Evaluation of Single Sign-On Frameworks, as a Flexible Authorization Solution

OAuth 2.0 Authorization Framework
Abstract
This work introduces the available authorization frameworks for the purpose of Single Sign-On functionality within an enterprise, along with the fundamental technicalities. The focus of the work is on SAML 2.0 and OAuth 2.0 frameworks. Following the details related to available protocol flows, supported client profiles and security considerations, the two frameworks are compared in accordance with a set of factors given in a criteria. The report discusses the possibilities provided by a Microsoft Windows based infrastructure, as well as different scenarios and their feasibility in an enterprise environment. The preferred framework, OAuth 2.0, is selected according to the given criteria and the comparative discussions.

Keywords: Single Sign-On, Authorization, SAML 2.0, OAuth 2.0, OAuth 2.0 flow, Federated authentication, Delegated authorization
Preface

As someone studying computer science and specialized in the security aspect of the field, I have come across the fact that, as you move ahead in your academic career, you are introduced with the latest advancements in the field and you will incorporate these in your daily tasks. I have personally experienced the impact of Single Sign-On (SSO) technology when it was introduced for IT infrastructure at Linnaeus University and it triggered not just an understanding about SSO, but also revelations regarding how closely the knowledge and the practice are related in this field. I also realized how a single technology can make students' lives easier. I knew right away that there are much more potential and much more diverse use-cases to be covered.

As a result, I was encouraged to explore the possibilities provided for businesses by utilizing SSO. I was also curious about the positive and/or negative consequences surrounding the topic. This work is a mere try in fulfilling this endeavour.

Acknowledgements

I would like to take this opportunity to thank my supervisor Ola Flygt, for his encouraging guidance throughout this work. His experience led me to conduct this work as efficiently and swiftly as possible. I also would like to thank Stefan Petersson for setting up the necessary infrastructure and virtual machines and Niklas Brandt for his technical advice.

My special thanks goes to Tobias Holm, my contact person at Eurostep Group AB and Peter Bergström, for introducing me to the company.

Last but not least, I am grateful for Mathias Hedenborg’s meticulous critique as the examiner of this work, taking the quality of this report to the next level.
Contents

1 Introduction 1
  1.1 Background .......................................................... 1
  1.2 Problem definition .................................................... 1
  1.3 Previous research ...................................................... 2
  1.4 The purpose of the work .............................................. 2
  1.5 Scope and limitations .................................................. 2
  1.6 Target group .......................................................... 3
  1.7 The outline of the report ............................................. 3

2 Methodology 4
  2.1 Scientific approach ................................................... 4
  2.2 Data collection ....................................................... 4
    2.2.1 Industry best-practices .................................... 4
    2.2.2 Implementation ................................................ 5
  2.3 Approaches for reporting results .................................... 5
  2.4 Analysis ............................................................ 5
  2.5 Ethics and social impacts ............................................. 5

3 Authorization frameworks and related topics 7
  3.1 Authentication and authorization ................................... 7
  3.2 Federation and delegation .......................................... 7
  3.3 The Importance of Single Sign-On technology ....................... 8
  3.4 Active Directory Federation Services (ADFS) ....................... 8
  3.5 The importance of BYOD ............................................. 9
  3.6 SOAP and RESTful services ......................................... 9
  3.7 Single Sign-On categories and frameworks .......................... 10
    3.7.1 SAML 2.0 ....................................................... 10
    3.7.2 OAuth 2.0 ..................................................... 10
    3.7.3 OpenID 2.0 .................................................... 10
    3.7.4 OpenID Connect 1.0 .......................................... 11
  3.8 The case of Eurostep ................................................ 11

4 Results 12
  4.1 SAML 2.0 internals .................................................. 12
    4.1.1 The SAML 2.0 actors ....................................... 12
    4.1.2 Web browser SSO profile .................................... 12
    4.1.3 Securing SAML 2.0 traffic ................................... 12
  4.2 OAuth 2.0 internals ................................................ 13
    4.2.1 The OAuth 2.0 actors ....................................... 13
    4.2.2 Different client profiles .................................... 14
    4.2.3 Authorization code grant flow ................................ 14
    4.2.4 Implicit grant for browser-based client-side applications flow ........................................ 15
    4.2.5 Resource owner password credentials grant flow .......... 16
    4.2.6 Client credentials grant flow ................................ 16
    4.2.7 Securing OAuth 2.0 traffic .................................. 16
  4.3 Development best-practices ......................................... 17
    4.3.1 Native applications with SAML 2.0 .......................... 19
    4.3.2 Native applications with OAuth 2.0 .......................... 20
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Analysis and discussion</td>
<td>21</td>
</tr>
<tr>
<td>5.1</td>
<td>The criteria</td>
<td>21</td>
</tr>
<tr>
<td>5.2</td>
<td>Scenarios to consider</td>
<td>22</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Company internal domain (or customer’s local domain)</td>
<td>22</td>
</tr>
<tr>
<td>5.2.2</td>
<td>External accounting and authorization</td>
<td>22</td>
</tr>
<tr>
<td>5.2.3</td>
<td>External authorization</td>
<td>23</td>
</tr>
<tr>
<td>5.2.4</td>
<td>Connecting to a cloud</td>
<td>23</td>
</tr>
<tr>
<td>5.3</td>
<td>The comparison</td>
<td>24</td>
</tr>
<tr>
<td>5.4</td>
<td>Legacy system support</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>28</td>
</tr>
<tr>
<td>6.1</td>
<td>Final word</td>
<td>28</td>
</tr>
<tr>
<td>6.2</td>
<td>Future work</td>
<td>28</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>
1 Introduction

This introductory section will go through the problem tackled in this report, as well as the necessary definitions to understand the structure and extent of this report. A formal problem description will be formulated and presented, along with the limitations involved with this study.

1.1 Background

Today’s business IT needs and requirements are more and more evolving into a recognizable pattern. Whether provided internally or through agreements with third parties, IT companies are dependant on services. These services are either used to serve customers, or internal needs of the employees. One way or the other, the point is, there are a number of clients, not necessarily in the same geographical location, accessing more than one service, again not necessarily provided by a single provider. On the other hand, real clients access these services using client applications. Client applications will be in different flavours and consequently will be running on different devices.

Two major types of client applications are web-based client applications, like web browsers, and native client applications, where native applications include static (desktop) applications and mobile applications. Considering the acceptance of Bring Your Own Device (BYOD) practices by companies, as well as employee tendencies in this regard, it is important to come up with an efficient and more importantly, secure way to support all of these different requirements. The fact is, the same set of requirements are the case within the industry, as it can be observed in Zach Dennis’ writing [1].

As it might be imagined, this is a rather broad spectrum to be discussed. This thesis work will deal with a specific set of circumstances as its problem. In this way, the set of variables is slightly smaller and thus, the complexity of the situation is easier to tackle.

1.2 Problem definition

The problem to be solved in this thesis work can be explained as the following.

Which framework from the available frameworks can be considered as the best unified and scalable solution to control authorization over accessing different resources in an enterprise infrastructure? The solution is expected to support both web applications and mobile devices, as well as inter-domain and intra-domain resource access scenarios. The detailed technical definition of the scope and limitations to be taken into consideration is described in subsection 1.5.

The fact of the matter is that, one seldom can achieve a general solution within information and communication technology requirements. Therefore, it is important to stress that the results of this work are not intended to be practical or even feasible for any scenario using any enterprise infrastructure.
1.3 Previous research

Most of what can be found as a reference, comparing different Single Sign-On frameworks, is in the form of solutions to certain use-cases. These use-cases define a strict set of requirements, resulting in what can be considered as a consultant’s opinion. Although this report also relies on an acceptance criteria to compare different frameworks, the number one resource for the author is the set of standards and Request for Comments (RFCs), available for each framework.

The work of Ranjbar and Abdinejadi [2] is a great relevant starting point for this topic, since they also follow a chosen criteria to compare between different SSO frameworks in an academic fashion. One should also mention the great blog post by Zach Dennis, *Choosing an SSO Strategy: SAML vs OAuth2* [1], as an informative resource. His business requirements oriented approach in comparing SSO frameworks is a great example of real-world requirements oriented solution seeking. By combining both approaches from these and also other sources, one can have a better understanding of the necessities for academia, business world and the connection between the two.

1.4 The purpose of the work

*Small and Medium Businesses (SMBs)* usually do not have the means to employ dedicated professionals to manage and expand their IT infrastructure and keep it secure at the same time. Thus, SMBs depend on consultants or contracted outside companies to do the job for them. It would be much more beneficial for a company to be familiar with the kind of solution that is going to tackle its problems; and since the problem at hand is a recurring one, many businesses with similar enterprise IT infrastructure and the same set of requirements can benefit from the given solution.

This report aims at finding the most suitable Single Sign-On framework, providing delegated authorization. More specifically, the report will consider Eurostep’s set of requirements as an example from the industry. The comparison that is going to be given between different frameworks can also be beneficial for other use-cases involving these frameworks.

1.5 Scope and limitations

In order to be able to walk in a productive path in a research and with the purpose of avoiding distractions, the scope of the study has to be decided on.

The focus of our evaluation throughout this work would be the Single Sign-On frameworks that can be used within an enterprise environment. We are specifically looking for independent implementations. For instance, the kinds of implementations using third-party entities such as Google or Facebook are not relevant for our purposes. Although the same technologies discussed in this writing might have applications in such implementations. We also require adaptability to and interoperability with a Microsoft Windows infrastructure. This requirement is dictated by the case of Eurostep AB, described in subsection 3.8. Considering this, we will see that the two prevalent choices are SAML 2.0 and OAuth 2.0. As a result, these two frameworks will be the subjects of our scrutiny.
1.6 Target group

This writing is intended as a self-containing report for audiences possessing Bachelor level computer science knowledge. It is assumed that readers are familiar with computer security concepts. Consequently, the text should be self-explanatory for people with professional background in computer and network security topics as well. It is my hope that the information provided here can be of use to businesses and individual professionals within the industry.

1.7 The outline of the report

Following this introductory section, the report encompasses three major parts. The first one, section 3 includes the necessary theory involved with the topic and the problem. All major theoretical concepts will be gone through briefly, to basically familiarize the reader with what the report is about. The next part, section 4 along with section 2 involves the approach taken during this thesis work towards answering the problem at hand. It will include the practical aspects of the work, as well as available best-practices. Finally, these results will be followed by a consequent discussion and comparison in section 5. This part will go through the pros and cons of the facts retrieved from the practical and literary study part. As a final wrap up, section 6 will be our concluding part, repeating the most important facts from other sections, along with ideas to improve this work in the future.
2 Methodology

This section goes through the processes and methods used in this report for connecting the dots. The goal is to start from credible and reproducible facts and arrive at a credible result, improving our insight regarding the topic.

2.1 Scientific approach

Considering the fact that, the introduced frameworks and standards have drastic differences in their detail, implementation and application, a selective approach according to a general criteria, as well as the requirements of the company, will provide us with a more decisive and clearer choice. By following this doctrine, it is the goal of this writing to provide a solution, based on the facts surrounding the premisses of the question.

If we are to categorize the work in terms of qualitative or quantitative, different aspects of this research has the characteristics related to a qualitative approach. Our research is exploratory and thus its focus is the big picture. There are no numerical data gathered. Instead, what we have are descriptions. Also, the data gathering instrument is the researcher himself, using mostly documentary analysis. The approach towards the comparisons of our results is also a qualitative one, for we are choosing qualitative properties, qualities, of the introduced frameworks and standards. As a result of all this, the judgement will be subjective, biased and impossible to measure, the same as any other qualitative data. One can consider a qualitative research as the first phase of a more focused quantitative research and in our case, the result of this work can be extended using qualitative measurements to get a more concrete result in a more focused area.

To minimize the aforementioned characteristics of the result and to increase its academic credibility, an acceptable criteria will be introduced and the preference will be chosen in accordance with this criteria. It is the author’s hope to farther extend the usefulness of this writing, to include more than a single application, by following such a strategy.

2.2 Data collection

The approach explained above requires factual data collection as its first step. To fulfil this, we will make use of resources such as the official description of standards like Request for Comments (RFCs), academic publications, books, on-line resources and industry best-practices.

2.2.1 Industry best-practices

Considering the practical nature of the research question and the fact that the frameworks and standards are meant to be implemented in the industry, one of the major resources to be used as a proven basis, is industry best-practices. As it is the case for any other comparison, one can always make good use of earlier experiences and experiments.
2.2.2 Implementation

We will follow this with an implementation of our own with the purpose of testing the ease of implementation, as well as checking the compatibility and interoperability with a Microsoft ecosystem. The premises of the implementation will be dictated directly by the requirements and the infrastructure specifications of the company.

2.3 Approaches for reporting results

When it comes to reporting results, we will mention the facts, resulting from the data collection effort. This will include the details and mechanics of the frameworks under our evaluation. One can consider section 4 as the elaboration of the findings of this study. The actual comparison, using these findings along with its necessities, will be followed in section 5.

2.4 Analysis

The analysis will include the full description of the frameworks chosen for comparison. After going through the details such as defined roles and authorization flows, as well as the interaction between these roles, the comparison will be carried out. We will conduct the comparison in accordance with the desirable elements, defined within the criteria. In other words, the criteria will be considered as a ruler to put the comparison in an acceptable and understandable perspective. Obviously, certain comparisons, conducted by previous publications and matching our requirements, will be used.

It is worth mentioning that, to have an efficient report, we will only consider the likely candidates and obvious non-compliant frameworks will be discarded without further ado.

2.5 Ethics and social impacts

There are two different approaches to ethics when dealing with academic research. An academic work, published and circulated in a society, has an impact on it. It is absolutely necessary for the author to consider this in advance and encourage the ethically correct aspects and uses of the work within the society. It is paramount to take responsibility of one’s ideas and findings, published to the public domain and try to lead the correct use of it.

The first aspect is the ethical use of the results in any possible application. Generally speaking, within the field of Computer Science, security consciousness is the cornerstone for data-processing. Regarding our specific case, any service using a network medium to send and receive data needs to take the authenticity and confidentiality of data into consideration. Access control being the main reason for Single Sign-On, needs to be implemented with safekeeping user credentials and access rights in mind. The results and surrounding knowledge expressed in this work should not be used to produce services with hidden malicious features, or creating unnecessary means of control, where it is not needed. Furthermore, the developers are encouraged to adapt an open policy when it comes to the security sensitive parts of their product, or to follow practices such as dual licensing to keep both public and business interests fulfilled.
The second consideration is the way in which the research has been conducted. This deals with the process of extracting data, reading literature, testing and finding answers out of the gained information and knowledge. A researcher has to report initial wrong ideas, incorrect results, mistakes and even bias. Consequently, the work will be much more fruitful and better fulfilled, making the reader aware of the reasons for failure, helping them not to thread in the same path. For instance, in our case, the initial belief was of the advantage of SAML 2.0 protocol. Later on, following the academic procedure and feature comparisons according to the defined criteria, resulted in the OAuth 2.0 being the chosen framework.

Both introduced aspects have to be taken into consideration and readers are strongly encouraged not to take this matter lightly.
3 Authorization frameworks and related topics

As previously mentioned, we start with creating a knowledge base by briefly going through the relevant theoretical definitions and explanations. This will pave the way to gather, understand and analyse the information within literature and the results or practical tests.

3.1 Authentication and authorization

Every security aware infrastructure includes the necessary mechanisms for Authentication, Authorization and Accounting (AAA). Our focus here is authentication and authorization.

**Authentication**  The process of checking an entity’s identity, or in other words, to check if someone (or something) is who they claim to be, is called **authentication**. This can be fulfilled by means of a set of credentials, which can include different forms of identifiers, provided by the entity to an authenticator [3]. Identifiers or identities can be in different forms such as, user-names, unique email addresses, etc. The user will be challenged for the other part of a credential and has to respond correctly for a successful authentication. This part can be something the user knows, something the user has, or something the user is, such as a password, a smart card, or a biometric feature, respectively [3].

**Authorization**  Authorization can be described as the verification of user access rights to a requested resource. This can be in the form of different levels of interaction with certain resources, such as files, or accessing certain services, such as an email account. An authorization process usually requires a successful authentication of the user [5].

3.2 Federation and delegation

The concepts of federated authentication and delegated authorization are crucial parts of Single Sign-On technology. Both notions are described by Boyd [5] and we shall go through them.

**Federated authentication**  Having federated authentication means that applications use authentication services that are external to them [5]. This means the account information is kept and managed by external services and the authentication is done using these services through the application connecting to them. An example of such a service is Active Directory (AD).

**Delegated authorization**  By delegating authorization to another application, owner of a resource grants them permission to perform some action on their behalf. A crude example can be giving the keys to your car to another person, granting them permission to park your car [5].
3.3 The Importance of Single Sign-On technology

Today’s business requirements dictates that employees need to use numerous services. Whether these services are provided internally within the organisation, or externally, this requirement translates to having different authentication mechanisms on multiple systems. The same applies to users’ personal activities on the Internet. The trend can lead to users experiencing password fatigue and consequently deviate from the best-practices given within the organisation’s security policy [6]. Different resources unanimously choose Single Sign-On as a solution to this predicament [1], [4], [6]. In a Single Sign-On configuration, the user directory and authentication mechanisms are unified, instead of traditional distributed settings.

In a multi-domain infrastructure, one domain holds user credential data and makes it accessible to other domains by means of a trust relationship. Figure 3.1 further illustrates this concept [7]. Here, the secondary domain trusts the primary domain to perform the authentication process on its behalf.

3.4 Active Directory Federation Services (ADFS)

Active Directory Federation Services (ADFS) is a standards-based service and a component in Microsoft Server family of operating systems, starting from Windows Server 2003 R2 with ADFS 1.0. The latest available version is ADFS 3.0 on Windows Server 2012 R2. ADFS provides web Single Sign-On within the boundaries of
organisational Active Directory forest infrastructure. ADFS also provides identity federation between trusting organisations through inter-forest trust relationships [8, 9]. The SAML 2.0 standard is supported by ADFS, as well as OAuth 2.0 in the most recent version, ADFS 3.0 [10, 11].

Software developers can make use of provided libraries, such as Windows Azure Active Directory Authentication Library (ADAL), to enable their client applications to use the provided SSO capabilities to access one or more web-based APIs, provided by Resource Servers [12]. The Microsoft Developer Network page includes GitHub links for native ADAL libraries for different mobile or static platforms, including iOS, OSX and Android [12].

### 3.5 The importance of BYOD

Bring Your Own Device (BYOD) and more generally, Bring Your Own Technology (BYOT), are descriptive terms for connecting employee owned devices or technologies (hardware or software) to company infrastructure. The practice is considered a liberty for employees, but actually involves far reaching benefits and risks for both parties. For instance, having employee owned devices connecting to business services annuls the requirement of buying company owned hardware, as well as certain amount of time required for the relevant training. On the other hand, a refusal by the employer would create the dissatisfaction of employees [13].

According to statistics mentioned by Miller et al., there was a 35 percent increase in smart-phone ownership rate between May 2011 and February 2012 in United States. Furthermore, 71 percent of people between the ages of 25 and 34 own smart-phones [13]. Since having two or more devices is far from being convenient or practical, the BYOD approach sounds like an eventualty. On the other hand, the risk to security (with regards to company data) and the risk to privacy (with regards to employee personal data) goes hand in hand with the adoption of BYOD [13].

As a result, a feasible solution can be the use of personal hardware as a portal to connect to a centralized data storage or service. Cloud computing can be an example. This is where Single Sign-On comes in and could provide us with the means for the necessary access control.

### 3.6 SOAP and RESTful services

Simple Object Access Protocol (SOAP) is a protocol for exchanging information in a structured manner in decentralised environments [14]. Web services in networked environments is an example of this type of decentralisation. SOAP uses Extensible Markup Language (XML) for its information exchange. SOAP messages are XML envelopes with a header and a body, carrying both a payload and relevant control information [15]. SAML 2.0 (or any other framework in general) can use SOAP for sending and receiving requests and responses. The details are defined in SAML SOAP binding [15].

Given the fact that SAML 2.0 goes back to year 2005, it uses SOAP 1.1 specification from 2000. The latest iteration of SOAP is 1.2 from 2007.

REpresentational State Transfer (REST) is an architectural style, based on the architecture of World Wide Web. A RESTful Service is the one built according to REST constraints and is mostly used in implementations of Service Oriented Architecture (SOA) [16]. According to Pautasso [16], these constraints are addressability,
uniform interface, stateless interactions, self-describing messages and hypermedia. The architecture uses the HTTP protocol, URI and HTML to achieve these goals.

Without getting into the details of RESTful Services (within a network), or RESTful Web Services (on the Web), OAuth 2.0 can be utilized for efficiently securing of such services, providing access for authorized users [17].

3.7 Single Sign-On categories and frameworks

When it comes down to different protocols and frameworks currently present in the industry, one can only think of a few names. These are Security Assertion Markup Language (SAML), OAuth, OpenID and very recently, OpenID Connect. Here, we are going to briefly describe these different standards. Further detail on the chosen ones will be included in section 5.

It needs to be particularly mentioned that, one of the major uses of the aforementioned frameworks and protocols is the handling of the authorization requirements of Application Programming Interfaces (APIs). This is especially true about web applications. Whether it is a Google, or Facebook type of on-line service provided as an API, or it is a company’s internal web application API, there will be a need to provide delegated access to certain resources using a API as an interface. Thus, the accessing application has to be authorized for delegated access. In legacy applications, the authorization was done by giving the user-name and password of the owner to the application [5]. One can easily imagine how tedious it will be to work with a number of applications accessing different APIs and asking the credentials required for each authorization separately. It also creates higher security risks, since the owner has no choice but to trust these applications [5].

3.7.1 SAML 2.0

SAML was created by OASIS Security Services Technical Committee and its current version is SAML 2.0, dating back to 2005 [18]. SAML relies on XML-based data to transfer authentication and authorization details. Using SAML, users, applications and services can exchange identity information [2]. This is done through SAML Assertions, which are compressed, encoded and possibly encrypted XML nodes [1].

3.7.2 OAuth 2.0

The latest iteration of OAuth, formalized in 2012, is the version 2.0 [19]. As one can imagine, it is much more accommodating to current trends and needs in the industry, as we will see in section 5. OAuth includes the notion of Access Token as the mechanism of choice for allowing access to restricted resources. In other words, an Access Token is the authorization issued to a client [19].

3.7.3 OpenID 2.0

At the time of this writing, the latest iteration of OpenID from 2007, the version 2.0, has become obsolete by the introduction of OpenID Connect. We will briefly mention a general description here and move on to OpenID Connect. Both standards are created by OpenID Foundation.

OpenID focuses on providing decentralized authentication for end-users, throughout cooperating websites. Meaning, users can use a single identity to authenticate
against different websites. The idea is called *Bring Your Own Identity (BYOI)* and it is in wide use today. Standard HTTP(S) requests are used for this purpose [20].

### 3.7.4 OpenID Connect 1.0

*OpenID Connect 1.0*, finalized on February 2014, adds an identity layer to OAuth 2.0, enabling the verification of an end-user. This is done by using the data from an authentication, which an involved OAuth 2.0 Authorization Server performs [21]. Considering the similarities between OAuth 2.0 and OpenID Connect 1.0 and given the fact that the current implementation at Eurostep AB is using OAuth 2.0, it would be enough for our purposes to focus on OAuth 2.0.

### 3.8 The case of Eurostep

Eurostep Group AB produces a Product Life-cycle Management (PLM) solution. The solution is called Share-A-space (SAs) and functions as a server-client service, using Microsoft SharePoint with clients running in web-browsers.

Currently, the product supports Single Sign-On, utilizing OAuth 2.0 *Client-Side Web Application Flow*, along with Active Directory Federation Services (ADFS). Different authorization flows are explained in subsection 4.2. The company is interested in OAuth 2.0 and all that it offers as a Single Sign-On solution, within an enterprise. Utilizing BYOD possibilities is a major focus here.
4 Results

The "Results" section includes the immediate raw findings of this study, whether they are theoretical results based on the literature, or practice results based on practice and testing. The Analysis and discussion regarding these findings will be followed in the upcoming section.

4.1 SAML 2.0 internals

To understand how SAML 2.0 works, we need to get familiar with certain vocabulary and a protocol flow used for SAML 2.0 Single Sign-On. SAML 2.0 transmissions carry SAML 2.0 assertions, which is a "package of information", structured in XML format. These assertions (can be thought as SAML 2.0 tokens) are made by SAML authorities, or asserting parties [18]. Together with relying parties, we can generalise them as actors.

4.1.1 The SAML 2.0 actors

Here is a brief description of different SAML 2.0 actors.

Service provider The server holding protected resources in different forms, which is being accesses by a client, by means of an API. The API is provided by the service provider.

Client The party requesting access to a protected resource on the service provider is the client. SAML 2.0 only supports the web browser SSO profile as a client, meaning, the client always connects to the service provider using a web browser as its User-Agent.

Identity provider This party checks the identity of clients and issues assertions to allow or deny access to protected resources.

4.1.2 Web browser SSO profile

SAML 2.0 only supports web browser based clients. The most common protocol flow for SAML 2.0 is web browser SSO profile. The flow of events can be seen in Figure 4.1. The flow itself is the author’s own work, with the introduced flows in [1] and [22] for the web browser SSO profile in mind.

The steps in Figure 4.1 are self-explanatory. The important fact to be pointed out is the lack of support for native applications by SAML 2.0. This is not surprising at all if we consider the release date of the protocol, 2005. Back then nobody could imagine the explosive popularity of mobile devices within consumer and business communities.

4.1.3 Securing SAML 2.0 traffic

Security of a SAML 2.0 implementation can be addressed in different levels. In fact, the standard includes a separate documentation for this purpose [23]. The more specific part to be mentioned, other than the use of SSL/TLS, is XML Signature. Considering the fact that SAML 2.0 Assertions are in XML format, signing these
documents provides integrity and message authentication, securing communications at message-level. The details regarding XML Signature is given in *XML Signature Syntax and Processing (Second Edition)* [24].

### 4.2 OAuth 2.0 internals

To understand the mechanics of the OAuth 2.0 framework and comprehend the notion itself, first we need to explain the theory behind the involved blocks.

#### 4.2.1 The OAuth 2.0 actors

When dealing with different client profiles, the OAuth 2.0 framework works according to different predefined protocol flows. A number of actors take part in these protocol flows.

**Resource server**  
This is the hosting machine, holding user-owned material, including different files such as, photos and videos, as well as different datasets such as, contacts and calendar, etc. These resources will be accessible by means of an API (provided by the resource server) and will be protected by OAuth 2.0.

**Resource owner**  
*Resource owner* is the user, owning the resource, which is being served by the resource server. The owner can grant access to their resources on the request of a client.

**Client**  
This is the party, requesting access to protected resources using the provided API. A resource owner delegates authorization to a client for manipulation.

---

Figure 4.1: A step-by-step flow of events in the *web browser SSO profile* flow
of the resources, on their behalf.

**Authorization server** Clients, requesting access to protected resources, get their access tokens from an *authorization server*. Access tokens are issued with the permission of the resource owner, on their behalf. *Authorization server* and *resource server* (API provider) can use the same application in small implementations.

### 4.2.2 Different client profiles

Different client profiles are defined within the OAuth 2.0 framework. All these different client types are used to access the resource server and consequently the protected resource, by means of the provided API.

**Server-side web application** This is the case of a client as a web application, running on a web server and being accessed by a user, who is an owner of the resource. The API call is done through server-side programming languages. In this type of client, the user has no access to the client secret, or access tokens.

**Client-side application, running in a web browser** This type of client runs inside a web browser on the user’s side. Different ways to distribute such a client are JavaScript code within a web page, browser extensions, or plug-ins e.g. Flash. The application code is accessible by the client and the OAuth 2.0 credentials cannot be considered confidential and can be accessed by the resource owner. Because of this, some API providers do not support this type of client.

**Native application** This case refers to an installed application with close similarities to the client-side application type. The OAuth 2.0 credentials are not considered confidential in this case either. Since this type is an installed application, it usually is much less capable, compared to a web browser.

### 4.2.3 Authorization code grant flow

The most commonly used OAuth 2.0 protocol flow is the *authorization code grant flow*, or alternatively, *server-side web application flow*. The flow of events can be seen in Figure 4.2. The flow itself is the author’s own work, with the introduced flows in [1] and [19] for the *authorization code grant flow* in mind.

In this flow and the next one, Figure 4.3, the *implicit grant for browser-based client-side applications flow*, the User-Agent is the browser, a portal to the Resource Server. As such, we consider it as an extension of the API provider, coloured in blue. The Client, whether it is a Server-Side Application or a Client-Side Application (in different flows), is coloured in green. Lastly, the combination of the Authorization Server with the Identity Provider is in dark yellow. We will follow this colour coding throughout this report.

The steps in Figure 4.2 are self-explanatory. What needs to be pointed out is that, in the *authorization code grant flow*, the Access Token does not pass through the User-Agent (browser), increasing security by reducing the risk of leaking it to client-side malicious code. This leak can happen through browser history, referrer headers, JavaScript, etc. [5]. One possibility with this flow is to receive an Optional
Refresh Token with the Access Token. Many applications, requiring long-lived offline access to data, use this flow along with Refresh Tokens. This creates the risk of access to more than one user’s data [5].

Finally, the flow should be used for

- Long-lived access scenarios,
- Server-side (web) application scenarios,
- Safekeeping of OAuth 2.0 tokens from being accessed by the user, amongst others [5].

### 4.2.4 Implicit grant for browser-based client-side applications flow

A slightly different flow, supporting client-side applications is the implicit grant for browser-based client-side applications flow, or implicit grant flow is shown in Figure 4.3. The flow itself is the author’s own work, with the introduced flows in [1] and [19] for the authorization code grant flow in mind.

The major difference for this flow is the User-Agent’s access to the Access Token. For this very reason, the flow does not support refresh tokens. The idea is that, if an Access Token gets leaked, it could be used only for a short limited time [5].

This flow should be used for

- Temporary data access scenarios,
- When the user performs regular logins to the API provider,
- Client-side (Web) application scenarios,
- When the User-Agent (Browser) can be trusted [5].
4.2.5 Resource owner password credentials grant flow

This flow requires the user to completely trust the client application, since the password is accessible by it. The flow provides the user with an Access Token in exchange for their credentials. It is still better than HTTP Basic access authentication (BA), since authentications happens only once for numerous applications [5]. Generally speaking, it is not recommended to be used and will not be our focus.

4.2.6 Client credentials grant flow

This flow is relevant for cases where the Client Application is the owner of the protected resource. Thus, there is no delegation of access present. The case of a Client Application, using remote storage APIs to read and update the application’s stored data can be a representation for this flow [5]. We will not focus on this flow as well.

4.2.7 Securing OAuth 2.0 traffic

The transition from OAuth 1.0 to OAuth 2.0 included the removal of cryptographic signature requirements. The initial intention was to provide an easier framework, simplifying development for mainstream developers. The Internet-Draft, *OAuth 2.0 Message Authentication Code (MAC) Tokens* [25], describes the use of *Message Authentication Code (MAC) Tokens*, when accessing protected resources. The latest iteration is from January 15, 2014.

Signatures  OAuth 2.0 uses *bearer tokens*, meaning these tokens are not signed or verified. As we mentioned above, this can increase the adaption of the framework and is in the interest of web developers. Enterprise communities however, much prefer the use of MAC Tokens [26].

Figure 4.3: A step-by-step flow of events in the *implicit grant for browser-based client-side applications flow*
The proposed design considers the Authorization Server as a trusted third party, providing session-keys (MAC keys) to consumers, which are clients and resource servers. The MAC itself is a hash function, using a session-key to create a unique and practically unforgeable hash of a message. The use of MAC Tokens addresses the concerns resulting from the importance of security in an enterprise business environment.

**Access Token validation**  The practical way for a Resource Server to make sure that an Access Token, included with a request for a protected resource, is valid, would be the validation process \[1\]. By validation we mean that, upon the receipt of the request, the resource server sends the Access Token to the Authorization Server for validation. The request will be handled only after successful validation of the Access Token.

**Access Token revocation**  In certain cases, it makes sense to revoke an Access Token, which is no longer needed. This is especially useful for long-lived access, such as the case of authorization code grant flow. An example can be a client application, cleaning after itself during its uninstall \[5\]. The revocation is defined in \[27\] and can be performed with a HTTP request, including

```
Context-Type: application/x-www-form-urlencoded
token="access token"
token_type_hint=access_token OR refresh_token (OPTIONAL)
```

as its content. A successful revocation results in a HTTP 200 response \[27\]. A revocation will result in the invalidation of Access Token, authorization grant and any other grant based on it. The revocation steps include, the authorization server’s user credential control prior to the token revocation, invalidation of the token (whether immediately, or with some delay), finally invalidation of related tokens and grants, considering the revocation policy of the authorization server \[27\].

**Cross-Site Request Forgery (CSRF)**  Cross-Site Request Forgery (CSRF) is defined within the OAuth 2.0 Authorization Framework \[19\], as a possible exploit, in which an attacker can manipulate a client’s redirection URI. By replacing the Authorization Code or the Access Token, the client will have access to the attacker’s protected resource. For instance, the client’s confidential information, such as bank account details, could end up being saved on the attacker’s protected resource \[19\]. This exploit can be mitigated by using the state parameter. For instance, in the case of OAuth 2.0 support in ADFS, a uniquely generated pseudo-random value will be sent in a request for an authorization code, along with the state parameter. The response should contain the same state value \[28\].

**4.3 Development best-practices**

There are also practical, software development oriented results available to be included in this report. As it is presented in Figure 4.4 for the authorization code grant flow and in Figure 4.5 for the implicit grant for browser-based client-side applications flow, much more detailed interactions between different actors can be seen. Both figures include different actors on separate lifelines using a simplified UML sequence style diagram. The colour coding follows the same pattern as other
diagrams throughout this report. Both flows are the author’s own work, with hints from the introduced flows in [29] and [30].

Figure 4.4: The detailed flow of events in the authorization code grant flow

The steps are self-explanatory, but in case of Figure 4.4, there is an alternate flow, which can be opted for. When the Server-Side App receives an Access Token in a response message from the Authorization Server, there might not be a need to redirect the Browser to the original URL for the protected resource. The Server-Side App can directly fetch the resource from the Resource Server, present it to the Browser and ultimately to the user. Let us further explain the transactions.

The initial request for a protected resource is in the form of a HTTP GET command for the protected resource’s URI. Note that the Browser has no access to the actual URI of the resource on the Resource Server. Instead, URI’s in Client App’s domain will be accessible and mapped to the original ones by the Client App [29]. For instance, the request can include

http://client-app-domain.com/customer1/2 ,

instead of


The HTTP 302 redirection includes [29]
• 302 redirect code,
• *location* header, including Authorization Server’s authentication URI,
• query string key-value pairs:

```
response_type="code"
client_id="client app identifier"
redirect_uri="redirection endpoint"
```

Successful authentication is also followed by a HTTP 302 redirection, including

• 302 redirect code,
• *location* header, including Client App’s redirection endpoint URI,
• query string key-value pairs:

```
code="authorization grant (authorization code)"
```

The response with the Access Token from the Authorization Server to the Client App is in the form of a HTTP 200 response, carrying the Access Token. The Access Token itself is a JSON object and includes

```
access_token - string
token_type - string
expires_in - integer (OPTIONAL)
```

Afterwards, as we mentioned earlier, the Client App can redirect the Browser to the initial URI of the protected resource (retained within a web browser cookie or the session state), or simply get on with the fetching of the resource. Similar transactions can be considered for the second diagram as well. Detailed examples for using the *authorization code grant flow* with OAuth support in ADFS can be found in [31].

The other point to be made is that, developers need to consider the inclusion of the Access Token validation process by the Resource Server in every flow, as you can see in our diagrams.

### 4.3.1 Native applications with SAML 2.0

If we consider the step D of Figure 4.1, when the Client is a native application such as a mobile application, there are certain hardships for implementing the solution. Since the message here includes a SAML 2.0 Assertion and is a long message, the specification requires the use of HTTP POST instead of HTTP Redirect. After the initial launch of a mobile application and the subsequent authentication, the relaunching of the application using the generated URI is fine, but the application can not access the message body of a HTTP POST. Therefore, the application cannot have access to the generated SAML 2.0 Assertion [1].

The solution is to either embed a web view inside the native application, giving access to HTML content and extracting the SAML 2.0 Assertion, or to implement
Figure 4.5: The detailed flow of events in the *implicit grant for browser-based client-side applications flow*

A proxy server. The proxy server can extract the SAML 2.0 Assertion from the HTTP POST message and create a custom URI, including the assertion at its end. Afterwards, the proxy server can generate a HTTP Redirect message using the generated URI to initiate the application’s launch [1].

### 4.3.2 Native applications with OAuth 2.0

Using OAuth 2.0 with native applications does not have the complexities of SAML 2.0. Depending on the existence of a back-end server for a native mobile application, providing the implementer’s own API, or using the API from a third party, both the *authorization code grant flow*, or the *implicit grant for browser-based client-side applications flow* can be used. In case of a third party API provider, the developer has to decide on a custom URI scheme for the `redirect_uri`. Other than that the confidentiality of the `client_secret` has to be taken into consideration [5].

In the case of an enterprise user, whether it is an internally used solution, or something for their remote customers, the mobile back-end scenario will be the case. The implementer need to decide on one of the two flows mentioned above according to the requirements such as short-lived, or long-lived access [5].
5 Analysis and discussion

In this section, we will compare the details of SAML 2.0 and OAuth 2.0, given in section 4, to see how they differ in handling Single Sign-On, how these handling compare to one another and finally, which one is a better fit for our criteria.

5.1 The criteria

As it was mentioned in subsection 2.4, it is crucial to define an acceptance criteria for any comparison and analysis. Although we will require certain specifics, restricting certain aspects of the outcome, but from a general point of view, our ultimate goal is to be vendor neutral in other aspects. In this fashion, the result will have more generality, making it applicable to more than one specific case.

Support for different clients One of the requirements is to be able to support different client profiles. As we saw in subsection 4.2, the preference is to support server-side web applications, web browser based client-side applications and native applications. Even though the current implementation of a solution might not include all these profiles, it must be possible to add any of them in the future.

Scalability The solution has to be scalable and in itself should be able to support increasing usage. By this, our concern and focus is not the supporting mechanisms like Active Directory, or network medium utilization, etc.

Ease of implementation The solution itself has to be easy to implement for all mentioned client profiles. We are looking for easy infrastructure installation, configuration and administration. This also extends to the usability of different programming languages and availability of extra tools and libraries.

Security It is highly important and desirable to have the best possible security measures. Meaning that the solution provides the means to have a secure implementation, whether used by the implementer or not. Basically, provided security features will be a comparison factor.

Interoperability with Active Directory As the only restricting requirement amongst our criteria and as the only vendor related specification, we require interoperability and compatibility with Microsoft’s Active Directory infrastructure. Consequently, having an Active Directory Domain Controller as our identity provider and benefiting from the capabilities of Active Directory Federation Services, will be two inseparable characteristics of our SSO design.

Comparing the two prevalent frameworks, SAML 2.0 and OAuth 2.0, according to these factors will provide us with the answer to our initial research question. This criteria encompasses the key factors that are expected from a Single Sign-On enterprise solution.
5.2 Scenarios to consider

Apart from their internal implementation, Eurostep also provides implementations and custom development for its customers. When it comes to benefiting from the application of SSO, considering the technical players (roles) in different authorization flows, different designs and placements can be discussed. The decisive factors in choosing any of these placements are a combination of

- Security measures, in terms of how sensitive the customer’s data is and is it acceptable to be handled by others,

- Technical upkeep measures, meaning how much the customer is willing to spend time and money on running its own infrastructure and training its own personnel,

- The collaborative nature of application, regarding the the kind of project, requiring collaboration of different parties with a common repository for data, but having two separate organisations.

Needless to say, these factors could be the case for any similar service-oriented use as well and are not unique to Eurostep. There is a certain degree of generality in the proposed designs. Moreover, our focus is on the SSO functionality.

In the following scenarios, different actors are shown consisting of more than one component. For instance, since working with Active Directory Federation Services (ADFS) is desirable, the authorization server will be shown as a block. Within this block, the Authorization Server part represent the added OAuth 2.0 support to ADFS. In case of Windows Server 2012 R2 or Windows Azure, OAuth 2.0 support for the authorization code grant flow is already provided [10], [11].

It should also be mentioned that the colour-coding of different blocks correspond to their colour-coding within subsubsection 4.2.4. The client can be considered either a client-side web application running in the browser, or a native application. For a server-side web application, the client (green colour) would have been on the server itself. The User-Agent is the browser, a portal to connect to the resource server. Thus, we consider it as an extension of the API provider, coloured in blue.

5.2.1 Company internal domain (or customer’s local domain)

The first scenario that can be considered is also the simplest one, where all different actors involved in the authorization flow are on the same domain. This is the case for the company’s internal implementation, as well as a complete local implementation for a customer, running its own infrastructure.

As it can be seen in Figure 5.1, different actors, resource server, authorization server and client, are residing on the same network and are depicted as blocks.

5.2.2 External accounting and authorization

The next possible combination is depicted in Figure 5.2. Here, the customer owns the resource server and consequently, runs it within its local network. The authorization server on the other hand, along with the accounting of users (identity provider), is offered as a service. Thus, the authorization server block resides on the service provider’s network.
This type of approach can be motivated by customer’s lack of infrastructure and technical know-how to maintain an Active Directory implementation. Nevertheless, this is questionable from a security standpoint. The control over a company’s user accounting is highly desirable, if not necessary. As a result, the scenario is not exactly practical.

5.2.3 External authorization

Considering what we have described in the previous scenario as an insecure practice, here we move the accounting part (identity provider) on the customer’s local network. The diagram can be observed in Figure 5.3.

This case would have made more sense if the resource server was on the service provider’s domain. If the resource server was on the service provider’s domain, we would have had a centralised resource server-authorization server duo, with the domain itself having trust relationships to different customer domains. This can be the case when there are numerous resource servers, for instance, a number of application servers, or a single server, running multiple applications serving different customers, or a combination of these examples. We have also mentioned that the latest versions of ADFS include OAuth 2.0 support [10], [11], so if the Authorization Server element is only intended for adding OAuth 2.0 functionality, it would not be required for those cases.

Nevertheless, considering what we have in Figure 5.3 it would make much more sense to have the resource server and the authorization server on the same domain, whether it is the customer’s, or the service provider’s.

5.2.4 Connecting to a cloud

The last, but not least approach is a currently popular trend for any ICT service. We can consider the complete solution, resource server and authorization server, the latter including both Authorization Server element and identity provider, as a remote service. In other words, the service provider offers a cloud environment for different customers. The benefits of such an approach are like any other cloud
As we have mentioned in our criteria, one of the important factors to be considered in these scenarios is accommodating BYOD clients, hence the inclusion of Mobile Client in *client* block. Because of its characteristics, the cloud scenario can be one of the most flexible solutions in this regard.

We should also mention that, considering the presence of ADFS 3.0 in Microsoft Azure (a cloud computing platform), supporting OAuth 2.0 framework’s *authorization code grant flow* and the possibilities of using native applications with OAuth 2.0 flows, all the necessary pieces for a complete cloud implementation is at hand.

### 5.3 The comparison

What we have listed as different characteristics of these frameworks can be boiled down to a general comparison, according to our defined criteria.

**Support for different clients**  
Regarding any framework, the more numerous its supported client profiles are, the more scalable the framework itself is, as it can
address a bigger variety of requirements. In our case, we saw that the SAML 2.0 has been designed with web based clients in mind. This does not comply with the reality of today’s ICT environments. On the other hand, OAuth 2.0 and anything based on it, can address both web based and native application requirements.

**Ease of implementation**  The ease of software implementation offered by OAuth 2.0, using bearer tokens does not actually count hear, since an enterprise environment requires the security offered by MAC Tokens. There is an abundance of libraries available in different programming languages for different platform.

**Security**  Speaking of security, from a general perspective, both frameworks support signing. SAML 2.0 using XML Signing and OAuth 2.0 using MAC Tokens. Although the usage of the latter is not a requirement of OAuth 2.0 and the technical specification is not finalized yet, it needs to be a part of an enterprise implementation. The reason behind the Access Token validation, explained in subsubsection 4.2.7 is the fact that the user identity information is not included with it. A SAML 2.0 Assertion does contain user identity information, because of signing.

**Interoperability with Active Directory**  There is good support when it comes to integrating either framework with Active Directory infrastructure. The ADFS
supports authorization code grant flow for OAuth 2.0 and SAML 2.0, by itself through the ADAL library. Regarding OAuth 2.0 though, this can not be appealing to businesses using other OAuth 2.0 flows, or their upgrade plan to the latest Microsoft Windows Server infrastructure is far ahead.

Here we could sum up the above mentioned comparison results in a more reader friendly format i.e. as [Table 1](#).

### 5.4 Legacy system support

One of the advantages of OAuth 2.0 is legacy system support. An enterprise only comes across this when upgrading from an obsolete or relatively old SSO solution to OAuth 2.0. As a matter of fact, this will be the case of utilizing OAuth 2.0, as the newer framework, along with SAML 2.0, as the older framework. This is the only feasible scenario, which can happen within an enterprise. In this fashion, enterprises can use the already in place SAML 2.0 infrastructure for authorization and authentication with OAuth 2.0.

A SAML 2.0 Assertion can be used as an authorization grant to request an Access Token from a OAuth 2.0 Authorization Server. For this, the `grant_type` value has to be
<table>
<thead>
<tr>
<th></th>
<th>SAML 2.0</th>
<th>OAuth 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for</td>
<td>web browser based</td>
<td>web browser based,</td>
</tr>
<tr>
<td>different clients</td>
<td></td>
<td>native applications</td>
</tr>
<tr>
<td>Ease of implementation</td>
<td>harder implementation,</td>
<td>easier implementation</td>
</tr>
<tr>
<td></td>
<td>mandatory signing and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>embedded web view</td>
<td></td>
</tr>
<tr>
<td>Security</td>
<td>higher security enforced,</td>
<td>MAC Tokens (not enforced),</td>
</tr>
<tr>
<td></td>
<td>XML Signing</td>
<td>Access Token validation</td>
</tr>
<tr>
<td>Interoperability with Active Directory</td>
<td>supported by ADFS</td>
<td>supported by ADFS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(latest version)</td>
</tr>
</tbody>
</table>

Table 1: The comparison, in a nutshell

grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer.

The assertion value has to be a single SAML 2.0 Assertion, encoded in base64url format

A SAML 2.0 Assertion can be used for client authentication against an OAuth 2.0 Authorization Server with the client_assertion_type value being


The client_assertion value has to be a single SAML 2.0 Assertion, encoded in base64url format. Further details can be found in the Internet-Draft, SAML 2.0 Profile for OAuth 2.0 Client Authentication and Authorization Grants.
6 Conclusion

The concluding section discusses the overall outcome of this report and a grand sum up of what we have introduced throughout previous sections, as well as the possibilities of extending it in the future.

6.1 Final word

Considering the evidence introduced throughout section 4 and the following discussion and comparison in section 5 what it all boils down to is that, the combination of having dependable security, improved user experience and universal accessibility of services (following Service Oriented Architecture practices), results in the use of SSO strategies. As we saw, OAuth 2.0 framework proves to be a scalable choice, supporting different types of clients. It is especially a good candidate to be used along with RESTful services. The increased support for the framework also adds to its credibility and adds to the convenience of setting up implementations and developing applications. For instance, the addition of OAuth 2.0 support to ADFS compels businesses even more, to utilize the framework.

Another deciding factor was the support for BYOD, or mobile devices in an enterprise business environment. We saw that this is where OAuth 2.0 excels and in contrast to SAML 2.0, considers the use of these client profiles. We have also mentioned different practices to add the enterprise-level security to OAuth 2.0 implementation, making up for the lack of signing requirement for OAuth 2.0 tokens within the specification.

Using what we have introduced in section 4 as our input, we have conducted a comparison between the SAML 2.0 and the OAuth 2.0 frameworks. This comparison was done according to a criteria, which included factors required to answer our initial research question. Thus we have addressed the research question in a step-by-step fashion, from theory to results and finally, to the concluding comparison.

Overall, the obvious fact to be stressed on is that, the choosing of an end-to-end solution, requires the evaluation to be conducted in different levels. Here, by end-to-end we mean that, the solution (Single Sign-On) encompasses the whole enterprise, including users, clients, infrastructure, servers and resources. As a result, the different levels to be considered for such an evaluation should include user experience, software implementation, hardware implementation, infrastructure compatibility, ease of administration, legacy system support and future expandability, amongst others.

6.2 Future work

There can be many ways to extend this work. One can consider the evaluation of the same Single Sign-On frameworks in a Unix or GNU/Linux infrastructure environment. The comparison between such an environment with the one using proprietary technologies can help businesses by introducing alternatives. It would also be a challenge to replace a single, uniform structure like Active Directory, with numerous intertwined subsystems. Since the only way to get the set of functionalities provided by Active Directory is to do so.

The other subject that comes to mind is the specific evaluation of security related aspects for these frameworks. One can look into the possible attacks, available
mitigation methods and generally speaking, vulnerabilities of Single Sign-On frameworks, especially SAML 2.0 and OAuth 2.0. As we have mentioned within the report, OAuth 2.0 support integration with already in place SAML 2.0 implementations. One can also study the details related to different migration scenarios, or the existence of different implementations using different frameworks and the interaction of these implementations between two cooperating enterprises.

Another interesting topic that we come across with is the use of XML Signature. Considering the fact that XML builds such a big part of software systems and basically anything related to computer science deals with XML in one way or another, one can spend an entire thesis work on the topic of XML Signature. One can look for the applications, vulnerabilities etc.

One can also work on the upcoming frameworks and protocols, even the ones that are not exactly within the premises of this report. Example are BrowserID protocol and Mozilla Persona, both as decentralized authentication solution. Our focus within the report was on the authorization, which might include authentication in its process.
References


