricky and create issues in the development).
Support for Mobile Applications

Finally, although this factor was not considered before starting the study, it has been one of the key criteria. Some protocols were designed before the arrival of smart-phones in 2007, which provoke implementation issues in native applications.
Chapter 5

State of the Art

This chapter covers the study of the state of the art and the best current technologies in the market.

The first section describes what the *IoT Framework* is, the system that is going to be extended and will include the AA module.

Section 5.2 will establish a comparison between the 2 most important protocols for *user authentication and authorisation*: OpenID Connect (5.2.1), a new, promising standard with modern features, and SENSEI (5.2.2), a reference implementation developed and quite used in the industry. These technologies will be fully explained, including a SWOT Analysis for our scenario.

Finally, Section 5.3 will talk about which kind of mechanism has been developed for *API protection*. The steps followed by Amazon, one of the best designs studied, will be enumerated and discussed. Also, the differences between Amazon and the developed solution will be mentioned.
5.1 IoT Framework

As it was explained in Section 2.1, all of this hardware and the complexity they require would be too high to be monitored only by humans. Thus, systems need to be built to handle this information and facilitate the management of this data, either suggesting humans what is the best option to choose, or even doing it automatically. Such a system would also need to understand meta-data regarding the devices to be able to decide how useful the data given from a device is.

The physical limitations of these devices lead to different ways of communicating. A device that is powered all the time could be polled for the latest value or be allowed to push data to the system, while on the other hand a device that is battery-powered might need to turn itself off to conserve energy and thereby may not be available for polling. This problem could be solved in two alternative ways:

1. The device pushes the latest value to the system every time it wakes up.

2. Using a mirror proxy [39], which could be polled by the system whenever it wants a new value and the device would then just need to update the value in the mirror proxy.

The purpose defined for this system was the ability to handle around 50 billion devices and index the data streams they produce as a large catalogue of live information. This streams could be created by organizations that provide public access to their data or, however, they could be also individuals who are willing to manage their private sensors (e.g. the temperature at home).

In addition, other advanced features include ideas such as the creation of rich, complex aggregations of streams (mixing values from different streams of the same type), the use of triggers executed when a specific condition is met on these streams and forecasting capabilities using statistical predictions. Such data, gathered either by polling values from devices or pushing values to the system, should have a visual overview in order to let the user understand easily the meaning of the data processed.
5.2 AAA Protocols

5.2.1 Alternative 1: OpenID Connect

OpenID Connect [29] is a new open protocol whose purpose is to establish and provide an Identity layer on top of the current TCP/IP Architectural Model [41] in which the Internet is based upon. The current version, released in February 2014 is the third version of the protocol, and it allows developers to verify the user’s identity, using authentication across websites or apps. Thus, it is not needed to manage and store passwords or any other credential data locally.

![TCP/IP architecture with the identity layer on top](image)

The first version of OpenID was released around 2005. This version started to motivate developers some interest in the industry about the potential of the concept. However, it never had a noticeable impact in the market.

The second version, released in 2007 as OpenID 2.0, was a highly improved version of the protocol that worked well regarding security, but was seen as a difficult-to-use technology due to its lack of interoperability with other protocols and tools. It also suffered from several design issues and limitations, specially the lack of support for native mobile applications, a market segment that strongly increased around those years.

One of the fundamental concerns that the Working Group members had in mind was the creation of a new version, dramatically easier for developers to use and integrate with existing systems, compared to any preceding Identity protocol. In consequence,
the technology would reach a much higher market impact than in previous versions.

To fulfill this purpose, OpenID Connect uses standard JSON Web Token [18] (JWT), a special set of fields written in JSON [6] format that are used when security signatures are required. Among this feature, the protocol follows a total RESTful [10] approach, which causes a much better interoperability with external apps.

Figure 5.2 explains the OpenID internals, and displays how the authentication mechanism worked in the first and second versions of the protocol.

Figure 5.2: OpenID authentication in prior versions explained by Nat Sakimura [30]

The reader may appreciate that an end-user called “Alice” is willing to use a certain service provider (SP) such as “Eve”. The interactions between both elements could be explained as follows:

1. First off, Alice wants to use Eve’s service, but Eve needs some kind of certificate from Alice to prove she is who claims to be.

2. Alice asks Google, which is one of the OpenID Identity Providers (IP, or IdP), for a proof of identity.

3. Then, the IP accepts the request and send to Alice the Identity certificate without exposing any private or confidential data.

4. Finally, Eve receives the certificate from Alice, trusting in the IP seal.
OAuth 2.0

However, since late 2006, certain competitors emerged using different authentication approaches. One of the most popular was OAuth, a well known authorisation protocol that represents a de-facto standard in the industry.

OAuth is an open standard protocol used for authorisation purposes in web, desktop and mobile applications. The development started in November 2006, when engineers from Twitter and Magnolia were interested in their OpenID implementations, founding several discrepancies in its internal design.

Although the name may cause confusion due to the term “Auth”, the main goal of the protocol is offering an easy-to-use authorisation mechanism. Nonetheless, it has been used also for authentication, an activity that is not quite advised since it actually performs a pseudo-authentication, which basically grants external applications access to certain private resources owned by the end-user. If the access is granted, then the user is trusted and the application retrieves the unique identifier for establishing the profile identity. However, in the case of OpenID, the app just trusts in the official OpenID Identity Provider without giving the app access to the private data.

Figure 5.3 shows the mentioned pseudo-authentication mechanism, common in the market but not secure enough if used alone [8].

Figure 5.3: Pseudo-authentication using OAuth explained also by Nat Sakimura [30]
In contrast to OpenID, since the main focus of OAuth is authorisation, the mechanism used to prove that an end-user should be trusted is slightly different. Unlike OpenID, which delivers a certificate from the IdP assuring the user has been verified successfully, OAuth delivers a key to the Service Provider (SP) that grants access to private identity data in the OAuth server. This may provoke dangerous results in case the SP has malicious intentions. The protocol interactions might be explained briefly as follows:

1–2. Same steps as in Figure 5.2 (OpenID authentication), but requesting a key instead. The IP has been replaced by an OAuth server (e.g. Twitter).

3. Then, the OAuth server accepts the request and sends the key to Alice. Since the OAuth server delivers a key instead of an identity certificate, if the mentioned key is used by an attacker, Alice’s private resources will be exposed.

4. Finally, Eve (SP) receives the key from Alice. If Eve uses that key, then she will trust and grant Alice access to Eve’s services. However, Eve would be able also to get some private information about Alice. This fact proofs that OAuth is not a full authentication protocol and it could be dangerous in a worst-case scenario.

Despite this design pitfall if used without any other protocol, OAuth 2.0 is probably one of the most widely used protocols nowadays on the Internet. It is highly adopted by the industry, supported by organizations such as Twitter, Google, Facebook, Microsoft, LinkedIn, Foursquare, GitHub, or Salesforce, among others, but also by individual developers. Thus, due to the great commercial adoption OAuth had from the beginning, it was chosen as a base protocol in the last OpenID Connect’s specification.

Finally, there already exist several implementations of this protocol written for servers in Java, JavaScript (Node.js), .NET, Python, Ruby, PHP and Erlang. Also there are implementations for the client-side as well, in the same languages, adding native support in iOS and Android.
Identity + OAuth = OpenID Connect

OpenID Connect is the third generation of the standard, whose internal architecture was re-factored and redesigned to make it way simpler. In this version, the OpenID Working Group decided to build the protocol on top of OAuth2.0, combining both identity and authentication capabilities with the authorisation layer needed in previous versions.

In consequence, applications with OpenID Connect might request and receive information about authenticated sessions and end-users in an authorized context. It delegates communication security to the TLS [9] protocol. Also, it supports optional features such as encryption of identity data, discovery of alternative OpenID providers and session management. However, it still lacks of accounting, the third element in the AAA issue.

Moreover, OpenID Connect is expected to be highly adopted in the future market, due to the strong support given by a powerful group of international corporations such as AOL, Deutsche Telekom, Facebook, Google, Microsoft, Orange, PayPal, Salesforce, Verizon or Yahoo!. Also, several academic organizations have worked on it.

The next diagram shows the improved interactions in Connect. Despite that it looks quite similar to OpenID (Figure 5.2), now the Service Provider accepts the user’s identity (ID Token) and an additional key (access token). Nonetheless, this key will be only useful for “opening a sort-of locker, not the whole room”.

![Diagram of OpenID Connect authentication using OAuth as a base protocol](image)

Figure 5.4: OpenID Connect authentication using OAuth as a base protocol [30]
 CLAIMS AGGREGATION AND DISTRIBUTED CLAIMS

As Nat Sakimura points out in his article [30]: “The key handed to Eve here is only good for the locker prepared for Eve. It contains Alice’s basic information. You can also add any other things, for example a certificate of qualification or even keys to other lockers”.

This approach allows external sources or services to get only limited access to basic end-user information, without exposing potentially private data. Thus, the protocol provides the two following features:

- OpenID Connect may put together at the same place several certificates, technically denominated as “claims”. This feature is called Claims aggregation.

- The protocol may create keys to other lockers from where a SP can get additional information. This is called Distributed Claims.

As a result, Sakimura states also that “OpenID Connect is a distributed identity framework that not only allows safe authentication but also make it possible for the sites to move the distributed data in the internet and provide better services” [30].

Figure 5.5: OpenID Connect’s claims aggregation and distributed claims [30]
SWOT Analysis

SWOT [38] is the acronym of “Strengths, Weaknesses, Opportunities and Threats”, a technique used in business contexts to analyse a product or an organization, according to certain factors. The goal is showing the benefits and drawbacks the studied entity offers, describing also if they are motivated by internal or external causes.

The next diagram represents a SWOT Analysis made to study the appropriateness of the OpenID Connect protocol in this thesis:

![SWOT Analysis Diagram](image)

Figure 5.6: SWOT analysis for the OpenID Connect protocol
5.2.2 Alternative 2: The SENSEI Project

SENSEI [1] is an European project initiated in January 2008 and finished on 31st December 2010. It was built with the philosophy of being “a reference architecture to link and integrate the physical and the digital world in the network of the future”.

The project creates an open, business driven architecture that addresses several problems such as the management of multiple heterogeneous “Wireless Sensor and Actuator Networks” devices (WS&AN) globally distributed, their integration into a common framework and the scalability issues to avoid when these devices form a very large number.

It provides network and information management services, enabling other services or applications to retrieve reliable context information, but also to interact with the physical environment through the managed devices.

Figure 5.7: The SENSEI philosophy consists of the abstraction of physical devices [1]
The Working Group was composed by 19 different partners from 11 European countries. Organizations that were involved in the project were mostly corporations such as Ericsson (Sweden), SAP (Germany), NEC (Great Britain), Telefonica (Spain) or Thales (France).

In spite of being an international project supported by important institutions, SENSEI does not own a high market share regarding AAA. The last technical paper shown by the official website [1] was written in 2011. The most recent article found that mentioned SENSEI was written by J. Song, and A. Kunz, M. Schmidt and P. Szczytowski [35] in 2013. In it, however, only the work done on the resource definition and representation was used, completely ignoring the AAA architectures.

**Architecture**

Figure 5.8 displays from a high-level perspective how the protocol works with the Single Sign-On server as a core component and how the end-user interacts with the rest of AAA modules.

![Lightweight SENSEI architecture with full AAA support](image-url)
Unlike OpenID Connect, SENSEI supports and includes mechanisms for accounting and billing. The important concept that remains in SENSEI is that authentication is a cross-resource step, while authorisation depends on the policies and rights the user owns depending on the resource and its provider. In addition, two different architectural models were defined in order to satisfy different, potential use cases. The next diagram shows the *Baseline* model. However, the *Enhanced* architecture is beyond the scope of this project (Appendix B).

**Message Flows**

![Figure 5.9: Baseline version of the SENSEI architecture [4]](image-url)
Figure 5.9 assumes there is an end-user (Alice) and a Resource Provider (Bob) and both have registered with different SENSEI Framework Providers, each running their own AAA services and token formats. The interactions between them are described in more detail as follows:

1. **RAI Request.** The end-user sends a request to the Resource Provider (RP) that needs to be authorised.

2. **Request Token Issuing.** Bob may use a token from Alice to authenticate her and authorise the access. However, if this is the first time Alice requests access in the system, Bob will request his authentication Token Service (ATS) a new token. In consequence, Alice will be redirected via HTTP towards this service.

3. **Request Token Issuing (II).** As Alice and Bob are using different SENSEI Framework Providers, Bob’s ATS requests the token from Alice’s ATS. Then, the last one generates the token for getting access, causing again another HTTP redirect.

4. **Authenticate Alice.** If this is the first time Alice has contacted the system during this session, she has to perform an authentication step, which means she would be redirected to her ATS. Otherwise, the system can directly look up Alice’s attributes and issue the appropriate token. Then, the ATS generates a digital coin (a token value that represents a particular resource has been used by a user) and passes it to the accounting service.

5. **Token Validation.** Since both ATS belong to different SENSEI domains, the format used might be different. For that reason, the token generated is returned back to the ATS that belongs to Bob’s domain. Then, the token is translated into Bob’s ATS format and validated.

6. **Request Token Retrieval.** If the token is valid, it is returned to Bob (the Resource Provider).

7. **Authorisation Request.** At this point, Bob already knows the end-user identity and permissions. Bob passes the token’s attributes to its Policy Decision Point (PDP), with the RAI request and any associated context. Then, the PDP service evaluates its policies, stores the digital coin in the local database in order to be synchronized with Alice, and finally, returns a decision as to whether the request should be authorised.

8. **RAI response.** The response to the original end-user’s request is returned.

After Alice has accessed Bob’s resources, Bob sends the list of stored coins to Alice. Then, Alice’s accounting service validates the coins by checking the HMAC values.
and comparing the results with its database. Alice’s and Bob’s accounting services remove the paid coins from their databases.

SAML

Security Assertion Mark-up Language, commonly referred as SAML, is a XML-dialect, mark-up language created in 2001 by the OASIS [23] Security Services Technical Committee.

It is an open standard widely used in the market, specially by enterprises, as a data format for exchanging authentication and authorisation data between parties. These parties are usually two elements, which play an essential role in the protocol: the Identity Provider (IP) and the Service Provider (SP). As the reader may appreciate, both SAML and OpenID uses almost the same terminology.

In 2011, a failure in the implementation of the XML Signature Wrapping was discovered and reported by Somorovsky et al. [34]. The XML Signature defines the syntax and processing rules for creating, representing and verifying XML-based digital signatures. His research found several buggy implementations from different service providers, including Salesforce and IBM XS40.

The vulnerability is caused by the elements contained in the XML, which may be copied from external documents among a valid signature. Although useful in certain cases, specially with business processes in which documents are processed and transformed, it also may turn out to be a dangerous property [21]. Apparently, those service providers managed to fix the found pitfalls in their implementations. Nonetheless, it is a key security aspect to take into account when using the protocol.

Moreover, complexity is one of the key limitations with SAML. This technology is clearly much more compatible with SOAP than with REST (as it was mentioned earlier, SAML is an XML-dialect). A possible work-around that may fix this issue is to wrap the tokens into the headers of the HTTP requests exchanged, in order to simplify the flow. This way, the result is an architecture more RESTful. Despite this, it will not follow the REST style and conventions at 100%.

Regardless of this issue, SAML has been so far the de-facto standard protocol in enterprise-oriented projects. There are libraries available for Java, C++, PHP, Ruby and Python.
SWOT Analysis

As we did before with OpenID Connect in Section 5.2.1, the next diagram represents a SWOT Analysis [38] showing the benefits and drawbacks of the SENSEI architecture in the current market.

Figure 5.10: SWOT analysis for SENSEI, based on the SAML protocol
5.2.3 Final Comparison

The next table summarizes the differences shown by SENSEI and OpenID Connect, the two potential alternatives for the implementation of the “AAA module”.

<table>
<thead>
<tr>
<th>Factors studied</th>
<th>SENSEI</th>
<th>OpenID Connect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finished specifications</td>
<td>December 31st, 2010</td>
<td>February 26th, 2014</td>
</tr>
<tr>
<td>Communication protocol</td>
<td>SOAP</td>
<td>REST</td>
</tr>
<tr>
<td>Data Format</td>
<td>SAML</td>
<td>JSON</td>
</tr>
<tr>
<td>Sponsors</td>
<td>Ericsson, NEC, SAS, Telefonica</td>
<td>Google, Microsoft, Facebook, Yahoo, Orange, Verizon</td>
</tr>
<tr>
<td>Support for native mobile apps</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support for accounting</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Potential market adoption</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Table 5.1: Comparison between the SENSEI project and OpenID Connect

As the previous table shows, the SENSEI project is older than OpenID Connect, which seems to be the most appropriate alternative in the future. Although it might be quite recent, it has been tested for the last two years in enterprise environments, according to Nat Sakimura. Currently, SENSEI does not have a great adoption in the industry and this trend is not expected to increase either. The same issue appears when comparing the communication protocols. SAML was released in 2001, which is a format not as well-known, supported and used as JSON nowadays.

Another important reason to consider in the selection, is the fact that SENSEI is mainly a SOAP-based solution. It is possible to use a sort of RESTful approach by wrapping the tokens in the headers of each HTTP request. However, this does not respect or follow the REST principles at all.

Therefore, the technology selected for the implementation of the AA module is OpenID Connect.
5.3 API Protection Designs

After being studied different approaches to protect the API of the IoT Framework, the Amazon design has been selected as alternative to our solution. It works with two security keys: an ID that is public and represents an authenticated user and a private Secret Key. The next list, among Figure 5.11, summarizes step by step how the process works:

1. The client builds a request to Amazon Web Services (AWS). The client generates a unique signature combining the secret key (kept locally) and the parameters of the request.

2. The client sends the request to Amazon. Include in the request the user’s public ID and the signature calculated in step 1.

3. Amazon uses the user’s public ID to look up the secret access key and calculates a new signature combining such a key and the parameters included in the request. The algorithm used to generate the signature is the same on both sides (client and server).

4. If the signature generated by Amazon matches the one the user sent in the request, the request is considered authentic. However, if the comparison fails, the request is discarded and Amazon returns an error response.

![Figure 5.11: Architecture of the IoT Framework and its modules](image)
In contrast to Amazon, the IoT Framework will reuse the access token provided by the OpenID protocol (issued by the IdP) as the secret key. However, sending a signature instead of the access token seems a smarter approach that avoids possible Man-in-the-middle attacks [24]. For that reason, the signature mechanism is considered in the design of the module but will be implemented in the future, out of this project.
Chapter 6
Implementation of the AA Module

This chapter contains five sections that fully explain how the solution was achieved.

The first section describes how the Architecture of the IoT Framework was designed previously within the context of the Project CS course at the university. Also, the section explains how the new AA module has been integrated, which elements are part of the framework and how they communicate with the module.

Section 6.2 defines the Trust Model utilized within the AA module, so as to confirm if a request should be trusted and the access granted.

Section 6.3 details the mechanisms the AA module uses to fetch the user’s identity from the identity provider (IdP). Due to the existence of quite a few different scenarios, 3 different execution flows have been developed in order to cover all the use cases.

The fourth section is about the Protection of the API the Engine provides, describing what information is needed to authenticate a user and authorise his requests.

Lastly, Section 6.5 shows a table with the Access Policies used by the AA module to verify if an authenticated user has enough permissions to get access to the API resources.
6.1 IoT Framework’s Architecture Design

The next list enumerates the relevant elements and modules that are part of the IoT Framework’s Architecture. The list also describes the role and purpose of each module:

**Engine.** The core of the system. It contains the logic and provides access to certain resources (users, streams and devices) through an API written in Erlang.

**Frontend.** A server written in Ruby on Rails that works as a web application, and shows the User Interface of the system.

**ElasticSearch.** A non-relational database the engine uses for saving and storing the vast amount of data the system manages.

**Pub/Sub System.** A system that publishes data values when arrive for a certain resource. Then, its purpose is to distribute this new data to all the clients who were subscribed to that resource.

**Polling System.** A module that periodically fetches data from multiple external resources and stores it into ElasticSearch or pushes it to the Pub/Sub system.

**AA Module.** The new developed module in this thesis to provide secure AA for users and API protection.

Figure 6.1 shows how each element works within the architecture and communicates with other elements. It also highlights how the new AA Module has been integrated:

![Figure 6.1: Architecture of the IoT Framework and its modules](image-url)
6.2 Trust Model

The *Trust Model* of the AA module used to protect the API is based upon two unique, random security keys generated by the IdP:

**Access Token.** A temporary key, valid for just one hour. The access token is sent by the user on the headers of every request made against the API. If the token is valid, the authentication process calls the authorisation process, which will determine if the user may or may not get access to the API resources.

**Refresh Token.** This key is permanent and never expire. The refresh token is sent to the IdP when the access token has expired in order to issue a new access token valid for the next hour. In addition, when the user decides not to use the IoT Framework any more, he may revoke the refresh token and his private identity will not be reachable again by the platform.

There is an exception in the Trust model. There are certain users who do not use OpenID or simply prefer to create an account on the IoT Framework using their own credentials (username and password). Those users will receive a pair of access and refresh tokens, but these ones will be issued by the AA module, because there is not any IdP to fetch information from.

This exception is slightly less secure than the OpenID verification, due to there is not any identity provider used as a third party. Nonetheless, the tokens are still secure enough for being used in production due to the strength of the keys if they are sent through TLS communications.

Any request without an access token, or an invalid token included in the headers, will be discarded and not handled by the engine, returning an error back to the API client.
6.3 User Creation and Identity Management

In this section, we are going to describe the execution flow of the IoT Framework when a new user is created on the system. The framework supports two mechanisms or flows to manage the creation of users who login using an existing OpenID Connect account and one flow for those users who do not, so they will be required to introduce their custom username and password.

It is important to remark that Google has been selected as Identity Provider (IdP since now). Thus, the AA module will make a call to the IdP every time a user needs to be authenticated. If the process returns a successful result, the IdP will return the user’s identity in the response, which might be used by the module.

6.3.1 Server-Based Flow

The first flow, denominated "Server-Based Flow", will be mostly used by other machines or external modules that might need to request access to the API resources of the framework. Although this flow may be used also by users through the use of other API clients (mostly command-line-based), they are more likely to use the Frontend application, which makes use of the "Client-Based Flow", further explained in Section 6.3.2.

Figure 6.2 displays as a business process how the module handles external requests:
The flow may be summarized as follows: the module calls the IdP to generate a unique URL the user will use to login. If it is successful, the user will be redirected to another URL while the module again requests the IdP to issue the access and refresh tokens. Finally, the user will receive the tokens in a JSON document.
6.3.2 Client-Based Flow

The second execution flow developed is the “Client-Based Flow” and it is purely utilized by end-users. The flow has been developed due to the existence of an official Google API library [12] that generates the security tokens avoiding redirections.

Below, Figure 6.3 represents how this flow has been implemented:

![Figure 6.3: User creation using the client-based flow](image-url)

Figure 6.3: User creation using the client-based flow
On the client-side, the Google library generates a one-time-valid *Code* and a temporary *State* variable. Both values are sent to the Frontend server. If the state sent matches the server’s state, then the Code is considered trustful. Then, the Frontend exchanges the Code and State to a new *access token*, and also fetches the user’s identity.

Since the Frontend owns a local database to manage sessions, it stores the user, if new, both locally and on remote (sending the information to the engine). Finally, the user signs in automatically and the Frontend redirects him/her to the Streams page.

### 6.3.3 Non-OpenID Flow

The last of the Identity flows corresponds to non-OpenID users. It was created for those users who do not own any OpenID identity. The AA module just creates its own random access and refresh tokens, obviously without the use of any IdP. The tokens are issued and added in the JSON document received from either the Frontend or any other API client.

![Process 1.c – Creation of User with Non-OpenID Flow (using our own tokens)](image)

Figure 6.4: Creation of non-OpenID users as a fallback mechanism
6.4 API Protection

The API protection can be achieved through the authentication and authorisation processes performed sequentially. Nonetheless, when a request includes an expired access token, the API will return an error and the API client will need to renew the token. For this scenario, a third execution flow was designed, denominated “Token Renewal”.

6.4.1 Authentication of a Request

First off, the AA module needs to be registered in the Google Developer’s Service [11]. That way, the IdP will grant a unique \textit{ClientID} and \textit{ClientSecret}, in order to make calls against the OpenID API. Afterwards, every time a request is made against the IoT Framework API, the AA module will call the IdP to verify the validity of the received access token, included in the request headers.

On the one hand, if the IdP response contains a field called \textit{Audience} and it matches the ClientID, then the token is considered valid and the authentication is successful. On the other hand, if the IdP response is an error or the Audience field does not exist, the AA module will verify if the token was issued by itself, as a fallback mechanism. If the token exists in the database and is not expired yet, the authentication is also successful. Otherwise, the authentication process fails.
### 6.4.2 Authorisation of a Request

The second step, the “authorisation process”, is called just after the authentication is done, and only when the result has been successful. It essentially retrieves the information of the request and checks if the user has “administrator privileges”, or if instead, the request matches the Access Policies established in the module’s configuration. A table with the selected policies may be found in Section 6.5.

If the authorisation is successful, the request is eventually handled and processed by the Engine. Otherwise, the API returns an error. Below, the next diagram displays and summarizes how the process works:

---

**Figure 6.5: Step 1: Authenticate the user who requested a resource**

<table>
<thead>
<tr>
<th>Process 2.a – Authentication of Requests against the API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>Request an API Resource Sending Access Token</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Figure 6.6: Step 2: Authorise and grant access if the request matches certain policies
6.4.3 Access Token Renewal

Finally, the last flow consists in receiving a special request that includes the refresh token and the User ID. If the token has been issued by the IdP, the AA module makes an API call to issue a new access token, valid again for an hour. Else, the module itself will generate a new token.

In both cases, the new access token will replace the old version in the ElasticSearch database and will be returned to the API client. Figure 6.7 displays the process:

![Access Token Renewal Flow Diagram](image.png)

Figure 6.7: Access token renewal flow
6.5 Authorisation Policies

This section explains the access policies the AA module follows in order to authorise a request. These policies depend on the required resource, the method used (GET, POST, PUT or DELETE) and the user’s id.

Every time a request is sent to the API, the AA module checks these policies. If the request does not match any policy, then the authorisation process failed and the API returns an error (the access is forbidden). Else, the system will grant the access and the requested resource will be available.

We need to note that at this step, the user is assumed to be successfully authenticated, because otherwise, he would have seen an error during the authentication process.

As we can see in Table 6.1, there are 5 different policies that determine what a user can do:

<table>
<thead>
<tr>
<th>Policies</th>
<th>Description of what a user can do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policy 1</td>
<td>A user can make any kind of manipulation of his/her data</td>
</tr>
<tr>
<td>Policy 2</td>
<td>A user can <em>not</em> manipulate data that belongs to a private user</td>
</tr>
<tr>
<td>Policy 3</td>
<td>A user can get personal data or streams / virtual streams from public users</td>
</tr>
<tr>
<td>Policy 4</td>
<td>A user can update the ranking of another user’s stream</td>
</tr>
<tr>
<td>Policy 5</td>
<td>A user can make GET /users or POST /users, /streams and /vstreams</td>
</tr>
<tr>
<td>Rule</td>
<td>Anything else is forbidden</td>
</tr>
</tbody>
</table>

Table 6.1: Access policies used by the AA Module to grant access to the API
Chapter 7

Evaluation

In this chapter, a list of potential attacks have been compiled. This list tries to test the security and reliability of the system using the AA module.

The techniques are organized in certain categories. Section 7.1 enumerates and defines four authentication attacks, while Section 7.2 is focused on the authorisation of the users.

Finally, Section 7.3 contains some common, uncategorised attacks.
Chapter 7. Evaluation

7.1 Authentication Attacks

This section will discuss the attacks used to circumvent or exploit the authentication process of the system. Each attack will include a technical explanation of what the attack is and how likely it is to happen.

Brute Force

“A Brute Force attack is an automated process of trial and error used to guess a person’s user name, password, credit card number or cryptographic key” [7].

Since the IoT Framework uses OpenID as an authentication mechanism, it is not concerned with Brute Force attacks because the login operation is delegated in the Identity Provider.

An attacker would need to guess a correct, not expired access token. But also, the token needs to be specifically issued for the IoT Framework. The reason is that there is still another verification step that takes the response given by the IdP. This response must contain an “Audience” field that must match the ClientID, a particular ID the IdP assigned to the Framework.

Therefore, it turns out to be extremely unlikely to guess a user’s token by Brute Force.

Insufficient Authentication

“Insufficient authentication occurs when a web site permits an attacker to access sensitive content or functionality without having to properly authenticate” [7].

The tests revealed all the API resources require a valid access token, returning an error message in JSON format if the request was not properly authenticated. The access token might be either generated by the IdP or by our AA module (the fallback mechanism developed to support non-OpenID users).

Weak Password Recovery Validation

“Weak Password Recovery Validation is when a web site permits an attacker to illegally obtain, change or recover another user’s password” [7].

Despite being technically impossible for an end user to change the access token manually, since this token has been issued by the IdP, it is possible to use the refresh token to issue a new temporary access token. However, the attacker would need to guess the refresh token, which is hardly ever used in the communications between...
The same principle applies for non-OpenID users. Although they make use of a username and password to create their accounts, they receive a pair of access and refresh tokens, created by the AA module without an external IdP. Then, these two tokens are used in the requests working as a normal OpenID, so that the system uses a consistent authentication flow.

Instead, the attacker would need to use another strategy to gain administrator rights in the system, and then manipulate illegally the data.

**Man in the Middle**

“A man-in-the-middle attack can succeed only when the attacker can impersonate one of the endpoints or parties within a communication. It might be considered an attack on mutual authentication, or lack thereof” [24].

This is the only test that failed during the evaluation. A solution might be achieved by combining the use of signatures (as the Amazon design does, already mentioned in Section 5.3) and communications based on the TLS protocol, which can perform authentication in one or both parties using a mutually trusted certification authority.

![Diagram of a Man-in-the-Middle attack](image)

*Figure 7.1: How a “Man-in-the-Middle” attack works between two end users [24]*
7.2 Authorisation Attacks

The authorisation section covers attacks that target a method of determining if a user, service or application has the necessary permissions to perform a requested action, or if instead, it only allow certain users to access specific content.

Credential/Session Prediction

“Credential/Session Prediction is a method of hijacking or impersonating a web site use” [7].

As it was aforementioned in the Brute Force attack (Section 7.1), it is very unlikely to guess an access token due to its length. However, the difficulty is even higher considering that the validity of the access token lasts only one hour.

Insufficient Authorisation

“Insufficient Authorization is when a web site permits access to sensitive content or functionality that should require increased access control restrictions” [7].

The design perspective that has been used in the authorisation process consists of six quite specific Access Policies, which are treated as exceptions. The rule is to always forbid the access to the API resources, and the tests done, included in the repository [25], have passed successfully.

Insufficient Session Expiration

“Insufficient Session Expiration is when a web site permits an attacker to reuse old session credentials or session IDs for authorization” [7].

When the IoT Framework receives a request, the token is sent to the IdP. Then the IdP decides if the token is expired checking if there are more than 3600 seconds between the current timestamp and the one when the token was issued. The token would also be invalid if it was revoked by the user within that period. Then, the IdP returns the verification to the IoT Framework, so that the request would be handled only if the IdP returned true. Thus, it is impossible to reuse old session credentials.

Session Fixation

“Session Fixation is an attack technique that forces a user’s session ID to an explicit value” [7].
There is no session management in the IoT Framework. The engine carries out pure processing tasks and offers a RESTful API, storing the outcomes, or the sanitized inputs, in the database. However, the Frontend server currently supports a legacy feature that consists of a basic user and session management. It was implemented for prior prototypes using the capabilities the framework (Ruby on Rails) provides. Since this feature is useless now, it will be removed.

As a result, the Frontend will just store locally the access tokens in order to use them while they are valid. These tokens might only be generated by the IdP, which is the only trusted source.

### 7.3 Other Common Attacks

Moreover, there are other usual methods that should be included in the list. Despite not being strongly related to the AA issue as the previous ones, they are common attacks to take into account, such as the SQL Injection and the Cross-site Scripting.

**SQL Injection**

SQL Injection is an attack technique used to exploit web sites typing SQL commands into text fields. These commands, if they are not treated safely, may produce fatal changes in the database (e.g. to delete the whole content) or just steal information.

The architecture of the IoT Framework was designed with separation of concerns, using one server for the Engine (with its AA module) and a web server that runs with Ruby on Rails whose purpose is to make API calls and render the data. With this design, the Backend trusts the input from the client-side since he expects to receive an access token on every request.

In consequence, the Frontend needs to store the access tokens in a local database. However the input fields are checked and sanitized before any processing, denying the request when it is not valid. As a result, no injections have been successfully made.

**Cross-site Request Forgery**

“Abbreviated as CSRF, is a type of malicious attack of a website whereby unauthorized commands are transmitted from a user that the website trusts” [26].

There are no tests for this technique. The web server has been developed using Ruby on Rails, a web framework that already includes security mechanisms to prevent
CSRF attacks through the use of particular CSRF tokens, and no recent attacks have been reported.

### 7.4 Evaluation Table

The next table summarizes the security attacks considered in the Evaluation process and the results found for each one:

<table>
<thead>
<tr>
<th>Category</th>
<th>Attack name</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Brute Force</td>
<td>Extremely unlikely to guess a user’s token, valid for the IoT Framework</td>
</tr>
<tr>
<td></td>
<td>Insufficient Authentication</td>
<td>Authentication process performed on every request, regardless of sessions</td>
</tr>
<tr>
<td></td>
<td>Password Recovery Validation</td>
<td>Only possible guessing the refresh token (not used very often)</td>
</tr>
<tr>
<td></td>
<td>Man in the Middle</td>
<td>Prevented using SSL and the signatures mechanism</td>
</tr>
<tr>
<td>Authorisation</td>
<td>Credential/Session Prediction</td>
<td>Very unlikely to guess a token before it expires</td>
</tr>
<tr>
<td></td>
<td>Insufficient Authorisation</td>
<td>The access policies are exception, the rule is to forbid access</td>
</tr>
<tr>
<td></td>
<td>Insufficient Session Expiration</td>
<td>Impossible to reuse expired tokens, whose validity usually lasts only for an hour</td>
</tr>
<tr>
<td></td>
<td>Session Fixation</td>
<td>No sessions on the IoT Framework Engine</td>
</tr>
<tr>
<td>Common</td>
<td>SQL Injection</td>
<td>Frontend checks all the input fields and calls the Backend API</td>
</tr>
<tr>
<td></td>
<td>Cross-Site Request Forgery</td>
<td>Frontend implement countermeasures through the use of CSRF tokens</td>
</tr>
</tbody>
</table>

Table 7.1: Prevention for security attacks and exploits
Chapter 8

Conclusions

The present chapter defines the conclusions reached during the development of the thesis.

OpenID Connect has proved to be the proper technology regarding authentication and authorisation. The protocol was easy to implement and the results of the evaluation were highly positive. In addition, the support given by relevant companies in the industry ensure OpenID Connect will become an open, highly used, web standard. Likewise, as we saw in Chapter 5, OpenID Connect does not support accounting capabilities. Accounting turns out to be the key factor that will completely transform current Resource Directories into actual data marketplaces, where information may be traded and sold. However, the HTTPA protocol (Section 9.1) might be a potential solution if combined with OpenID.

One of the most relevant consequences of IoT is the so-called Smart City concept, already mentioned in Section 2.3. With such a technological improvement, future cities will use better control and monitoring systems that will use more efficiently the resources, wasting less energy and saving large amounts of money per year. Due to such advances and the economic benefits expected from this revolution, it has been globally accepted that smart cities will change the current society, turning cities into a sustainable environment that will also make life easier, by taking advantage of the technological services provided around.

Therefore, fields like Internet of Things are vital to obtain these changes. Nonetheless, such changes would need improvements in other aspects such as security and data protection in marketplaces, access control for the raw data gathered by the sensors, and the communication protocols used between devices in order to increase the speed and the security of the messages.

Finally, this thesis has merged the security provided by OpenID Connect and the
common conventions regarding security in APIs to give as an outcome a reliable system where relevant information is highly protected, but also quite easy to be accessed.
Chapter 9

Future Work

The last chapter of this report contains three main topics considered as priority and where future contributions are expected.

The first issue consists in the lack of accounting features in current web technologies. HTTPA might be the first protocol to support such features, becoming a web standard solution that fixes this problem with the current public hypermedia model.

Section 9.2 goes in depth about how tomorrow’s software might use the advantages of Big Data and Internet of Things, managing a huge data set gathered from the users without breaking their privacy.

Finally, Section 9.3 focuses on Mobile Connect, an extension of OpenID Connect that potentially fixes quite a few issues that authentication and Identity management currently suffer in smart-phone and tablet applications.

9.1 HTTPA: Accounting supported by HTTP

As we have studied in the thesis, access control systems are suitable in managing access to resources on the web. Usually, they are able to identify when a user requesting a certain resource is an attacker. Hence, they are often effective controlling the access to valuable information.

However, they are ineffective in preventing information leakages, as it is very easy to copy or infer information on the web. When end users publish relevant data, currently there is not a proper mechanism to avoid third parties to use that information for different purposes. Since the nature of the web is strongly driven by layers of protocols, there is not any standard that automatically deletes data after a certain amount of time, or when the data has just been accessed and reused (although there
are standalone tools for such a purpose). For this reason, Oshani Seneviratne is working under the supervision of Tim Berners-Lee in a new web protocol called HTTPA [32].

HTTPA stands for “HTTP with Accounting”. The main idea in which the protocol relies on is, as Seneviratne said in his paper:

“The data provider will present a set of usage restriction choices to the data consumer based on the credentials presented upon data access. The consumer will select the appropriate usage restriction(s) based on the intention(s) of the access and convey this to the provider. This agreement is logged by a trusted third party that is part of the “Provenance Tracker Network” described in detail below.” [32].

There might be some issues in the implementation of HTTPA such as the required disk space for storing the accounting logs or the extra operations needed in comparison to normal HTTP. However, HTTPA may be a smart, open, standard solution. Accordingly, it would become a protocol on top of HTTP, following the same semantics without breaking the current web standards (e.g. REST).

9.2 Privacy Management in IoT

This issue is also one of the motivations in the HTTPA project, already described in Section 9.1.

Nowadays, end users tend to divulge sensitive information about their private life, thoughts, tastes and intentions on the web without thinking too much about the consequences. Since 2006, the year where online social networks started to proliferate on the web, there has been a revolution in the market. Quite a lot of companies have transformed their business model, based on the offer of certain services and information in exchange for users’ personal data, including email addresses and phone numbers.

As a consequence, people often receive unsolicited promotions and offers, realizing that their data could have been sold to third parties. Thus, protection of the data and proper authentication/authorisation is not enough. Privacy is a missing factor.

A new possible direction might be the combination of several levels of privacy and a federated identity using protocols like OpenID Connect that a user may utilize for authentication in different applications, as it happens now. This result would be a public account that might be used to log in a social network, where I want that other certain individuals (with a unique identity managed also by OpenID) would
be able to access to all my private data. Meanwhile, I might use such an account to log in a professional application where I just want to share basic data like my email, username and a basic description.

Depending on the purpose of external applications, and the privileges the user may want to grant, this account would be a universal identity that would let the user choose who should be able to get his data.

9.3 Mobile Connect and AAA

Consequently, several leading mobile operators such as Orange, Telefonica, China Mobile, China Telecom, Tata Teleservices and Telstra started a collaborative initiative that consists in the development of a new service called Mobile Connect.

Mobile Connect[15] is a service designed to allow consumers to establish a secure authentication, working on top of the OpenID Connect protocol, but using their phone number as a unique identifier rather than using an email address. The protocol would be used by developers to provide a wide list of digital services through a smart-phone, such as retail, healthcare, government and banking, as well as in-app payments for advanced features.

Moreover, when a user needs to use a certain service, he would just be required to fill in an input field with his PIN number instead of using a different username and password for each service. In consequence, the protocol would unify the login experience, which is the furthest purpose OpenID Connect traditionally has had only, with the aforementioned services. As a result, Mobile Connect would simplify the user interactions, while using a convenient, secure protocol that protects the privacy of the users.

Mobile Connect is being currently designed and the specifications will start to be evaluated in 2015.
Bibliography


Appendix A

IoT Marketplace: User Interface
Figure A.1: "Home" page of the IoT Marketplace
Figure A.2: Visualization of a Temperature Stream from a sensor in Stockholm
Appendix B

SENSEI’s Enhanced Architecture

Above, the Figure B.1 shows the interactions between the associated parties within the enhanced version of the SENSEI’s architecture. The study discarded this architectural model because of the introduction of different security token domains, an advance feature not needed for the project.

Figure B.1: Enhanced version of the SENSEI architecture