

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Identity management: analysis of secure authentication propositions

Rúben José da Silva Torres



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: José Manuel De Magalhães Cruz

Supervisor: Gonçalo Hermenegildo

September 20, 2020



# **Identity management: analysis of secure authentication propositions**

**Rúben José da Silva Torres**

Mestrado Integrado em Engenharia Informática e Computação

September 20, 2020



# Abstract

Identity-related digital security solutions typically provide identification solutions such as authentication, authorisation, and digital signature. As such, these solutions usually use a specific protocol; hence there is a considerable amount of identity protocols that developers need to take into account when building their systems. In addition to this, many legacy systems use old protocols that modern systems do not support. Continuous integration of new identity protocols is expensive and requires exertion to make everything work seamlessly.

From this, a need arises to interconnect systems effortlessly. To find a solution, we first reviewed the possibility of a protocol converter. This, however, was determined as unfeasible, because converting very different mechanisms is, at best, challenging and may even be impossible in some instances. We later reviewed open-source IAM solutions to continue with the Dissertation. From our review, we found the WSO2 Identity Server as an excellent candidate, with support for the initial requirements and comprehensive documentation.

Subsequently, we performed a security analysis of the WSO2 Identity Server. This security analysis can be divided into two steps: threat analysis and security audit using static analysis tools. The threat analysis allowed us to characterise the system, identify threats, assets and vulnerabilities, and, in the end, to propose a mitigation plan. The security audit concluded that packages such as `org.wso2.carbon.identity` and `org.wso2.carbon.balana` might need further reviews. We additionally assumed that developers should always use static analysis tools as an aid in the developmental process.

We further conclude that WSO2 IS is a well-thought solution; with everything in place to be an extremely secure IAM solution, but this really depends on the deployment configuration.

**Keywords:** Protocol Converter, Identity protocols, Identity Management, Identity and Access Management, Single Sign-On, Threat Analysis, Security Audit, Static Analysis



# Resumo

As soluções de segurança digital relacionadas com identidade geralmente fornecem soluções de identificação, como autenticação, autorização e assinatura digital. Como tal, essas soluções geralmente usam um protocolo específico; havendo uma quantidade considerável de protocolos de identidade que os *developers* precisam de ter em consideração ao construir seus sistemas. Além disso, muitos sistemas legados usam protocolos antigos que os sistemas modernos não suportam. A integração contínua de novos protocolos de identidade é cara e exige esforço para tudo funcionar perfeitamente.

Surge então a necessidade de interconectar sistemas sem esforço. Para encontrar uma solução, primeiro analisamos a possibilidade de um conversor de protocolos. Isso, no entanto, foi considerado uma ideia inviável, porque converter mecanismos muito diferentes é, na melhor das hipóteses, desafiante e pode até ser em alguns casos impossível. Posteriormente, revimos as soluções de IAM de código aberto para continuar com a Dissertação. Na nossa análise, descobrimos que o WSO2 Identity Server é um excelente candidato, com suporte para os requisitos iniciais e documentação abrangente.

Posteriormente, realizamos uma análise de segurança do WSO2 Identity Server. Essa análise de segurança pode ser dividida em duas etapas: análise de ameaças e auditoria de segurança usando ferramentas de análise estática. A análise de ameaças permitiu-nos caracterizar o sistema, identificar ameaças, ativos, vulnerabilidades, e no final propor um plano de mitigação. A auditoria de segurança concluiu que pacotes como `org.wso2.carbon.identity` e `org.wso2.carbon.balana` podem precisar de revisões adicionais. Além disso, assumimos que os desenvolvedores devem sempre usar ferramentas de análise estática como um auxílio no processo de desenvolvimento.

Concluímos ainda que o WSO2 IS é uma solução bem pensada; com tudo pronto para ser uma solução IAM extremamente segura, mas isso realmente depende da configuração de implementação.

**Keywords:** Protocol Converter, Identity protocols, Identity Management, Identity and Access Management, Single Sign-On, Threat Analysis, Security Audit, Static Analysis





# Acknowledgements

Two types of acknowledgements are due here.

The first, explicit, goes to my two supervisors, José Magalhães Cruz and Gonçalo Hermenegildo. Thank you for all the assistance provided during this dissertation.

The second, to the professors of the University of Porto, for their teachings and for their patience.

To AET Europe, for all the flexibility and for welcoming me like it was my home.

To BEST Porto, for teaching me everything that the University does not teach, and for giving me skills and experiences that undoubtedly contributed, to not only being a better professional, but also a better person.

To my friends and colleagues, your contributions made me who I am today.

To those who showed me the patience, wisdom, love, and support, I know I did not always deserve.

Last but not least, to my family, for the encouragement and understanding I gained during this time.

Rúben José da Silva Torres



*“It is better to go forward without a goal,  
than to have a goal and stay in one place,  
and it is certainly better than to stay in one place without a goal.”*

Andrzej Sapkowski



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Problem Definition . . . . .	1
1.3	Motivation . . . . .	2
1.4	Goals . . . . .	2
1.5	Structure . . . . .	3
<b>2</b>	<b>Theoretical Fundamentals</b>	<b>5</b>
2.1	Federated Identity Management . . . . .	5
2.1.1	Definition . . . . .	5
2.1.2	The 7 Laws of Identity . . . . .	6
2.1.3	Privacy protection . . . . .	7
2.1.4	Open source IAM . . . . .	7
2.2	Authentication, Authorisation and Accounting (AAA) . . . . .	9
2.2.1	Authentication . . . . .	9
2.2.2	Authorisation . . . . .	10
2.2.3	Accounting . . . . .	10
2.3	Identity Protocols . . . . .	10
2.3.1	SAML 2.0 . . . . .	10
2.3.2	OAuth 2.0 . . . . .	13
2.3.3	OIDC 1.0 . . . . .	14
2.3.4	Comparative Analysis . . . . .	17
2.4	Summary . . . . .	17
<b>3</b>	<b>State-of-the-Art</b>	<b>19</b>
3.1	Identity protocol translation gateway . . . . .	19
3.2	Interconnecting domains with heterogeneous key distribution and authentication protocols . . . . .	21
3.2.1	Mechanism 1: proxies. . . . .	21
3.2.2	Mechanism 2: hiding information in cryptographic expressions . . . . .	21
3.3	IAM solutions . . . . .	22
3.3.1	Shibboleth . . . . .	23
3.3.2	SimpleSAMLPHP . . . . .	23
3.3.3	CAS . . . . .	23
3.3.4	Keycloak . . . . .	24
3.3.5	Gluu server . . . . .	24
3.3.6	Apache Syncope . . . . .	25
3.3.7	MidPoint . . . . .	25

3.3.8	OpenAM and OpenIDM . . . . .	25
3.3.9	Unity IdM . . . . .	26
3.3.10	WSO2 . . . . .	28
3.4	Summary . . . . .	29
<b>4</b>	<b>Problem Statement</b>	<b>31</b>
4.1	Preliminary Study . . . . .	31
4.2	IAM solutions comparison . . . . .	31
4.3	Importance of a Security Analysis . . . . .	31
4.4	Research Methodology . . . . .	32
<b>5</b>	<b>Threat Analysis</b>	<b>33</b>
5.1	Context . . . . .	33
5.2	Methodology . . . . .	34
5.3	Step 1. Description of the system . . . . .	36
5.4	Step 2. Analysis of the technical background . . . . .	48
5.5	Step 3. Identification of assets . . . . .	50
5.6	Step 4. Determination of threats . . . . .	50
5.7	Step 5. Determination of vulnerabilities . . . . .	52
5.8	Step 6. Assets Mapping . . . . .	52
5.9	Step 7. Risk Management . . . . .	55
5.10	Step 8. Mitigation plan . . . . .	55
5.11	Discussion . . . . .	56
<b>6</b>	<b>Security Audit of WSO2 IS</b>	<b>57</b>
6.1	Context . . . . .	57
6.2	Methodology . . . . .	57
6.2.1	Repositories . . . . .	58
6.2.2	Tools . . . . .	61
6.3	Implementation . . . . .	63
6.3.1	FindBugs . . . . .	63
6.3.2	PMD . . . . .	63
6.3.3	SonarQube . . . . .	64
6.4	Results . . . . .	65
6.4.1	FindBugs . . . . .	65
6.4.2	PMD . . . . .	67
6.4.3	Sonarqube . . . . .	68
6.5	Discussion . . . . .	68
<b>7</b>	<b>Conclusions</b>	<b>71</b>
7.1	Contributions . . . . .	72
7.2	Future Work . . . . .	72
<b>A</b>	<b>Appendix</b>	<b>73</b>
A.1	WSO2 IS Extension repositories . . . . .	73
	<b>References</b>	<b>79</b>

# List of Figures

2.1	Relationship between SAML Components [35]	11
2.2	A simple SAML Use Case [40]	12
2.3	A simple OAuth Use Case [40]	14
2.4	A simple OIDC Use Case [40]	16
3.1	Identity protocol translation gateway - translation method [42]	20
3.2	Mechanism 2 [41]	22
3.3	Unity Features [22]	27
3.4	Unity fundamentals [22]	28
5.1	Steps of Threat analysis	34
5.2	Use case diagram - User management boundary system	37
5.3	Use case diagram - Authentication boundary system	38
5.4	Use case diagram - Access delegation and policies	39
5.5	Use case diagram - Administration boundary system	39
5.6	WSO2 IS architecture overview and process flow [23]	49
6.1	FindBugs number of reports per priority	66
6.2	FindBugs number of reports per Rank	66
6.3	PMD number of reports per priority	67
6.4	SonaQube results summarised	68





# List of Tables

2.1	Comparative Analysis between SAML, OAuth and OIDC . . . . .	17
5.1	Use case table - Managing Users and Roles . . . . .	40
5.2	Use case table - Admin-Initiated User Registration . . . . .	41
5.3	Use case table - Self-Registration . . . . .	42
5.4	Use case table - Password Patterns . . . . .	42
5.5	Use case table - Password Reset . . . . .	43
5.6	Use case table - User Name Recovery . . . . .	43
5.7	Use case table - Multi-tenancy . . . . .	44
5.8	Use case table - Account Locking . . . . .	44
5.9	Use case table - Single Sign-On . . . . .	45
5.10	Use case table - Multi-factor authentication . . . . .	46
5.11	Use case table - Federated Authentication . . . . .	47
5.12	Use case table - Access Delegation . . . . .	47
5.13	Use case table - Fine-Grained Access Control with XACML . . . . .	48
5.14	WSO2 IS Assets . . . . .	50
5.15	Threats . . . . .	52
5.16	Vulnerabilities . . . . .	53
5.17	Assets Mapping . . . . .	54
5.18	Threats associated with risk . . . . .	55
6.1	Major WSO2 IS repositories . . . . .	58
6.2	Packages found to contain problems . . . . .	65
6.3	Findbugs results per package - part 1 . . . . .	65
6.4	Findbugs results per package - part 2 . . . . .	66
6.5	PMD top 10 packages with the most bugs . . . . .	67





# Abbreviations

2FA	Two-factor authentication
AAA	Authentication, Authorisation, Accounting
AE	Authorization Endpoint
API	Application Programming Interface
AS	Authorization Server
CAS	Central Authentication Service
CSRF	Cross-site request forgery
DNS	Domain Name System
DoS	Denial-of-Service
ECP	Enhanced Client or Proxy
EU	End User
FIdM	Federated Identity Management
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IdM	Identity Management
IdP	Identity Provider
IWA	Integrated Windows Authentication
JAAS	Java Authentication and Authorization Service
JSON	JavaScript Object Notation
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
MITM	Man-In-The-Middle
OC	OAuth Client
OECD	Organization for Economic Cooperation and Development
OIDC	OpenID Connect
OS	Operating System
PAM	Pluggable Authentication Modules
PAOS	Reverse SOAP
RADIUS	Remote Authentication Dial-In User Service
REST	Representational State Transfer
RO	Resource Owner
RP	Relying Party
RS	Resource Server
SAML	Security Assertion Markup Language
SCIM	System for Cross-domain Identity Management
SOAP	Simple Object Access Protocol
SP	Service Provider
SSO	Single sign-on
TE	Token Endpoint

U2F	FIDO Universal 2nd Factor
UIE	UserInfo Endpoint
UMA	User Managed Access
UML	Unified Modeling Language
UNICORE	Uniform Interface to Computing Resources
URI	Uniform Resource Identifier
WS-Federation	Web Services Federation
WSO2 IS	WSO2 Identity Server
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language
XSS	Cross-Site Scripting



# Chapter 1

## Introduction

### 1.1 Context

The enterprise AET Europe, an enterprise offering digital security solutions, proposed this dissertation to solve a common problem in organisations, being the development of this work made in a business environment.

For two systems to be able to interact with each other, they need to understand each other first. With this in mind, computer systems usually follow a protocol implementation to be able to communicate and exchange information. Protocols can be associated with anything, and there is a need for some protocols to provide the identity information of the entity that initiated the communication. The instant that two or more systems use distinct protocols for communication, they inherently process identity information differently, usually making them inoperable and unable to communicate.

### 1.2 Problem Definition

The vast diversity of authentication protocols and multiple identity alternatives aggravates this problem. Identity and Access Management (IAM) protocols are designed specifically for the transfer of authentication information and consist of a series of messages in a preset sequence intended to protect data as it traverses through networks. Several IAM protocols exist to support strong IAM policies by securing data and ensuring its integrity during transfer. Generally known as “Authentication, Authorisation, Accounting” or AAA, these identity management protocols provide standards for security to simplify access management, assistance in compliance, and create a uniform system for handling interactions between users and systems.

Different standard identity communication protocols were designed to solve this problem. We will primarily discuss single-sign-on protocols. Single sign-on (SSO) is a property of access control of multiple related yet independent software systems. However, even though SSO protocols alleviate this problem, the different SSO protocols differ from each other. For example, SAML and

OIDC are two different SSO protocols that enable single sign-on across web applications; however, when the OIDC system attempts communication with a SAML system, they cannot exchange information. This variance means that the protocols are not interoperable, and an organisation that uses a specific protocol for SSO, often cannot provide this functionality to another organisation with a different protocol. We can conclude from this example that a large number of computer systems cannot communicate with each other because they are implemented with different SSO protocols. Consequently, they are incapable of managing identity information from the users from each other's systems.

Furthermore, different protocols used in the industry often have different incompatible versions of themselves. For example, an organisation may implement OAuth version 2.0, but this version will not be able to interoperate with OAuth version 1.0.

These discrepancies between protocols result in significant time and money investment for an organisation to implement every single identity protocol in all of its technological infrastructures. The increase of advances to user and session identity management has made it very difficult to choose the technology in a way that facilitates collaboration among enterprises. In a situation involving two or more organisations, this problem culminates in one of the organisations making a meaningful investment to adjust their infrastructure to provide/consume the identity information the way their partners expect.

This problem has been tackled in many IAM solutions [30],[41], [42], and [51].

### 1.3 Motivation

Techniques for efficiently handling multi-protocol authentication and authorisation in computer network environments would be advantageous in order to reduce costs and time investment, as would be techniques that take advantage of multi-protocol requests to provide new authentication and authorisation paradigms.

As stated in [41] and [42], the need arises to interconnect heterogeneous security domains. However, devising mechanisms to interconnect heterogeneous distributed systems is a challenging assignment, and if the heterogeneity also influences the security of the system, the interconnection problem is even harder.

### 1.4 Goals

The objective of this dissertation is to find the best way for organisations to overcome the inability to collaborate due to identity protocol inoperability. The idea is to implement or find the best solution possible so that various identity protocols can interact seamlessly without implementing every single protocol themselves, saving a lot of time and money. After, and since this is a sensitive topic in terms of security, this solution should be analysed in-depth for possible security deficiencies. WSO2 Identity Server was the adopted open-source solution, and, to contribute to



their project from a security standpoint, we developed a threat analysis and a security audit using static analysis tools.

Who does this interest? All service providers who wish to expand their support base to other forms of identity protocols and increase compatibility with other services and new market solutions that want to support multiple forms of authentication quickly.

## 1.5 Structure

This document is divided into six other chapters. Chapter 2 will provide sufficient theoretical fundamentals for the rest of the dissertation. This includes the concept of Federated Identity Management (FIdM), Authentication, Authorisation and Accounting (AAA), and a brief explanation of some identity protocols such as SAML 2.0, OAuth 2.0, and OIDC 1.0.

Chapter 3 presents a literature review of identity protocol converters and the current state-of-the-art of IAM solutions.

Chapter 4 builds upon Chapter 3 and delineates the rest of the work dissertation moving forward. In this chapter, we justify why the WSO2 IS was chosen as a solution to our problem, why the idea of an identity protocol converter might be unfeasible and why we chose to make a security analysis to WSO2 IS.

Chapter 5 corresponds to a threat analysis made to a WSO2 IS, and ends with a mitigation plan that should be considered in any WSO2 IS deployment.

Chapter 6 corresponds to a security audit made to the WSO2 IS, using open-source static analysis tools.

Finally, Chapter 7 focuses on summarising the work of this dissertation and providing a unification of the main ideas presented and conclusions drawn.



## Chapter 2

# Theoretical Fundamentals

### 2.1 Federated Identity Management

This section provides an introduction to Federated Identity Management, FIdM.

FIdM promotes the management of identity processes and policies among the collaborating entities without centralised control.

FIdM has many issues to consider, some of them contradictory to each other, which causes this not to be a wholly defined topic but one that will continue in active research for many years to come. These issues include ease of use, user privacy, security, single sign-on, the total cost of ownership, user profiling and retention of users by service providers, scalability, fine-grained access control, personalisation of services, and anonymity.

In this section, we will review what Federated Identity Management is, what are the requirements for a proper Identity and Access Management (IAM) solution and will give an overview of open-source software and its advantages in IAM.

#### 2.1.1 Definition

Identity Management is defined by [46] as a set of functions and capabilities used for:

- assurance of identity information (e.g., identifiers, credentials).
- assurance of the identity of an entity (e.g., users/subscribers, groups, user devices).
- enabling businesses to operate with secure applications.

Federation is defined by [45] as an association comprising any number of service providers and identity providers. What is implicit in this definition is the mutual trust of the providers included in this association and the willingness of them to inter-communicate. Usually, this type of communication includes authentication and authorisation information of users so they can access

resources in a different service provider, that is the concept of Federated Identity Management (FIdM).

FIdM promotes cooperation between enterprises where trust between the Service Provider (SP), and Identity Provider (IdP) is essential for authentication or authorisation. An IdP is a third-party service which manages identities for an SP providing a service.

Single Sign-On (SSO) is a protocol for authentication which allows user to log in with their credentials once and enter a federated environment to access different services without the need of having multiple credentials. An excellent example of SSO is Google, where logging in with a Google Account allows a user to use different services like Gmail, Google Drive, Youtube.

When correctly placed FIdM provides [28]:

- SSO capabilities, meaning users can use various service without having to authenticate more than once.
- reduced cost for service providers since they do not need to manage identity information.
- a more scalable solution considering the increase in the number of users does not impact the management of identity

### 2.1.2 The 7 Laws of Identity

Seven essential laws that explain the successes and failures of digital identity systems are defined in [27].

1. «**User Control and Consent** - Technical identity systems must only reveal information identifying a user with the user's consent.» [27]. The implied premise is that users will stop to trust a system that reveals their identity information to others without explicit consent. Users will only utilise a service if they have the confidence that their identity information is protected and their wishes on how this information should be used are respected.
2. «**Minimal Disclosure for a Constrained Use** - The solution which discloses the least amount of identifying information and best limits its use is the most stable long term solution.» [27]. The implied premise is that all systems have security flaws, meaning they are vulnerable to attacks and theft of information. Therefore, systems that capture less identity information and delete it after completing the purpose of the capture are more trustworthy.
3. «**Justifiable Parties** - Digital identity systems must be designed, so the disclosure of identifying information is limited to parties having a necessary and justifiable place in a given identity relationship.» [27]. The implied premise is that user information should only be used in a transaction between systems when it is essential for its operation.
4. «**Directed Identity** - A universal identity system must support both omnidirectional identifiers for use by public entities and unidirectional identifiers for use by private entities,

thus facilitating discovery while preventing unnecessary release of correlation random identifiers.» [27]. The implied premise is that users do not want their identifiers shared with service providers. However, some service providers to be able to contact the users, need their identifiers. In order to solve this, when users establish communication with service providers, their identifier should be private or a one time use identifier. This type of identifiers prevents service providers from joining information and creating a global user profile.

5. «**Pluralism of Operators and Technologies** - A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers.» [27]. The implied premise is that variety and trust relations between identity providers are excellent, and users should have the ability to change between pseudonymous identities whenever they need.
6. «**Human Integration** - The universal identity metasystem must define the human user to be a component of the distributed system integrated through unambiguous human-machine communication mechanisms offering protection against identity attacks.» [27]. The implied premise is that the user is part of the system's communication and should be considered as the weakest link in it. Therefore, securing this link is essential for any identity management system.
7. «**Consistent Experience Across Contexts** - The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies.» [27]. The implied premise is that the user needs to have a consistent experience regardless of the technologies, scenarios involved in the data exchange.

These seven laws should not be considered something to live by; nevertheless, they are laws identity management systems should follow since users are likely to reject or reduce trust levels at any system that does not implement them.

### 2.1.3 Privacy protection

Identifiable by seven laws of identity management, privacy protection is an issue that FIdM systems need to take into account. Many countries have their legal requirements, but these all derive from the Organization for Economic Cooperation and Development (OECD) privacy guidelines [11]. FIdM systems need to implement these principles. One approach to implement these principles is to separate the identity providers from the service providers and store identity information only in Identity providers. As we will see in Chapter 3 Section 3.3, all modern IAM systems are capable of this and designed with this in mind.

### 2.1.4 Open source IAM

In recent years, Identity and Access Management (IAM) solutions started to receive more attention. With the appearance of more open-source solutions, choosing an IAM solution just based on

the number of features is no longer a viable strategy, since most offers are similar in functionality [2], [3], [6], [9], [10], [12], [13],[19], [20], [22], [23].

Enterprises should consider open-source IAM solutions since these offer an array of unique proposition inherent to open-source software. Open-source software is more receptive to open standards and has a more significant focus on extendability at a lower cost and freedom. Besides, enterprises can evaluate open-source software or create a proof of concept without waiting for bureaucracies. The freedom to use the software without locking to a solution is also a considerable advantage. Open-source software architectures are designed to have extensibility in mind to allow changing requirements and different customisations [21].

The advantages of open-source software match what the IAM experience should be, customisable to offer a unique digital experience among competitors with a lower cost. However, using open-source software for an enterprise also has risks associated with the lifespan of the project. When an enterprise chooses an open-source solution for its project, this project becomes reliant on the community behind the open-source project. The contributors of the open-source have in-depth knowledge of the software; however, they are not obligated to support it or to help users. This poses a considerable risk to an enterprise who chooses open-source software but can be mitigated by hiring some of the contributors or training the existent work-force. Furthermore, open-source projects tend to stall when the lead contributor leaves the project or the community, in general, loses interest [21]. We additionally have to consider that commercial solutions give a warranty and legal paid support for the solutions, and that is part of their task to identify risks and act in conformance.

There is also the misconception that open-source software is more prone to vulnerability attacks because anyone can look at the source code. However, this should be considered as an advantage; open-source usually have more contributors to the source code, which means more people looking at it and finding problems. There are also many open-source security tools available which help in finding vulnerabilities in the development phase [31].

#### **2.1.4.1 What to Look for When Choosing Open Source IAM?**

As mentioned, the number of features is no longer a viable strategy for choosing an IAM solution, but some requirements need to be met depending on the use-case [21]:

- The chosen IAM solution should have the best coverage of features required for the project's functional requirements.
- The selected software should be completely open-source and not just open-core.
- The IAM solution should be easily extendable or customisable, giving it a different user experience from competitors.
- The IAM solution should be easily integrated into the current technology stack since it will act as an entry point to all services

- The IAM solution can easily incorporate new algorithms and security protocols when they emerge. If the community behind the project is active, this leads to a fast product update with the latest security standards.
- The IAM solution needs to comply with security standards and regulations.
- The chosen IAM solution needs to support open-standards to minimise vendor-locking.
- The community behind the project should be large enough and active to prevent the project from stalling and obtain better and faster support.

## 2.2 Authentication, Authorisation and Accounting (AAA)

Authentication, Authorisation and Accounting (AAA) principles provide a solution for issues related to user access to services or systems and how can they use them [24]. When a user is accessing a service from a particular system, there are three issues to resolve between them:

- The user needs to input its credentials (Authentication).
- The system must decide whether a user can access the system and what are the resources available (Authorisation).
- All activities must be logged (Accounting).

### 2.2.1 Authentication

Authentication is the process of identifying an individual. After authentication, the user can access network resources based on the user's authorisation. Each user needs a set of criteria to access his allowed resources, in the process of authentication, the AAA system compares the user's authentication credentials to stored credentials in the system's database. If the credentials match, the network grants access to the user, if they do not match the authentication fails. The European Central Bank defines strong customer authentication based on the use of two or more of the three types of factors, categorised as knowledge (e.g., password, id), ownership (e.g., smart card, mobile phone) and inherence (e.g., fingerprint) [32]. For strong authentication, these factors must be mutually independent so that the breach of one of them does not compromise the others. Besides, at least one of the used factors must be a non-reusable factor and not easily stolen. A user can authenticate into a system using one or more of the following methods, sorted from a lower to a higher level of security:

- Single-factor authentication - only uses one element out of the three-factor categories. One single factor, in the case of IAM solutions, does not provide sufficient protection against malicious intrusion and misuse.
- Two-factor authentication - also known as 2FA, the user's identity is confirmed by using a combination of two independent components from two different factor categories.

- Multi-factor authentication - is similar to 2FA, but it can combine more than two authentication factors for enhanced security.
- Strong authentication – similar to multi-factor authentication but requires the use of at least one non-reusable and non-replicable factor.

### 2.2.2 Authorisation

Authorisation is a process of giving individuals access to system objects based on their identity. After authentication succeeds, the authorisation process determines whether the user has the authority to use some or all of the system's activities, resources or services. Authorisation occurs within the same context as of authentication. Once a system authenticates a user, they may be authorised to different types of access or activity.

### 2.2.3 Accounting

Accounting, also known as auditing, is a process of logging user's activity while accessing the network resources, including the amount of time spent in the network, the services accessed while there, and the amount of data transferred during the session. Accounting is essential for all systems because it helps to keep the system secure by logging suspicious changes in user activity and facilitating the analysis of these changes.

## 2.3 Identity Protocols

This section explains three of the most popular FIdM standards [29], SAML, OAuth, and OpenID Connect (OIDC) since they cover practically the entire FIdM industry. We will review SAML 2.0, OAuth 2.0, and OIDC 1.0.

### 2.3.1 SAML 2.0

The Security Assertion Markup Language (SAML) [35] defines a XML framework for exchanging security assertions between entities. It permits the exchange of authentication and authorisation information. SAML shares identity information using SAML assertion/message payload expressed in XML. SAML has three key-roles in any transaction:

- Identity Provider (IdP)
- Service Provider (SP)
- User

The IdP is the trusted organisations responsible for authentication and authorisation. The SP is an organisation that provides services and relies on the IdP for authentication or authorisation.



The user is the entity that initiates the communication; this can be an application program that requests the use of service available in the SP.

SAML 2.0 [35] defines the following components:

- **Assertions** - describe how the identities are represented.
- **Protocols** - is a sequence of XML messages.
- **Bindings** - describe how XML messages are transported (eg. HTTP, SOAP)
- **Profiles** - define how the SAML assertions, protocols, and bindings are combined for interoperability in particular usage scenarios.

For more information on these components, refer to [35]. The relationship of these components can be seen in Figure 2.1.

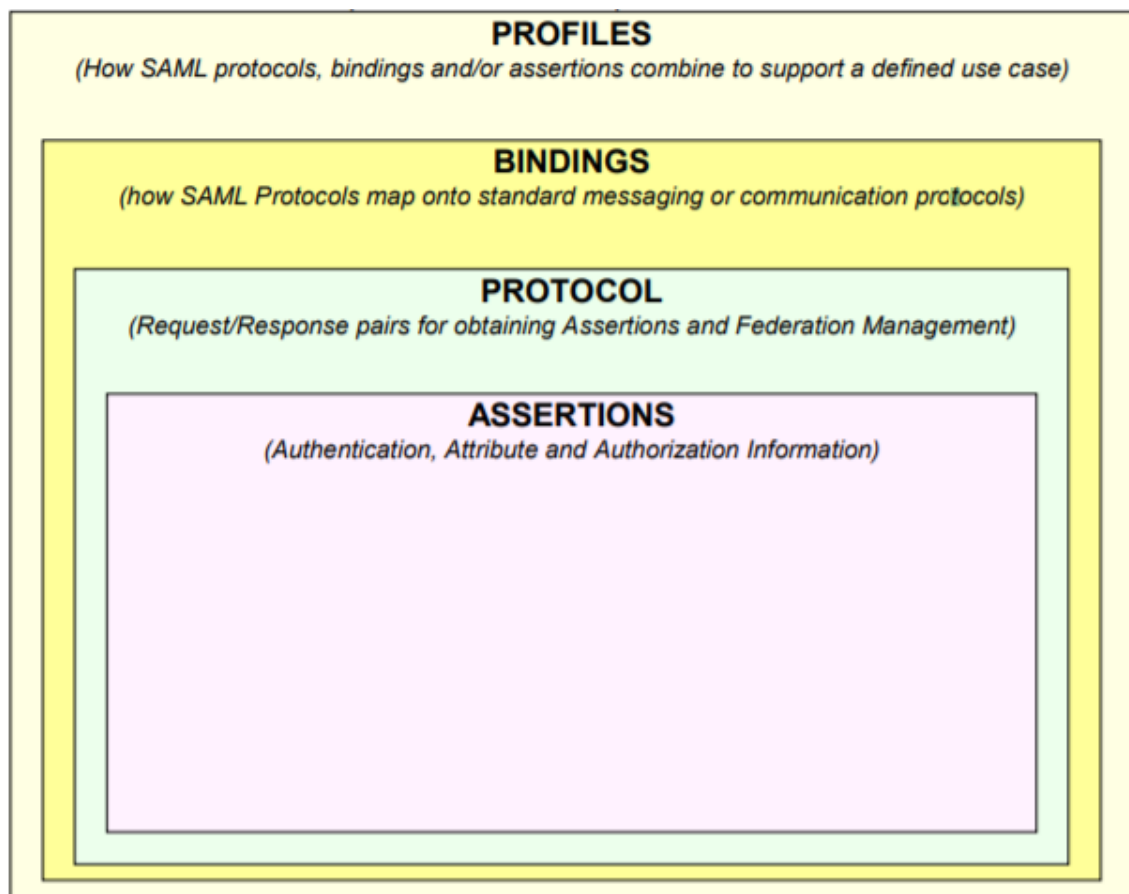


Figure 2.1: Relationship between SAML Components [35]

A simple use case of SAML for identification can be seen in Figure 2.2. Each step of communication is described below:

1. user tries to access a service from the SP

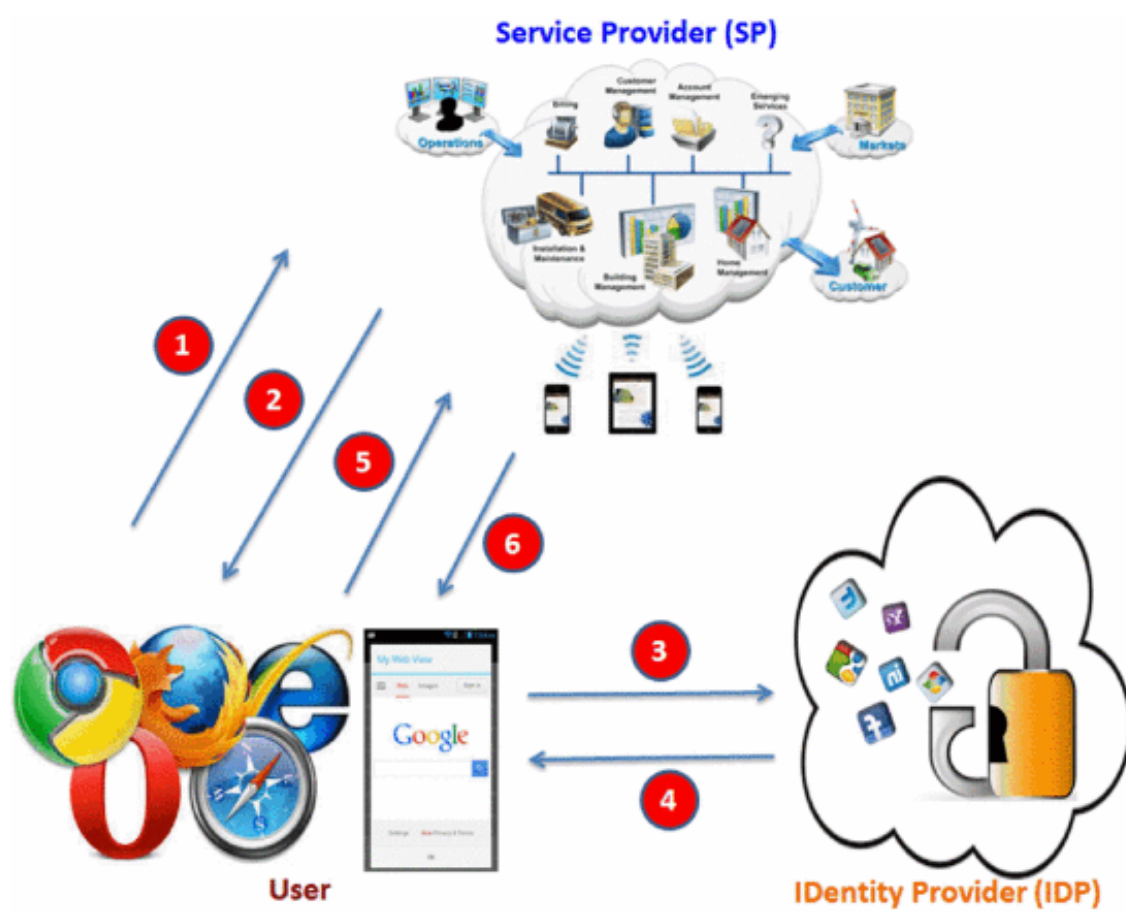


Figure 2.2: A simple SAML Use Case [40]

2. SP creates a SAML request and sends it to the user
3. The SAML Request gets redirected to the IdP
4. The IdP authenticates the user and generates a SAML response sending it to the user
5. The SAML Response is sent to the SP
6. The SP validates the SAML response, making the user have access to service initially wanted.

### 2.3.2 OAuth 2.0

OAuth [33] can be considered a delegation protocol; it allows the user to grant access to an application to perform authorised tasks on behalf of the user. In other words, OAuth allows third-party applications to gain access to a service on behalf of the User/Resource Owner. OAuth introduces an authorisation layer and separates the role of the client from that of the resource owner. The client requests access to the use of resources controlled by the resource owner and accessible on the resource server. For this, the client never has access to the resource owner's credentials but obtains an access token issued by an authorisation server after the approval of the resource owner. This token is sufficient for the client to get access to the protected resources on the resource server.

By this simple explanation, we can observe that OAuth defines four roles:

- Resource Owner(RO)
- OAuth Client (OC)
- Resource Server (RS)
- Authorisation Server (AS)

The OAuth Provider is just the provider that supplies the OAuth service and is not defined on the OAuth Specification [33].

OAuth 2.0 is a highly extensible framework with a large number of optional components, making this specification prone to produce many different implementations that may be non-interoperable between themselves [33]. This makes OAuth 2.0 the perfect candidate for our problem and one of the most beneficiaries of any solution.

A simple use case of OAuth 2.0 can be seen in Figure 2.3, and each step of the communication is described below:

1. RO using an application needs access to resources from a different organisation
2. OC makes a request to the AS for a Request token and Secret Key
3. AS issues the Request Token and Secret Key and sends it to the OC
4. OC sends an URL containing the Request Token for the user to authorise

5. RO clicks on the URL
6. AS asks the RO to authorise the OC
7. RO authorises the OC
8. After the RO authorisation, an access token is created by the AS, with the previously generated Secret key, and sent to the OC
9. The OC sends the access token to Resource Server and acquires the needed resources.
10. RS sends the resources to the OC
11. RO can now use the resources on the OC

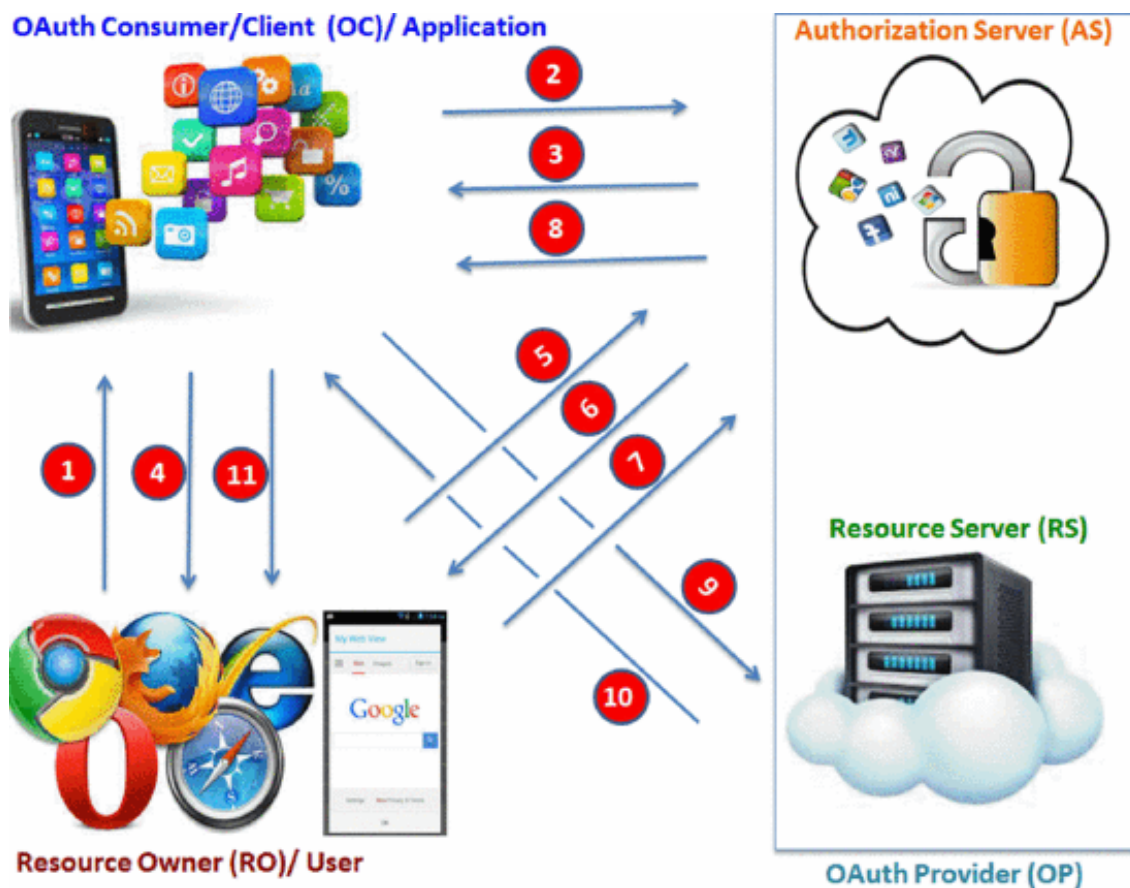


Figure 2.3: A simple OAuth Use Case [40]

### 2.3.3 OIDC 1.0

OpenID Connect (OIDC) [44] is specification built as a profile of OAuth 2.0 rather than a distinct utterly new protocol. This specification is a framework that allows sending digital identity via

RESTful APIs. Even though OIDC is based on OAuth, it is seen as an evolution to OpenID 2.0. The primary difference between OIDC and OAuth 2.0 is that it enables End-Users to be Authenticated with an ID Token. The ID Token has additional information about the authenticated user. This token is signed by the IdP and can be read and verified without communicating with the IdP.

Even though OIDC offers some flexibility in its implementation, it has a lot of more standardised parameters that were left up to developers to decide in OAuth (such as for instance scopes, endpoints discovery, and dynamic registration of clients).

The OIDC defines five key roles:

- Relying Party (RP)
- End User (EU)
- Authorization Endpoint (AE)
- Token Endpoint (TE)
- UserInfo Endpoint (UIE)

These roles will not be furthermore explained in this document since they are very similar to the OAuth 2.0 roles [33]. Refer back to the OIDC specification [44] for more information on these roles.

A simple use case of OIDC 1.0 can be seen in Figure 2.4, and each step of the communication is described below:

1. EU sends their OpenID login details to the RP
2. RP forwards this login details to the AE
3. The AE authenticates the user after verifying the login details
4. EU is logged in
5. The EU sends the authorisation code to the RP
6. RP sends to the TE the authorisation code and some additional secret information
7. TE responds with the ID Token and Access Token
8. RP validates the ID Token
9. RP sends the Access token to the UIE
10. UIE sends the information about the user to the RP
11. RP can now deliver services to the EU

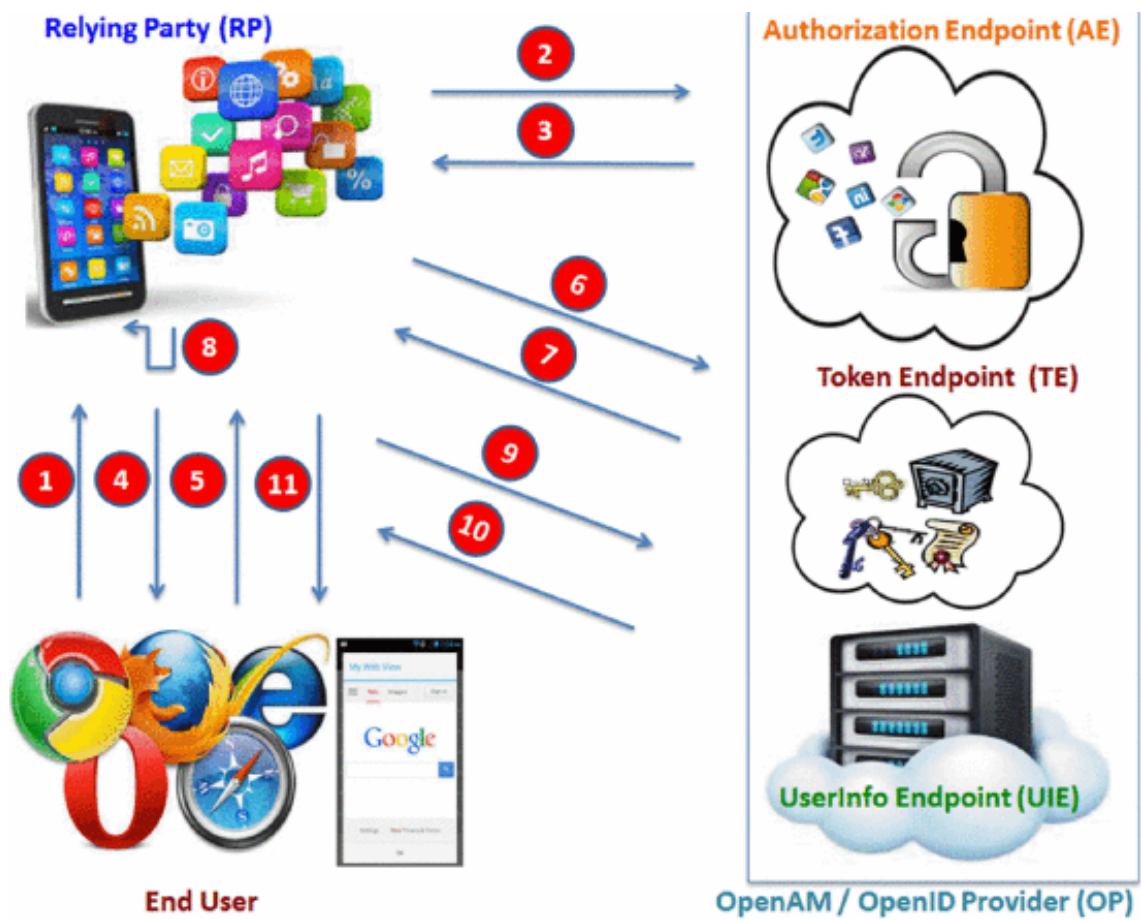


Figure 2.4: A simple OIDC Use Case [40]

### 2.3.4 Comparative Analysis

The paper *Securing digital identities in the cloud by selecting an apposite Federated Identity Management from SAML, OAuth and OpenID Connect* [40] shows the pros and cons of each protocol.

The Table 2.1 is a summary from the information on [40].

Table 2.1: Comparative Analysis between SAML, OAuth and OIDC

Criteria	SAML 2.0	OAuth 2.0	OIDC 1.0
Main Usages	FIdM, SSO	API authorization	FIdM, SSO
Authentication / Authorization	Y/Y	N/Y	Y/Y
Token format	XML	XML,JSON,JWT	JSON,JWT
Token content	User identity but no credentials	User identity but no credentials	User identity but no credentials
Protocols used	XML, HTTP, SOAP	JSON, HTTP, REST	JSON, HTTP, REST
User consent	Not responsible	Collects before sharing attributes	Collects before sharing attributes
Client Discovery and On-Boarding	No dynamic introductions	No dynamic introductions	Dynamic introductions
Data integrity	XML signature - X.509	Token contents can be protected with DS or a MAC	JWS - HMAC SHA-256
Web and native mobile app support	Web	Both	Both

## 2.4 Summary

In this chapter, we established needed background knowledge to understand the following chapters better. This knowledge includes understanding what Identity and Access Management (IAM), Federated Identity Management (FIdM) and what makes the right IAM solution. We established the essential and somewhat confusing notion of Authentication, Authorisation and Accounting (AAA), needed for a basic understanding of some identity protocols used in FIdM.





## Chapter 3

# State-of-the-Art

In this chapter, we will analyse available solutions for the proposed problem. *Identity protocol translation gateway* patent [42], the *Interconnecting domains with heterogeneous key distribution and authentication protocols* paper [41], and some *Open Frameworks and toolkits* will all be evaluated and considered for the possibility of using them on this dissertation. We will place a higher focus on Unity IdM [22], and WSO2 IS [23].

### 3.1 Identity protocol translation gateway

Identity protocol translation gateway [42] is a patent for a product that promises one or more methods for translating identity protocols and a system implementing such methods. Figure 3.1 depicts the method for translating identity protocols in this patent. To facilitate interactions between organisations and solve the problem mentioned in Chapter 1, this patent was created promising an identity translation protocol gateway. The method mentioned can be divided into the following steps:

1. A configured the gateway communicates with a first and second identity protocol. Being these different from each other.
2. The first communication between a first computer system and the gateway using the first identity protocol, with the first identity protocol having identity information about one or more entities associated with that request.
3. This identity information is then translated into a canonical representation. This canonical representation can have additional information obtained by the gateway.
4. From the canonical representation, the identity information is then translated to a second identity protocol.

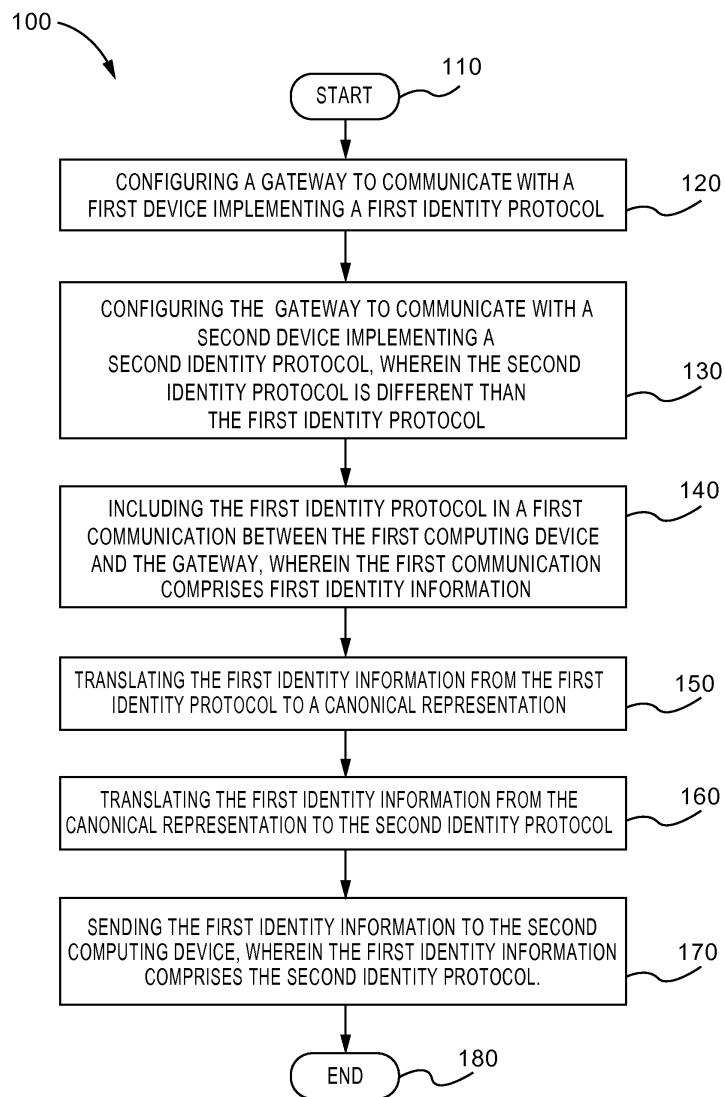


Figure 3.1: Identity protocol translation gateway - translation method [42]

5. The identity information of the first communication is sent to the second computer system, using the second identity protocol created from the canonical representation.

This patent, however, does not describe any technical information, there is no product available, and could not correspond to a real developed technological product. There is no information on how information is transformed into a canonical representation and vice-versa. The canonical representation is not defined in the document. However, the idea as a whole can be harnessed.

## **3.2 Interconnecting domains with heterogeneous key distribution and authentication protocols**

The paper [41] describes two mechanisms for designing a protocol converter for authentication and key distribution protocols; however, they only consider symmetric encryption and the meaning of certificate in this paper does not include current digital certificates. The first mechanism is based on proxies and a synchronisation protocol. The second mechanism addresses the problem of the statefulness of the protocol converter. Both can be used in separate or in combination. As said, in this paper, “When properly combined, they provide for a robust, transparent, and safe protocol converter for authentication and key distribution protocols.” [41]. These interconnection mechanisms do not interfere with the security of the systems involved, since it has no access to these systems secrets, making it safer. However, designing this type of system is always a difficult task, mainly if it affects the security architecture of the system. Using a protocol converter to translate cryptographic tokens for security protocols safely, is in itself a contradiction because the protocol converter needs to be involved in the conversion of secure end-to-end flows.

### **3.2.1 Mechanism 1: proxies.**

The proxies task can be summarised as follows:

- They have to impersonate the principals they represent.
- They have to notify the other network with everything that happens in its network, using a synchronisation protocol.
- They have to listen to messages from the other network to know the events that have taken place there. They have to initiate the same events in their own network.

### **3.2.2 Mechanism 2: hiding information in cryptographic expressions**

This mechanism is based on the following observation. The certificate for the server that the client receives is encrypted under the server’s master key. Since the client does not have this key, the encrypted part of the certificate is not understood by it. Taking this into account, it is possible to add additional information to the certificate without the client noticing. This information can be

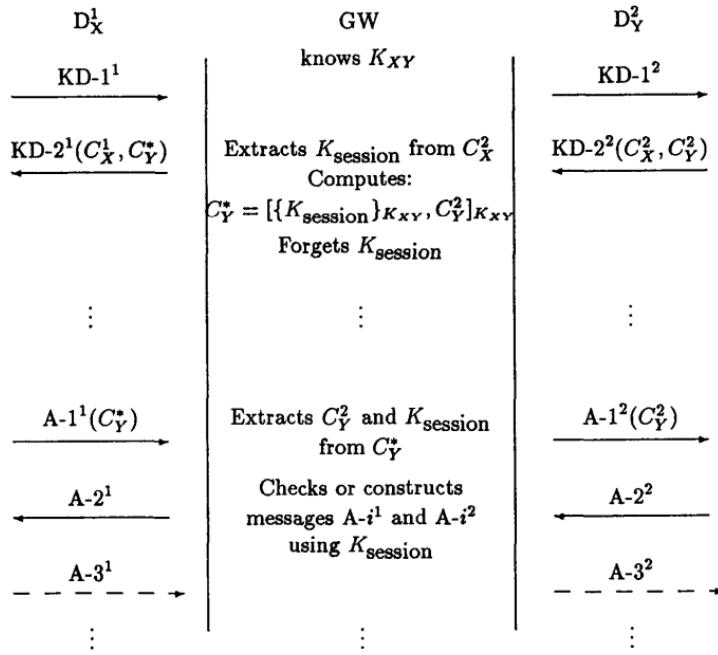


Figure 3.2: Mechanism 2 [41]

extracted in the future when the client tries to initiate an authentication session. Being used to help in the translation of the messages.

Put differently; the certificate is used as storage as is seen in Figure 3.2. Throughout the key distribution phase, the gateway saves in the certificate the session key and any other relevant information. This certificate is then sent to the client who cannot differentiate it from an unaltered one. During the authentication phase, the gateway extracts the information saved on the certificate and sends it to the server, making the certificate the same as the original one.

For more information on these mechanisms, their limitations, implementation issues, and design choices refer to [41].

### 3.3 IAM solutions

There are many IAM solutions on the market; most of them are commercial ones such as Okta Workforce [18], RSA SecurID Suite [8], Oracle Identity Governance [14], and IBM Security Identity and Access Manager [7]. For reasons presented on Chapter 2 we will only explore open-source IAM solutions and one interesting open-core solution.

### 3.3.1 Shibboleth

Shibboleth [37] [19] is an open-source SAML implementation.

The purpose of Shibboleth started as a Single Sign-On (SSO) and identity federation solution for academic institutions. However, currently, it is developing into a general FIdM solution that can be adapted to organisations needs. It supports authentication, authorisation, content personalisation, and enables single sign-on across a wide range of services from several providers. Shibboleth consists of several individual components: IdP, SP, and discovery service (DS), which may be deployed separately depending on the specific needs of the organisation.

To handle the SSO usage scenario, Shibboleth specifies two SSO profiles for Web browsers, which are based on corresponding SAML profiles. Shibboleth also provides built-in support for a pre-defined attribute schema called eduPerson based on the Lightweight Directory Application Protocol (LDAP) standard which makes it possible to integrate Shibboleth into existing LDAP directory services.

### 3.3.2 SimpleSAMLPHP

SimpleSAMLphp [20] is an open-source native PHP application with a focus primarily on dealing with SAML 2.0 as a Service Provider (SP) and SAML 2.0 as an Identity Provider (IdP). However, it also supports some other identity protocols and frameworks, and the development of new modules is simple since this framework is easily extendable.

### 3.3.3 CAS

Central Authentication Service project (CAS) [3] is a unique SSO solution for the web and attempts to be a complete platform for authentication and authorisation. CAS also refers to an open authentication protocol. CAS project has support for the CAS protocol and additional identity protocols and features, with everything being an open-source solution.

The following items include a summary of the features and technologies offered by the CAS project:

- pluggable authentication support such as LDAP, Database, X.509, SPNEGO, JAAS, JWT, RADIUS, MongoDB.
- support for multiple identity protocols such as CAS, SAML, WS-Federation, OAuth2, OpenID, OpenID Connect.
- support for multi-factor authentication via a variety of providers.
- support for delegated authentication to external providers such as ADFS, Facebook, Twitter, SAML2 IdPs.
- built-in support for password management, notifications, terms of use and impersonation.
- support for attribute release, including user consent.

- monitor and track application behaviour, statistics and logs in real-time.
- manage and register client applications and services with specific authentication policies.
- integration with InCommon, Box, Office365, ServiceNow, Salesforce, Workday, WebAdvisor, Drupal, Blackboard, Moodle, Google Apps.

### 3.3.4 Keycloak

Keycloak [9] is also a complete open-source IAM solution for modern applications and services.

The following items include a summary of the features and technologies offered by the Keycloak:

- support for multiple identity protocols such as OpenID Connect, OAuth 2.0 and SAML 2.0.
- centralised management for admins and users
- connection to existing user directories with LDAP and Active Directory
- support for delegated authentication to external providers with OpenID Connect or SAML 2.0 IdPs
- extensible customisation through coding
- password Policies and their customisation
- support for fine-grained authorisation policies and combines different access control mechanisms

### 3.3.5 Gluu server

The Gluu Server [6] is a container distribution of open-source software for identity and access management (IAM). Gluu Server can be used for user authentication, identity information, and policy decisions.

The Gluu Server supports the use of the following open standards for authentication, authorization, federated identity, and identity management:

- OAuth 2.0
- OpenID Connect
- User Managed Access 2.0 (UMA)
- SAML 2.0
- System for Cross-domain Identity Management (SCIM)
- FIDO Universal 2nd Factor (U2F)

- FIDO 2.0 / WebAuthn
- Lightweight Directory Access Protocol (LDAP)
- Remote Authentication Dial-In User Service (RADIUS)

### 3.3.6 Apache Syncope

Apache Syncope [2] is an open-source solution for managing digital identities in enterprise environments.

The following items include a summary of the features and technologies offered by the Apache Syncope:

- connection to existing user directories such as LDAP and Active Directory, flat files (e.g. XML, CSV) and relational databases.
- support for provisioning to keep identity data synchronised
- user self-service features
- support for various IAM protocols such as OAuth, XACML, SAML and OpenID Connect.
- Web-Based Administrative User Interface
- logging, auditing, and reporting

### 3.3.7 MidPoint

MidPoint [10] is an open-source identity, and organization management and governance platform. MidPoint covers both technological and business requirements of the organisation. However, it does not cover the access management and identity federation world. That means that midPoint does not implement the “server side” of OAuth2, OpenID Connector or SAML protocols since this solution is not supposed to be an identity provider, authentication server or SSO server. Nonetheless, midPoint can support those protocols indirectly integrating them with another solution creating an identity and access management (IAM) solution.

### 3.3.8 OpenAM and OpenIDM

OpenAM [12] and OpenIDM [13] are two separated open-core solutions for Identity and Access Management. Open-core meaning they are the core solution for a more prominent and commercial solution, in this case, they are the open-core community editions for ForgeRock Access Management and ForgeRock Identity Management respectively.

OpenAM is a solution used for managing users, roles, and access to resources. The following items include a summary of the features and technologies offered by the OpenAM:

- authentication with Java Authentication and Authorization Service (JAAS) and Integrated Windows Authentication and authorisation policies
- adaptive risk authentication to assess risks during the authentication process
- Federation services using standard identity protocols SAML, WS-Federation, OpenID Connect, Fedlet, OAuth2, and OpenIG Federation Gateway. Can act as a multi-protocol hub, translating for providers who rely on other, older standards.
- system failover and session failover. These two key features further ensure that no single point of failure exists

OpenIDM is a solution used to simplify the management of identity, as it can help synchronise data across multiple resources. Each organisation can maintain control of accounts within their respective domains.

The following items include a summary of the features and technologies offered by the OpenIDM:

- Web-Based Administrative User Interface
- user self-service features can streamline onboarding, account certification, new user registration, username recovery, and password reset.
- role-based provisioning
- password management
- logging, auditing, and reporting

### 3.3.9 Unity IdM

Unity IdM, or Unified identity management [22], is an extremely flexible authentication service. However, it should not be seen as a bundle of several coupled systems. It is a complete solution, fully web and cloud-ready, for identity, federation, and inter-federation management. Supporting multiple protocols with different configurations for many relying parties. The authentication process in Unity can be processed locally, with the built-in database or externally using one of the supported Identity Providers (IdPs). Unity also supports merging the information from upstream IdPs with the local database. This functionality can be seen on Figure 3.3 (available in Unity IdM website [22]):

Unity is built around three design principles:

- **Simplicity** - Unity should be as user friendly as possible.
- **Extensibility** - Seamlessly support for all the implemented protocols and possible extensions/future development.
- **Security** - Unity doesn't provide inherently insecure solutions.



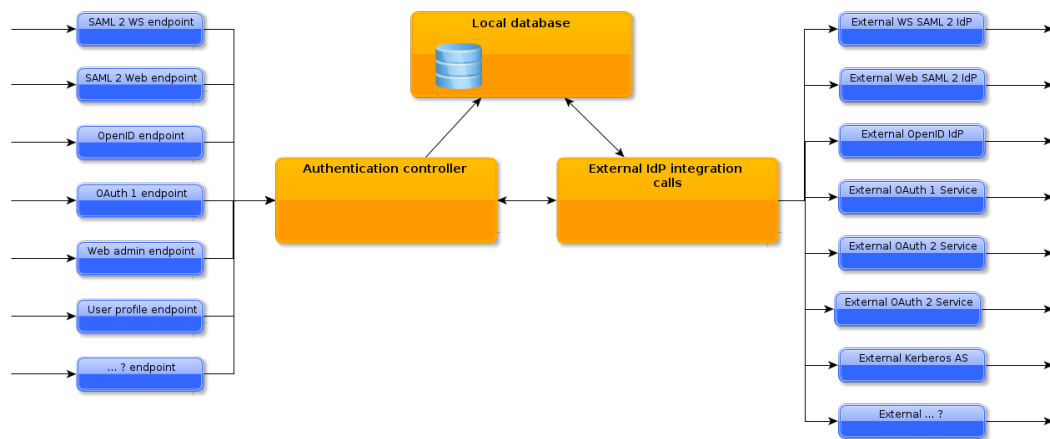


Figure 3.3: Unity Features [22]

Unity is composed of two parts, an orchestration platform and a rich set of extensions. The latter provides support for the actual Unity Features. As said in the documentation of the latest release of Unity, “the core platform provides persistence of the service state (i.e., managed entities, attributes, groups, etc.), extensions management, orchestration, and several cross-cutting features. Such core features are preferences management, notifications, registrations support, etc.” [22].

Unity is written only using Java, working as a set of services available via an embedded HTTP server. Being the most of its functionality controlled via a Web Admin interface.

To understand unity we first need to understand what an endpoint and authenticators are. Figure 3.4 helps to visualise the following text. Endpoints are the entry points for Unity, being this divided into Identity Providers and Services. Identity Providers provide the fundamental Unity features to the clients/relying parties and Services are the remaining functionality. Each endpoint’s authentication is configured by associating it with authenticator(s). As mentioned before, the authenticators can be local or external. The local authenticator performs verification of credentials against the data stored in the local Unity database. On the other hand, an external authenticator uses a 3rd party service to validate the credentials.

At the time of writing this document, Unity supports the following Endpoints:

- SAML 2, Web SSO profile
- SAML 2, SOAP binding
- SAML 2, PAOS binding (for ECP)
- Web Console interface
- REST admin interface
- Web user profile management
- SAML 2, Web binding + UNICORE profile

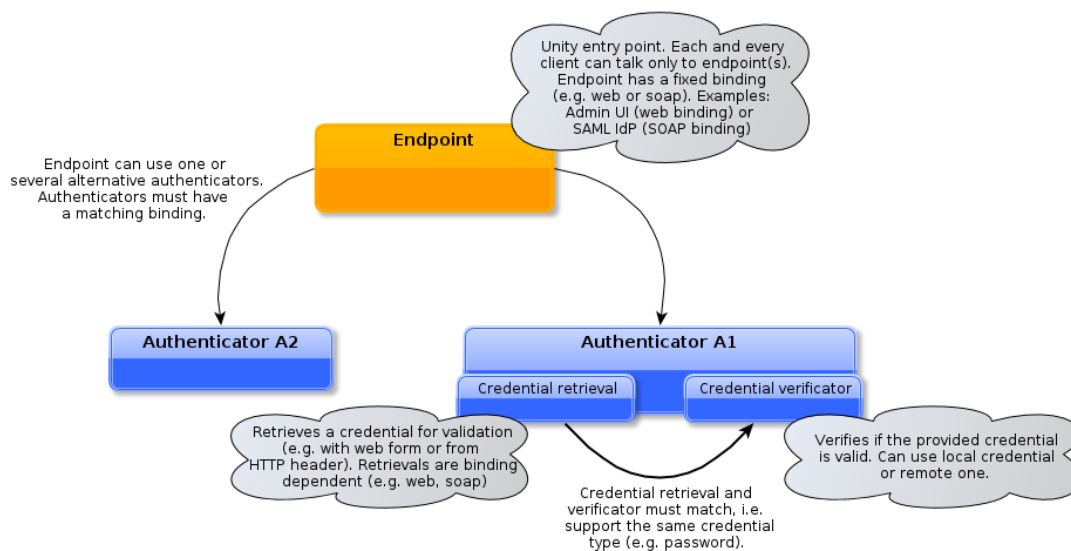


Figure 3.4: Unity fundamentals [22]

- SAML 2, SOAP binding + UNICORE profile
- OpenID Connect
- OAuth 2

Moreover, has support for the following upstream IdPs:

- LDAP
- SAML 2
- OAuth 2
- OpenID Connect
- Host OS PAM

### 3.3.10 WSO2

WSO2 Identity Server (WSO2 IS) [1] [23] is a product built on top of WSO2 Carbon. WSO2 Carbon redefines middleware by providing an integrated, and componentised middleware platform that adapts to the specific needs of any enterprise IT project, and it enables easy customisation and modularity through its architecture based on components.

WSO2 Identity Server (IS) is used directly by multiple users. Each user can have roles associated with him, where each role can have privileges attributed to them. User's roles can be changed at any time by an administrator of the system. Apart from such registered users, Identity Server can be used as an identity provider for third party applications, with their own set of users.

WSO2 Identity Server (IS) helps organisations to build agile, extensible IAM solutions to bring in better and seamless user experiences for their customers.

WSO2 Identity Server follows open standards and open-source principles, permitting independence from vendor lock-in. The product comes with seamless, easy-to-use integration capabilities that help connect applications, user stores, directories, and identity management systems.

The following items include a summary of the features and technologies offered by the WSO2 IS:

- Enterprise and cloud Single Sign-On and Federation with protocols such as SAML 2.0, OpenID Connect, WS-Federation.
- support for cross-protocol single logout
- support for multi-factor authentication and strong authentication
- administrator interface and self-service users portal
- support for LDAP, Microsoft Active Directory, or any JDBC database
- support for rule-based identity provisioning
- authorisations services and protocols such as role-based access control, XACML and OAuth2
- monitoring, reporting and auditing
- GDPR Compliance

### 3.4 Summary

In this chapter, we analysed available solutions for the proposed problem in Chapter 1. *Identity protocol translation gateway* patent [42], the *Interconnecting domains with heterogeneous key distribution and authentication protocols* paper [41], and some *Open Frameworks and toolkits* were evaluated and considered for the possibility of using them on this dissertation. We can now justify some affirmations made in Chapter 2, most of the solutions solve our problem in some form, this is not only about the number of features of each one. We can also recognise that there is not much research made on the topic of an identity protocol converter.



## Chapter 4

# Problem Statement

### 4.1 Preliminary Study

Previous work to this dissertation essentially concluded that solutions like the two previously mentioned in Section 3.1 and Section 3.2 are not the right way to tackle the problem at hands—the idea of a protocol converter able to transform the identity information from one identity protocol into another seems like a complicated task. As stated in [41], converting very different mechanisms is, at best, hard and may even be impossible in some instances. The FIdM protocols are indeed very different mechanisms having problems with interoperability within different implementations of the same version. Using an open-source IAM solution as a work base seems more promising and solves the problem introduced on Chapter 1.

### 4.2 IAM solutions comparison

Most of the previously mentioned IAM tools in Section 3.3 are very similar and complete IAM solutions. They follow the seven laws of identity (Section 2.1.2), take into account privacy guidelines on its core (Section 2.1.3), and in general tick every box on why they should be the chosen option for an open-source IAM solution (Section 2.1.4). However, even though they all are great candidates, OpenAM plus OpenIDM are open-core solutions, something that can be seen as a problem as we mentioned in Section 2.1.4 for an IAM solution. Furthermore, WSO2 IS has added comprehensive documentation and in general, is a more mature solution with more functionality.

Since most support the required functionality, we will base this dissertation on WSO2 IS following the preference of the company that proposed the problem.

### 4.3 Importance of a Security Analysis

Commercial solutions in principle force users to accept the level of security of the product; on the other hand, open-source solutions allow users to increase security as high as they want.

Software security is fundamentally in simple terms: run perfect software, and the system is secure. However, perfect software is not in the realm of possibility. This infeasibility creates the need for finding the means to prevent security flaws in software [31]. One way is to assure that the software only does what is supposed to do. Methods for this can be divided into three categories software auditing, vulnerability mitigation and behaviour management. The software auditing method prevents vulnerabilities by searching for them ahead of time, with or without automatic analysis. The vulnerability mitigation method is a compile-time technique that stops bugs at runtime. Behaviour management is the features of the operating system that either limit potential damage or block specific dangerous behaviours [31]. During this dissertation, we will only cover software auditing as a method.

The process of auditing existing applications is encouraged to find and remove flaws before attackers discover them. Open-source software is the ideal candidate for this since it enables anyone to make a software security analysis and share the results to the world.

IAM solutions deal with information that represents digital identities in different contexts. Protecting this information and its management is one of the most crucial security concerns [49].

## 4.4 Research Methodology

Taking into account what was mentioned in the previous sections, the main goal of this work is to contribute to a state-of-the-art IAM solution, in this case, WSO2 IS, by making a security analysis.

The current public documentation for WSO2 does not mention any security analysis previously done, and solutions dealing with identity must be as secure as possible.

In order to do so, we will apply current state-of-the-art threat analysis techniques and software auditing tools in the following chapters describing and giving context to all of them. This analysis will also give us a more detailed overview of the WSO2 IS product.

## Chapter 5

# Threat Analysis

### 5.1 Context

Security is about ensuring that assets do not undergo misapplication or malicious actions. The term asset is explained in [47], as something of value. Assets are what the attacker wants, and what people want to protect. An asset can be abstract or a concrete resource (eg.: passwords, credit card information).

Security threats strengthen security requirements; the identification of these threats is called threat modelling and is one of the essential steps in securing a system [39]. It involves identifying flaws and challenges which need to be approached by implementing countermeasures and realistic security requirements [52]. Threat analysis is the process of identifying, documenting and mitigating security threats of a system.

Threat modelling is one of the phases in threat analysis and has three components: system, assets and attacker. The three most popular approaches to threat modelling are asset-centric, attack-centric and software-centric [47]. These approaches are named based on the focus used to implement threat modelling. Software-centric models focus on the development and deployment of a system [47]. Attacker-centric models focus on profiling the attacker's characteristics, skillset and motivations to identify vulnerabilities [47]. Asset-centric models focus on protecting assets [47].

There are many threat analysis methodologies developed, [25], [38], [50] ; however, in this dissertation, we will essentially follow the threat analysis methodology developed by [50]. This methodology divides the analysis into three main phases:

- Threat modelling - Method for evaluating and documenting security risks of an application with an attack-centric focus. This method allows to enumerate threats and discover security flaws. The list of threats can and should be updated as needed.
- Asset mapping - Phase for documenting the concrete and abstract resources of the system. In this phase, there is a need to prioritise assets since this value is used to calculate threat risks and prioritise countermeasures.

- Building a mitigation plan - This phase is as simple as selecting the most effective combination from a list of all the proposed countermeasures. There are two steps to choosing the most effective combination. First, estimate the mitigation level of each countermeasure as if it is the only one. Second estimate the total level of the mitigation provided by all countermeasures.

The proposed methodology in [50] formalises this steps into a new approach. In Figure 5.1, this threat analysis methodology is shown step by step.

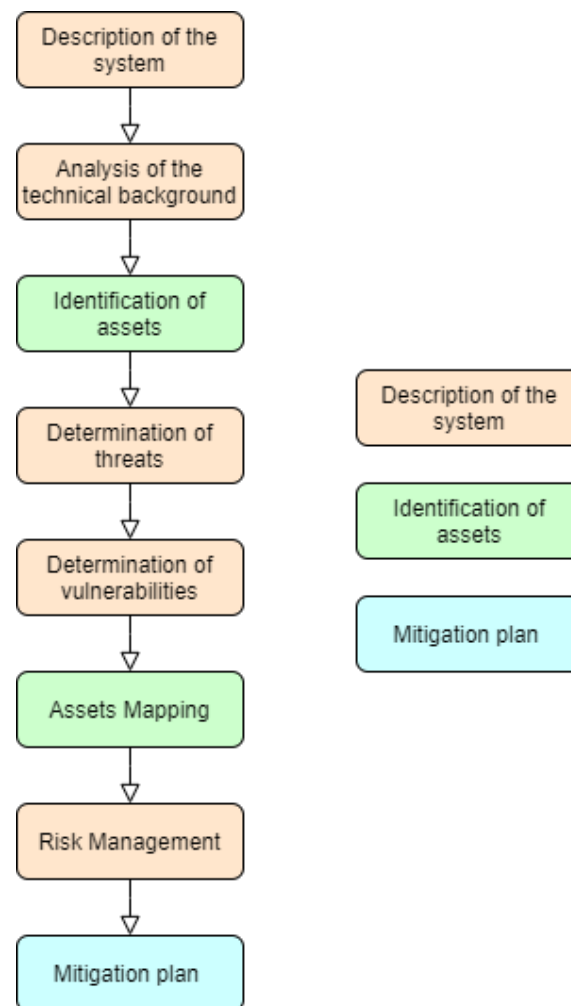


Figure 5.1: Steps of Threat analysis

## 5.2 Methodology

As previously said, we will follow the threat analysis methodology developed by [50] to analyse the WSO2 IS v5.10 IAM solution. The chosen threat analysis methodology was created for personal networks, not for an IAM solution; however, we will adapt the method to our problem at



hands.

This methodology is divided into eight steps. Within this section, we will explain each step and understand their purpose.

In step 1, we will make a description of the system. In order to describe the system, we will define the scenarios and use cases to help understand every system component and its interconnections. For this characterisation, we will make UML use case diagrams with a descriptive table. This diagram allows the understanding of what the system is capable of doing by showing each interaction between use cases and actors.

In step 2, we will analyse the technical background of the use cases. In this step, it is possible to explain how the technologies are used and who is using them. The idea of this step is to identify how the system components/modules interact with each other for the system use cases.

In step 3, we will identify the system's assets. In other words, we will identify everything that can be damaged or violated in the network. The protection of assets is one of the most critical parts of software security. Organisations often focus on protecting the system from threats which can leave a system vulnerable. The previous two steps allow to more easily identify the general and specific assets of each use case. The assets will be enumerated in a table with ID, name and a brief description of the asset.

In step 4, with the information from the previous steps, we will start to identify the system's threats and its sources. A threat-source can be considered as any circumstance or event with the potential to cause harm to the system. Afterwards, threats should be matched to their associated assets. This step results in a table, named threats profile, this table has an ID, a name, the source of the threat, and the assets involved. Threats also will be analysed to define whether the system is an easy target to them.

In step 5, we will list the system's vulnerabilities that could be exploited by the potential threat sources. This step results in a table with vulnerabilities and the corresponding threats that are exploiting the previous specific use cases analysed in the previous steps. A vulnerability ID, followed by its description, the name and the corresponding threat will compose the table.

In step 6, the list of assets from step 3 will be checked for inclusion of all assets. The objective of this step is to assign a value to the assets that categorises its priority. To assign priority to assets, we will use the method proposed by [50], dividing this prioritisation into three different values:

- High - Assets with this value are critical and need more emphasis on protecting them. This assets usually have a significant financial value.
- Medium - Assets usually linked to common services; they are not critical but still important to protect.
- Low - Assets of minor importance.

In step 7, we will make a risk management analysis to balance what is possible and what is acceptable. From step 4 and 5, we can extract which threats pose the highest risk value. The

aspects, which we will take into account to categorise vulnerability risk are the impact, the damage to the assets, the size of the vulnerability and the possibility of it happening.

In the final step, step 8, we will develop a mitigation plan that involves the selection of countermeasures. The threats selected for mitigation are addressed as by one or more countermeasures. This step results in a set of proposed countermeasures that mitigate the threats that were identified. However, the implementation of all proposed countermeasures usually is impractical due to budget, time and resources constraints. The goal of this final step, and therefore the purpose of this threat analysis is to propose the set of the most cost-effective countermeasures related to the identified threats.

### 5.3 Step 1. Description of the system

In the first step, we made four UML use case diagrams used to describe what the system must be able to do and to help in the description of the system. This chapter was made using the information on WSO2 documentation [23], and our personal experience with the solution; there are minimal UML diagrams made on WSO2 IS, this is standard in other open-source community-driven products. As said by a member of the WSO2 IS team when contacted, "WSO2 IS ... is designed by community programmers for programmers, thus additional complexity and overhead for UML does not provide much advantage. It only adds cost of tools for design and collaboration".

Since WSO2 IS is an extensive and highly-featured solution, we divided the use case diagram of the system by its boundary of systems. We can identify four boundary systems in WSO2 IS:

- The user management boundary system with its use cases represented in Figure 5.2
- The authentication boundary system with its use cases represented in Figure 5.3
- The access delegation and policies boundary system with its use cases represented in Figure 5.4
- The administration boundary system with its use cases represented in Figure 5.5

We also made use case tables to better represent the solution (Tables 5.1-5.11). Some use cases were excluded from these tables because of their nature. WSO2 IS is a developer-oriented solution, not something plug-in and play, and use cases such as adaptive authentication and provisioning are developer-oriented features. For example, to make a use case table for inbound provisioning using SCIM 2, it would be running a curl command to `https://<Your WSO2IS URL>/scim2/Users`.

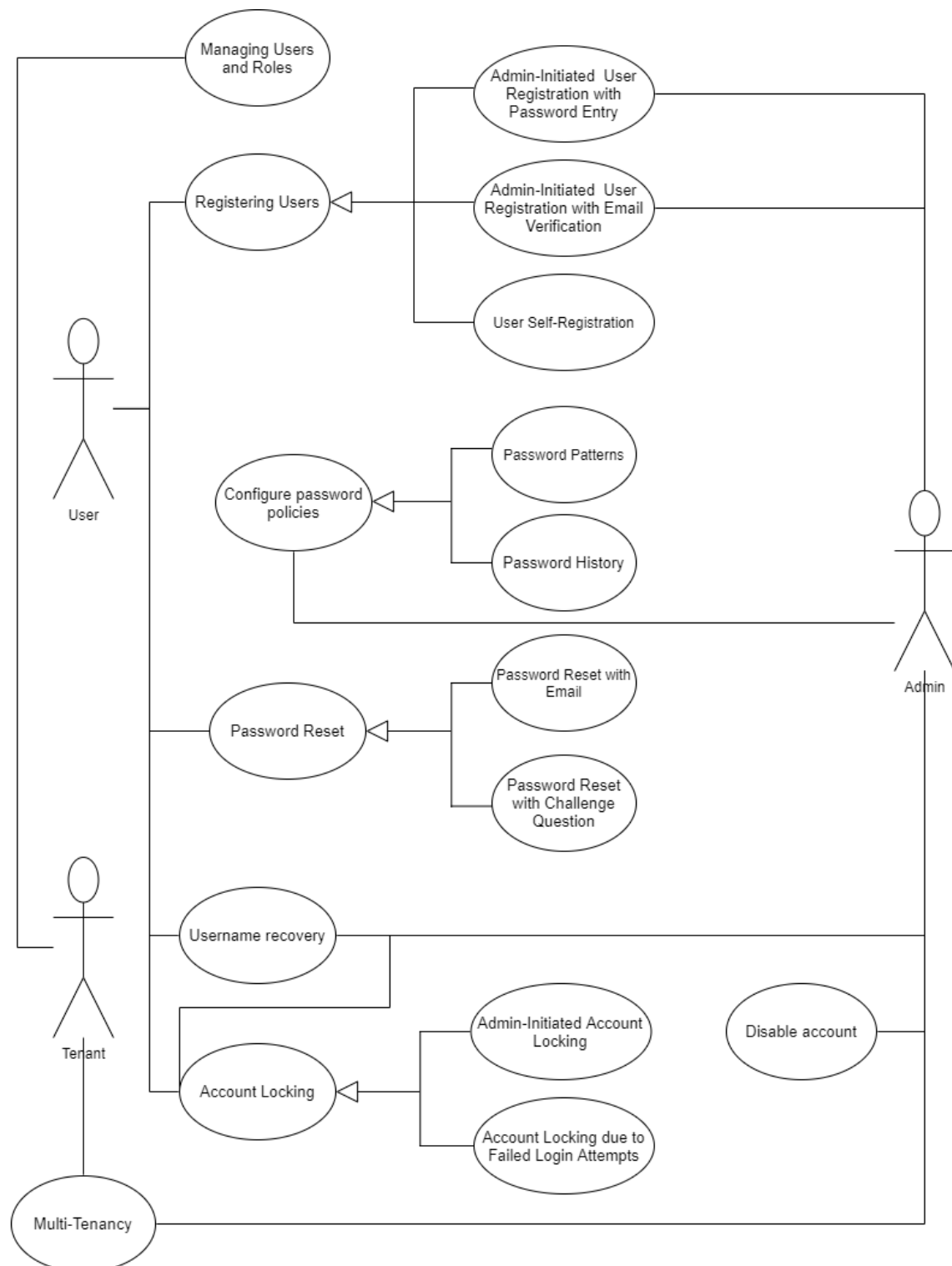


Figure 5.2: Use case diagram - User management boundary system

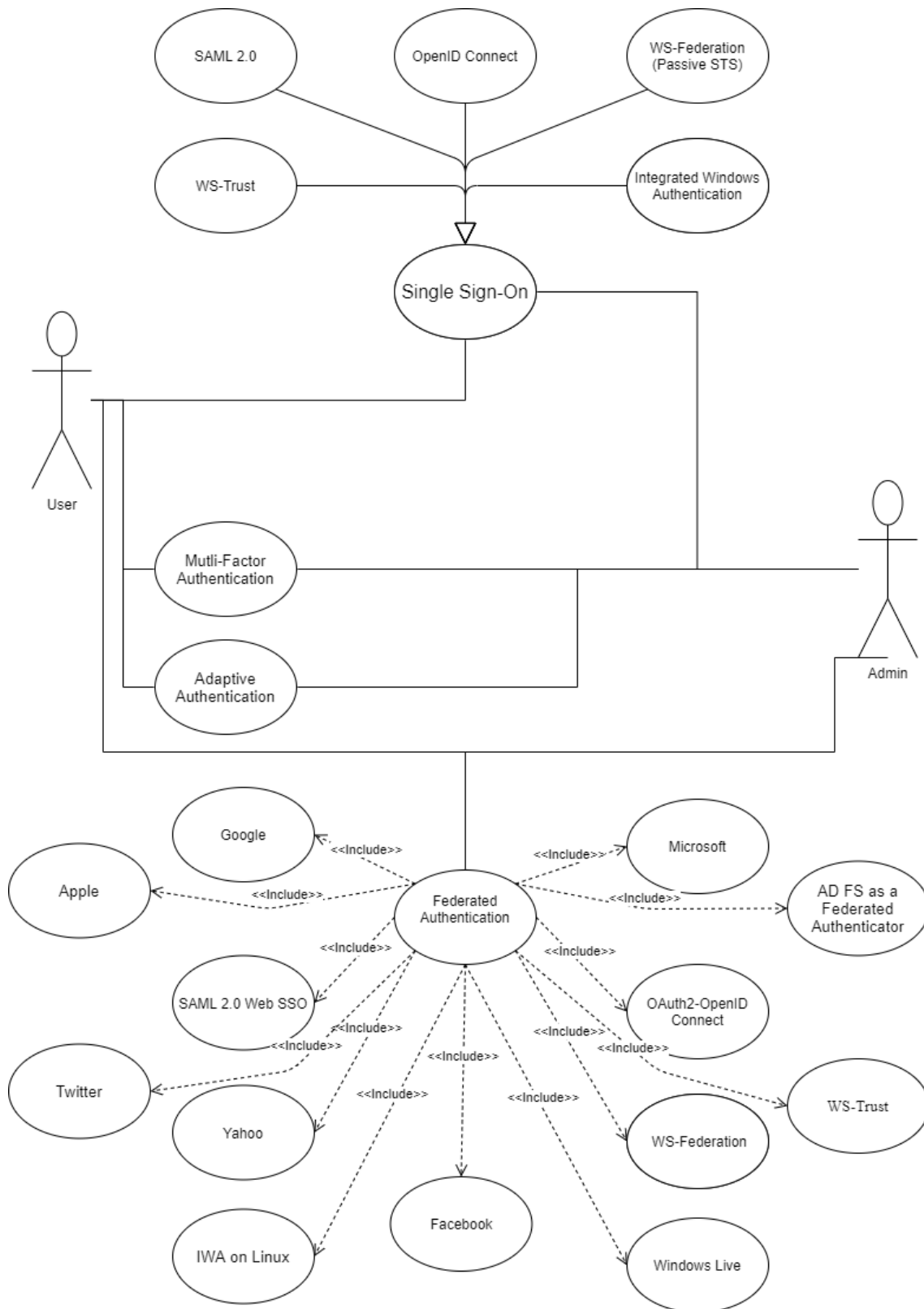


Figure 5.3: Use case diagram - Authentication boundary system

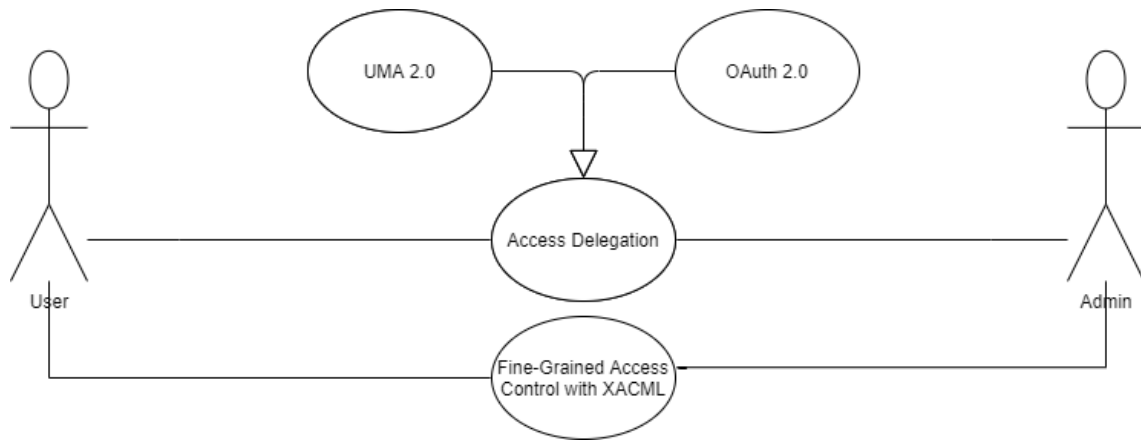


Figure 5.4: Use case diagram - Access delegation and policies

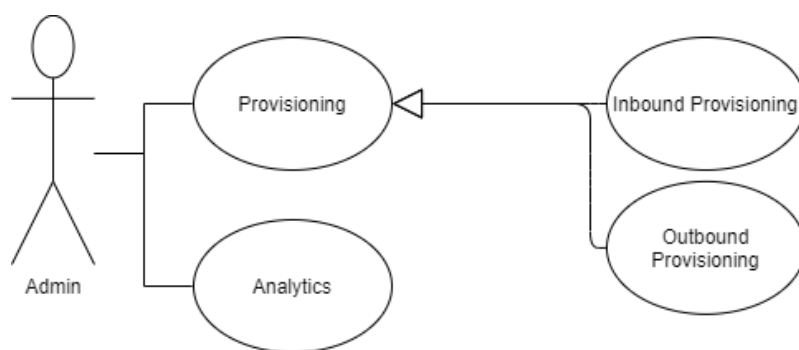


Figure 5.5: Use case diagram - Administration boundary system

Table 5.1: Use case table - Managing Users and Roles

<b>Use case name</b>	Managing Users and Roles
<b>Goal in context</b>	Creation of roles and users
<b>Preconditions</b>	Tenant or Admin account
<b>Successful end</b>	New system user assigned to a new role
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console</li> <li>2. Admin clicks Users and Roles &gt; Add &gt; Add New Role.</li> <li>3. Admin selects the user store in which he wants to create this role and enters the role name.</li> <li>4. Admin clicks Next and Selects the permissions that he wants the users with this role to have.</li> <li>5. Admin clicks Update.</li> <li>6. Admin clicks Add under Users and Roles on the Main tab of the Management Console followed by clicking on add new user</li> <li>7. Admin enters a unique username and password and clicks Next to assign the User to a specific role.</li> <li>8. Admin selects the role that they want the new User to have and clicks Update.</li> </ol>

Table 5.2: Use case table - Admin-Initiated User Registration

<b>Use case name</b>	Registering Users
<b>Goal in context</b>	Admin-Initiated User Registration
<b>Preconditions</b>	Admin account
<b>Successful end</b>	A new User is registered
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks click Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks User Onboarding under the Account Management Policies section and selects the Enable User Email Verification check box.</li> <li>4. Admin enters the password entry validity period (in minutes) in the Ask password code expiry time text box.</li> <li>5. Admin clicks Update.</li> <li>6. Admin clicks Add under Users and Roles on the Main tab of the Management Console followed by clicking on add new user.</li> <li>7. Admin enters the required data, activates the option ask password from user and clicks finish.</li> <li>8. The User receives an email; in this email, the User clicks Create Password.</li> <li>9. A Reset Password screen appears and the User enters a preferred password and clicks Submit.</li> </ol>

Table 5.3: Use case table - Self-Registration

<b>Use case name</b>	Registering Users
<b>Goal in context</b>	User self-Registration
<b>Preconditions</b>	Admin account
<b>Successful end</b>	A new User is registered
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks User Self Registration under the Account Management Policies section.</li> <li>4. Admin selects the Enable Self User Registration, Enable Account Lock On Creation Enabled and Enable Notification Internally Management check boxes.</li> <li>5. Admin enters the rest of the required information and clicks Update.</li> <li>6. User Accesses WSO2 Identity Server User Portal at <a href="https://&lt;WSO2 IS URL&gt;/user-portal/">https://&lt;WSO2 IS URL&gt;/user-portal/</a> and clicks create account.</li> <li>7. The User receives an email; in this email, the User clicks Create Account, then the User enters its desired username and clicks Proceed to Self Register.</li> <li>8. The User enters the required information and clicks Register.</li> </ol>

Table 5.4: Use case table - Password Patterns

<b>Use case name</b>	Configure Password Policies
<b>Goal in context</b>	Configuration of a password pattern
<b>Preconditions</b>	Admin account
<b>Successful end</b>	Users can't have passwords that don't follow the password pattern
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks click Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks Password Patterns under the Password Policies section.</li> <li>4. Admin enables the password policy feature and enters the required information, including the Policy Pattern.</li> <li>5. Admin clicks Update.</li> </ol>



Table 5.5: Use case table - Password Reset

<b>Use case name</b>	Password Reset
<b>Goal in context</b>	Password Reset with Email
<b>Preconditions</b>	Admin account and User account
<b>Successful end</b>	User receives an e-mail to reset its password
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks Account Recovery under the Account Management Policies section.</li> <li>4. Admin selects Enable Notification Based Password Recovery check box and clicks Update.</li> <li>5. The User enters the Access WSO2 Identity Server User Portal and clicks Password.</li> <li>6. The User inputs its user name, selects the Recover with Mail option and clicks Submit.</li> </ol>

Table 5.6: Use case table - User Name Recovery

<b>Use case name</b>	User Name Recovery
<b>Goal in context</b>	Recovery of forgotten username
<b>Preconditions</b>	Admin account and User account
<b>Successful end</b>	User receives an e-mail with its username
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks Account Recovery under the Account Management Policies section.</li> <li>4. Admin selects the Enable Username Recovery and Enable Internal Notification Management checkboxes, and clicks Update.</li> <li>5. The User enters the Access WSO2 Identity Server User Portal and clicks Username.</li> <li>6. The User inputs his information clicks Submit.</li> </ol>

Table 5.7: Use case table - Multi-tenancy

<b>Use case name</b>	Multi-tenancy
<b>Goal in context</b>	Create a new Tenant
<b>Preconditions</b>	Admin account
<b>Successful end</b>	A new Tenant can manage users and service providers of his associated domain
<b>Primary actors</b>	Admin and Tenant
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Add New Tenant on the Configure tab.</li> <li>3. Admin inserts the domain of the new Tenant and his details (e.g. user credentials).</li> <li>4. The Tenant navigates to WSO2 Identity Server User Portal and authenticates using his credentials.</li> </ol>

Table 5.8: Use case table - Account Locking

<b>Use case name</b>	Account locking
<b>Goal in context</b>	Account locking for Failed Login Attempts
<b>Preconditions</b>	Admin account and User account
<b>Successful end</b>	An email that informs that informs the user about the account locking is sent to him.
<b>Primary actors</b>	Admin and User
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity &gt; Identity Providers &gt; Resident.</li> <li>3. Admin clicks Account Locking under the Login Policies section.</li> <li>4. Admin inputs the desired Maximum Failed Login Attempts and Account Unlock Time.</li> <li>5. The User enters the Access WSO2 Identity Server User Portal and tries to login with the wrong password multiple times.</li> </ol>

Table 5.9: Use case table - Single Sign-On

<b>Use case name</b>	Single Sign-On
<b>Goal in context</b>	Single Sign-On Using SAML2
<b>Preconditions</b>	Admin account and 2 sample web apps with support for SAML and already integrated with WSO2 IS
<b>Successful end</b>	Admin is authenticated in both sample applications
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Main&gt;Identity&gt;Service Providers and then clicks Add.</li> <li>3. Admin enters &lt;your first SAML 2 web app&gt; in the Service Provider Name text box, and clicks Register.</li> <li>4. In the Inbound Authentication Configuration section, the Admin clicks Configure under the SAML2 Web SSO Configuration section.</li> <li>5. Admin sets the Issuer and the Assertion Consumer URL for the first SAML web app, clicks Yes in the message that appears and finally clicks the following checkboxes: Enable Response Signing, Enable Single Logout, Enable Attribute Profile, Include Attributes in the Response Always, and Enable Signature Validation in Authentication Requests and Logout Requests.</li> <li>6. Admin clicks Register to save changes and repeats steps 1 to 5 for the second SAML 2 web app.</li> <li>7. Admin navigates to the Assertion Consumer URL of the first SAML web app and authenticates himself using the SAML 2 feature.</li> <li>8. Admin navigates to the Assertion Consumer URL of the second SAML web app and is automatically authenticated.</li> </ol>

Table 5.10: Use case table - Multi-factor authentication

<b>Use case name</b>	Multi-factor authentication
<b>Goal in context</b>	Multi-factor authentication with SMSOTP
<b>Preconditions</b>	Admin account, User account, SMS provider already integrated with WSO2 IS, and a SP that uses WSO2 IS as its IdP (e.g. SP from Table 5.12)
<b>Successful end</b>	Admin authenticates successfully
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity Providers &gt; Add on the Main tab.</li> <li>3. Admin expands the SMS OTP Configuration tab under Federated Authenticators and selects both checkboxes to Enable SMSOTP Authenticator and to make it the Default.</li> <li>4. Admin enters the SMS URL and POST as the HTTP Method and clicks Register.</li> <li>5. Admin clicks Service Providers &gt; List and Edits the desired service provider.</li> <li>6. Admin expands Claim configuration and selects <a href="http://wso2.org/claims/mobile">http://wso2.org/claims/mobile</a> as the Subject Claim URI.</li> <li>7. Admin clicks Add Authentication Step, selects basic under Local Authenticators and click Add Authenticator to add the basic authentication as the first step.</li> <li>8. Admin clicks Add Authentication Step and selects smsotp under Federated Authenticators and click Add Authenticator to add SMSOTP authentication as the second step.</li> <li>9. Admin clicks Update.</li> <li>10. The User navigates to the Service Provider URL and attempts to log in.</li> <li>11. The User inputs his credentials after being redirected to the login page of the WSO2 Identity Server.</li> <li>12. A SMSOTP code will be sent to the User's mobile number. The User enters the code and clicks Authenticate.</li> </ol>

Table 5.11: Use case table - Federated Authentication

<b>Use case name</b>	Federated Authentication
<b>Goal in context</b>	Federated Authentication with SAML 2.0 WEB SSO
<b>Preconditions</b>	Admin account
<b>Successful end</b>	Federated authentication is enabled
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Identity Providers &gt; Add on the Main tab.</li> <li>3. Admin enters in the details in the Basic Information section.</li> <li>4. Admin clicks SAML2 Web SSO Configuration under the Federated Authenticators section</li> <li>5. Admin clicks the Enable SAML2 Web SSO and Default checkboxes.</li> <li>6. Admin clicks Register.</li> </ol>

Table 5.12: Use case table - Access Delegation

<b>Use case name</b>	Access Delegation
<b>Goal in context</b>	Access Delegation with OAuth 2.0
<b>Preconditions</b>	Admin account and 1 sample web app with support for OAuth 2.0 and already integrated with WSO2 IS
<b>Successful end</b>	Admin shares his info with the service provider
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console</li> <li>2. Admin clicks Main&gt;Identity&gt;Service Providers and then clicks Add.</li> <li>3. Admin enters &lt;your OAuth 2.0 app name&gt; in the Service Provider Name text box, and clicks Register.</li> <li>4. Admin inputs the Callback URL, clicks Add, and saves the OAuth Client Key and Client Secret that is displayed.</li> <li>5. In the sample web app replace, in its designated place, the consumer key and secret consumer values with the OAuth client key and OAuth client secret values saved in the previous step.</li> <li>6. Admin navigates to Callback URL of the OAuth 2.0 app and notices that its possible to make authorisation requests to the WSO2 IS deployment.</li> </ol>

Table 5.13: Use case table - Fine-Grained Access Control with XACML

<b>Use case name</b>	Fine-Grained Access Control
<b>Goal in context</b>	Fine-Grained Access Control with XACML
<b>Preconditions</b>	Admin account
<b>Successful end</b>	Creation of an Fine-Grained Access Control policy
<b>Primary actors</b>	Admin
<b>Main flow</b>	<ol style="list-style-type: none"> <li>1. Admin Accesses the WSO2 Identity Server Management Console.</li> <li>2. Admin clicks Entitlement &gt; PAP &gt; Policy Adminisration.</li> <li>3. Admin clicks Add New Entitlement Policy &gt; Simple Policy Editor.</li> <li>4. Admin configures the desired policy and clicks Add.</li> </ol>

## 5.4 Step 2. Analysis of the technical background

In this section, we will make an architecture overview of the WSO2 IS to understand the system flow. Figure 5.6 helps to visualise the following text.

When an authentication request is sent to the WSO2 IS the Inbound Authenticators component processes it. The Inbound Authenticator component sends this request to an in-channel of the Authentication Framework. The Authentication Framework handles claims; WSO2 IS [23] defines a claim as data that defines the User, anything that the User is, and is associated with. The Authentication Framework maps claims in the service provider to claims in the identity server, afterwards it can be used to map claims in the identity server to claims in an external application that acts as the identity provider. The Authentication Framework sends a request to the Local Authenticators component if the authentication is to be done by the identity server itself or for integrated windows authentication. If the authentication needs to be done by an external application or an identity provider, the request is sent to the Federated Authenticators component. The Federated Authenticators component will send the request to an external application where authentication is done.

When the authentication is completed, the response is sent to the out-channel of the Authentication Framework. This is where claim mapping happens again. The Authentication Framework has a connector named JIT provisioning for federated authentication. This sends user information from the identity provider to the Provisioning Framework. The Provisioning Framework ensures that users from the identity provider are provisioned into the WSO2 User Store Manager. Afterwards, the out-channel of the authentication framework sends the response to the response generator to process it. The authentication process ends when the authentication response is sent back to the service provider.

Identity provisioning is in simplistic terms the process of creating, deleting, and maintaining user accounts and their related identity in one or more systems.

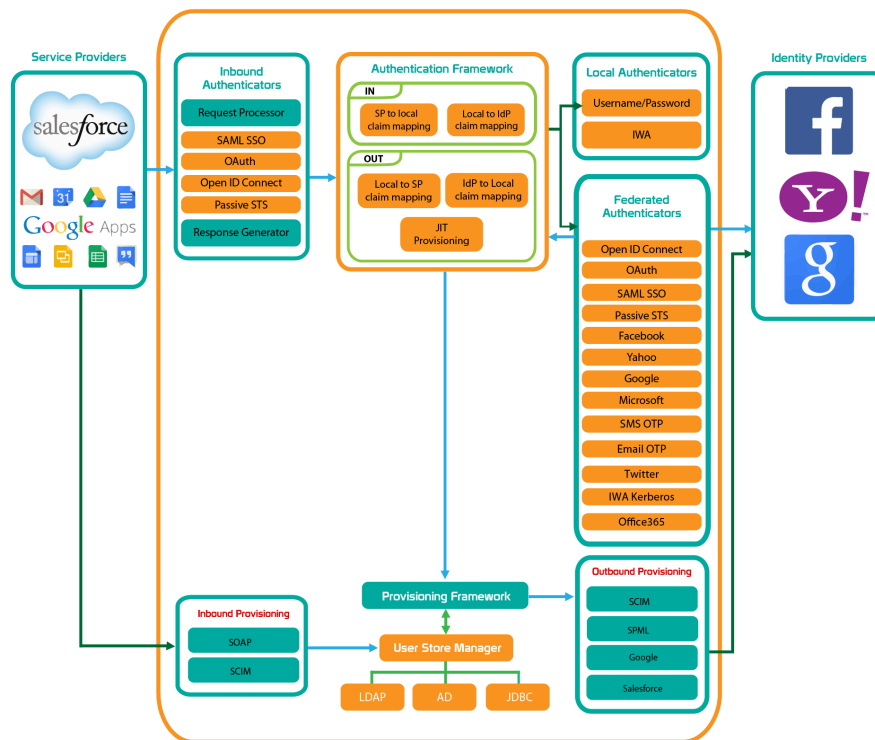


Figure 5.6: WSO2 IS architecture overview and process flow [23]

The Provisioning Framework can be used to provision users in the WSO2 IS, or the third-party identity providers via the WSO2 IS, in response to business processes. In-bound provisioning requests are met in the In-bound Provisioning component. Then it connects to the user store manager, which has a component called the provisioning listener. The User is added to the user store in the WSO2 IS. The provisioning listener communicates with the Provisioning Framework and passes on the request. If the user provisioning needs to be done in an external application, the Provisioning Framework sends a request to the out-bound provisioning connectors component. The out-bound provisioning component is responsible for ensuring that the users are also added to external applications.

## 5.5 Step 3. Identification of assets

Analysing the results of steps 1 and 2, and the database resultant of the WSO2 IS database creation scripts, it is possible to identify the assets of the system. This step results in the Table 5.14.

Table 5.14: WSO2 IS Assets

ID	Name	Description
A1	IDs	E.g. User ID
A2	Personal Data	General Information about the user (eg. last connect IP)
A3	Availability/access of the services	The rights to access some services
A4	Access control	Rules to access data and services.
A5	User's credentials	e.g. Username, password hash.
A6	Contact Information	e.g. E-mail, Phone number.
A7	Server Characteristics	Hardware and Software of the server.
A8	Registry activities	Registry is a content store and a metadata repository for various artefacts such as services, WSDLs and configuration files. All configurations about modules, logging, security, data sources and other service groups are stored in the registry by default.
A9	Domain	WSO2 supports the use of different user stores. Domain data includes which rules are related to which users, that are related to which user stores.
A10	Identity providers data	Information and configurations about registered Identity providers.
A11	Service providers data	Information and configurations about registered Identity providers.
A12	Identity protocols related data	This contains all the data generated by identity protocols data, in its creation and its use. (e.g. OAuth2.0 request tokens).
A13	Certificates	All the certificates used to operate WSO2 IS.
A14	Logs	The server should be continuously logging the solution.

## 5.6 Step 4. Determination of threats

In the fourth step, the possible threats and the assets compromised by them will be shown in Table 5.15. WSO2 IS is as insecure as it features. Because of this, we decided to investigate which threats could be exploited by using all the protocols in WSO2 IS. The shown results are resultant of personal analysis of previous steps, the work in [48] and tentative of extension of this work.



In [48], was conducted a systematic survey of security analysis in Federated Identity Management (FIdM). The goals of this survey were to understand the Security Landscape in FIdM, find the common Attack Classes (threats) and check if the people who reported the problems proposed solutions or mitigations to those problems. With this goal in mind, the authors used the following query in Scopus:

```
1 TITLE-ABS-KEY ( ( Analysis OR Evaluation OR Examine OR Proof OR Attack OR Intrusion
    OR Vulnerability OR Risk ) AND Security AND Identity AND (OAuth OR OpenID OR "
    Liberty Alliance" OR WS-Federation OR SAML OR "Security Assertion Markup
    Language" OR "Microsoft Passport" OR ( Passport AND Protocol ) OR Cardspace OR
    "Facebook Connect" OR "Google Accounts" OR Shibboleth ) ) AND ( LIMIT TO (
    SUBJAREA, "COMP") )
```

And the following query in Google Scholar:

```
1 Protocol Security Identity Analysis OR Evaluation OR Examine OR Proof OR Attack OR
    Intrusion OR Vulnerability OR Risk
```

In a tentative to extend their work to other protocols and functionality used in WSO2 IS we created the following query to be used in Scopus:

```
1 TITLE-ABS-KEY ( Analysis OR Evaluation OR Examine OR Proof OR Attack OR Intrusion
    OR Vulnerability OR Risk ) AND Security AND Identity AND ( UMA OR "User Managed
    Access", OR SCIM OR XACML OR "Integrated Windows Authentication" OR WS-Trust
    OR IWA)) AND ( LIMIT-TO ( SUBJAREA, " COMP"))
```

After using the same filtering process as in [48], this process resulted in no further threats or vulnerabilities to be considered in this threat analysis.

Table 5.15: Threats

ID	Name (classification)	Source	Assets
T1	Spoofing to accede private information	Human	A1, A3
T2	Eavesdropping on federation members	Human	A2, A12, A10, A11
T3	Identity theft	Human	A1 - A6
T4	Denial of Service	Human	A1 - A6
T5	Information disclosure	Human	A1 - A6
T6	Replay Attack	Human	A1 - A6
T7	Malicious Providers	Human	A2, A6, A10 - A12
T8	MITM	Human	A2, A6, A12
T9	Message Modification	Human	A2, A6
T10	Brute Force	Human	A1-A6
T11	XSS	Human	A2, A6
T12	Session Swapping	Human	A2, A6
T13	DNS Poisoning	Human	A2, A6
T14	CSRF	Human	A2, A6, A11
T15	UI Redressing	Human	A2, A6, A12
T16	Malware	Human	All
T17	Bogus Merchant	Human	A2, A6

## 5.7 Step 5. Determination of vulnerabilities

In step 5, we collected the vulnerabilities of WSO2 IS by analysing the enumerated threats. This results in Table 5.16 filled in with the main vulnerabilities that are exploiting the specific use cases analysed in previous steps, with each vulnerability will be matched with the corresponding threats. The same process used to gather the information for step 4 is used to gather the needed information for step 5.

## 5.8 Step 6. Assets Mapping

The output of this step is a final table (Table 5.17) with all the assets and the corresponding value assigned taking in to account the scenarios and the particular use cases.

Table 5.16: Vulnerabilities

ID	Vulnerability	Corresponding threats	Description
V1	An adversary gain access to a User or Admin account	T1, T2, T3, T5, T6, T10, T16	This could happen on a situation where a User/Admin the device without logout and an adversary steals it.
V2	Unencrypted Communications	T2, T6, T9	An adversary intercepts communication between members.
V3	User blind trust	T5	The User can give personal information without checking if the recipient is trusted.
V4	Centralised Infrastructure	T4, T7	A centralised infrastructure can lead to confidentiality problems with some authentication protocols and its an easy target for Denial of Service attacks.
V5	Lack of Binding	T6, T8, T9, T12	A sent message is not sufficiently tied to the sender.
V6	No Trust Infrastructure	T17	Situation where there is no safe infrastructure list, meaning a user does not know who to trust.
V7	Weak DNS	T13	It is often the case that certain vulnerabilities lead to specific threats. In this case, a Weak DNS leads to an easy DNS poisoning target.
V8	Vulnerable SP	T11, T14	Happens when an SP is not properly secured.
V9	Vulnerable IdP	T15	Happens when a IdP is not properly secured.
V10	Automatic Authorisation	T11	Happens if a User has granted a privilege it is automatically granted again.
V11	Message Formatting	T9	Happens when a parameter or part of the message is not signed properly.
V12	Weak User Credentials	T10	Same case as V7, weak credentials makes an easy target for brute force attacks.

Table 5.17: Assets Mapping

ID	Name	Description	Priority
A1	IDs	e.g. User ID, IdP ID, SP ID	LOW
A2	Personal Data	General Information about the user (eg. last connect IP).	HIGH
A3	Availability/access of the services	The rights to access some services.	MEDIUM
A4	Access control	Rules to access data and services.	MEDIUM
A5	User's credentials	e.g. Username, password hash.	HIGH
A6	Contact Information	e.g. E-mail, Phone number.	HIGH
A7	Server Characteristics	Hardware and Software of the server.	LOW
A8	Registry activities	Registry is a content store and a metadata repository for various artefacts such as services, WSDLs and configuration files. All configurations about modules, logging, security, data sources and other service groups are stored in the registry by default.	HIGH
A9	Domain	WSO2 supports the use of different user stores. Domain data includes which rules are related to which users, that are related to which user stores.	MEDIUM
A10	Identity providers data	Information and configurations about registered Identity providers.	HIGH
A11	Service providers data	Information and configurations about registered Identity providers.	HIGH
A12	Identity protocols related data	This contains all the data generated by identity protocols data, in their set up and their use. (e.g. OAuth2.0 request tokens).	HIGH
A13	Certificates	All the certificates used to operate WSO2 IS.	HIGH
A14	Logs	The server should be continuously logging the solution.	MEDIUM

## 5.9 Step 7. Risk Management

From the list of assets and their priority, the list of threat and the list of vulnerabilities it is possible to extract the information about which threats pose the highest risk, this results in Table 5.18.

Table 5.18: Threats associated with risk

ID	Name (classification)	Vulnerabilities	Risk
T1	Spoofing to accede private information	V1	HIGH
T2	Eavesdropping on federation members	V1, V2	MEDIUM
T3	Identity theft	V1	HIGH
T4	Denial of Service	V4	LOW
T5	Information disclosure	V1, V3	MEDIUM
T6	Replay Attack	V5	HIGH
T7	Malicious Providers	V4	MEDIUM
T8	MITM	V5	HIGH
T9	Message Modification	V2, V5, V12	HIGH
T10	Brute Force	V1, V12	HIGH
T11	XSS	V8, V10	HIGH
T12	Session Swapping	V5	HIGH
T13	DNS Poisoning	V7	LOW
T14	CSRF	V8	HIGH
T15	UI Redressing	V9	MEDIUM
T16	Malware	V1	LOW
T17	Bogus Merchant	V6	LOW

## 5.10 Step 8. Mitigation plan

The last step of the threats analysis is the construction of a mitigation plan that involves the selection of the countermeasures. The threats with higher risk are of the most worthwhile to be secured. The developed mitigation plan includes all high-risk threats and some medium to low risk.

Starting with malicious providers, this is a threat challenging to stop; to mitigate this threat, we propose an individual analysis of each provider that would be included in the system.

Unencrypted communications and replay attacks can be mitigated by forcing the use of SSL/TLS in all communications; this includes the enforcing of HTTPS.

The bogus merchant, an attack in which an imposter sets up an imitation web site, and problems derivated by the use of weak user credentials can be mitigated by introducing a higher level of authentication (preferable a strong authentication as defined by [32]).

Binding requests to the session can mitigate the CSRF threat; this can be achieved by hashing a secret together with a session id and appending that token into a hidden field in the login form. This token should be unique across requests.

In order to mitigate MITM attacks, access tokens should always be bound and explicit to a single SP.

Session Swapping could be mitigated by adding a state value binding the message. This can be achieved by binding the session ID to other user or client properties, such as the client IP address, User-Agent, or client-based digital certificate.

Proper sanitisation of inputs prevents the effectiveness of XSS attacks.

## 5.11 Discussion

In this chapter, we established a threat analysis for the WSO2 IS IAM solution. By characterising the system, identifying threats, assets, and vulnerabilities, and defining the threats and their effects, we gained a deep understanding of the WSO2 IS for purposing a mitigation plan, in order to prepare for those threats mentioned above.

The WSO2 IS already mentions in its documentation ways to mitigate some of the mentioned threats. Furthermore, WSO2 IS has all the tools necessary for the implementation of a secure deployment with the propose mitigation plan in mind.

The WSO2 IS team proposes the use of OWASPCSRFGuard project [15] to mitigate CSRF attacks, the use of the recovery system or reCaptcha after a certain number of failed attempts to authenticate, in order to mitigate brute force attacks. Moreover, the use of timestamps in WS-Security is proposed by the WSO2 IS team to mitigate replay attacks [23].

Can we conclude the WSO2 IS is a safe solution? The security of WSO2 IS is very dependent on its deployment configuration. A straightforward example of this is multi-factor authentication, its the Admin option to activate this functionality, but he has the choice not to.

This threat analysis cannot replace a threats analysis made to specific WSO2 IS deployments, only supporting on its development. However, we can confidently assure that WSO2 IS has all the tools necessary to be a secure solution for Identity and Access Management.

## Chapter 6

# Security Audit of WSO2 IS

### 6.1 Context

Today many open-source software tools are available to aid on the development of secure software. Modelling, architectural, code analysis, box testing and penetration testing tools are used to support the development of software on its entire lifecycle, integrating security along the way [26], [31], [36].

These tools are used for security code review practices. They can be divided into static analysis tools and dynamic analysis tools. Static analysis tools can analyse software without executing it; they examine the source code and report any suspicious code sequence that can be vulnerable. However, these analysers may report suspicious code that is, in fact, safe; this translates into a high false-positive rate [26], [31], [36].

There are many types of static analysis tools such as source code security analysers, bytecode scanners and binary code scanners as tools. Source code security analysers evaluate source code to detect weaknesses that can lead to a security vulnerability. Bytecode scanners examine the generated bytecode from the compiler to identify vulnerabilities and can be used in situations where the source code isn't public. Binary code scanners detect vulnerabilities through software disassembly and pattern recognition. In contrast, dynamic analysis tools examine the application during runtime looking for potential security flaws. Essentially, this means running the program under test loads and seeing what it does. These tools are used to supplement static analysis tools since static analysis tools, usually, report a high number of false-positives. [31].

### 6.2 Methodology

The WSO2 IS product maintains the source code under two GitHub organisations: wso2 and wso2-extensions. These two GitHub organisations contain code related to all WSO2 products, within hundreds of repositories. It is hard to find which jar file comes from where, so we used the Repo Explorer (rEx) [17] tool to address this problem.

We used rEx to download the master branch of all of the repositories related to WSO2 IS on 15 of June of 2020. There were no updates to the local source code after this date.

All the software auditing tools were used on a Windows 10 OS build 19041.450.

We will use static code scanning tools to find code vulnerabilities and discrepancies in programming code automatically. All scanners generally highlight potential security flaws. Each tool tested serves a particular function that is unique to itself.

For our study, we selected three well-known, publicly available bug-finding tools on PMD [16], FindBugs [5] [34], and Sonarqube Community [4]. We use three different static analysis tools because analysing the papers [26], [36], [43] results show that there is no correlation of warning counts between pairs of tools, this means that their results are unique compared to other tools and one package with a high number of warnings in one tool does not correlate to a high number of warnings in a different tool.

Another potential threat to validate is that we did not exactly categorise every false positive and false negative from the tools. Doing so would be extremely difficult, given a large number of warnings from the tools. Instead, in Section 6.4, we will look at raw numbers and analyse the system that way.

### 6.2.1 Repositories

WSO2 IS being a modular software solution, allows its source code to be divided into git repositories as individual modules. These Git repositories are currently all being hosted on Github and can be divided into three types:

- product-is is the main repository, and the only one needed to build the solution from the source code.
- identity and security repositories, this can be considered the major repositories containing the modules with the main features. They can be stand-alone feature repositories, integration tests for WSO2 IS, or product builds with build-scripts. The product-is repository could also be considered in this type of repositories instead of its individual type.
- extension repositories are mostly feature repositories used by WSO2 products.

The following Table 6.1 shows the major repositories for WSO2 IS and explains their purpose.

Table 6.1: Major WSO2 IS repositories

Repository URL	Description
<a href="https://github.com/wso2/product-is.git">https://github.com/wso2/product-is.git</a>	This repository is the only one needed to build the product from the source code. It has a maven configuration that builds the product based on its dependencies.
Continued on next page	



Table 6.1 – continued from previous page

<b>Repository URL</b>	<b>Description</b>
<a href="https://github.com/wso2/carbon-kernel.git">https://github.com/wso2/carbon-kernel.git</a>	WSO2 Carbon kernel can be viewed as a framework for server development. All the WSO2 products are created as a collection of reusable components operating on this kernel. These components inherit all the core services provided by Carbon kernel such as Registry/repository, User management, Transports, Caching, Clustering, Logging, Deployment related features.
<a href="https://github.com/wso2/carbon-commons.git">https://github.com/wso2/carbon-commons.git</a>	Carbon Commons repo includes the standard components and features shared by many WSO2 products.
<a href="https://github.com/wso2/balana.git">https://github.com/wso2/balana.git</a>	Balana is WSO2's open-source implementation of the XACML specification. Balana has an entitlement engine that allows externalising authorisation from the application. With its modular architecture, it is possible to achieve the fast development of a complete entitlement solution.
<a href="https://github.com/wso2/carbon-multitenancy.git">https://github.com/wso2/carbon-multitenancy.git</a>	This repository contains the multitenancy feature. The goal of multitenancy is to maximise resource sharing by enabling multiple users to log in and use a single server at the same time, in an isolated environment.
<a href="https://github.com/wso2/charon.git">https://github.com/wso2/charon.git</a>	WSO2 Charon is an open-source implementation of SCIM protocol which is an open standard for Identity Provisioning.
<a href="https://github.com/wso2/security-tools.git">https://github.com/wso2/security-tools.git</a>	This repository contains security-related tools developed by the WSO2 team or adapted for them.
<a href="https://github.com/wso2/carbon-identity-framework.git">https://github.com/wso2/carbon-identity-framework.git</a>	This repository contains the main components and features required by the WSO2 Identity Server.
Continued on next page	

Table 6.1 – continued from previous page

Repository URL	Description
<a href="https://github.com/wso2/carbon-security.git">https://github.com/wso2/carbon-security.git</a>	The Carbon Security project provides an authentication and authorisation implementation for the carbon products based on JAAS.
<a href="https://github.com/wso2/identity-feature-category.git">https://github.com/wso2/identity-feature-category.git</a>	This repository contains the maven configuration needed to build some of the identity-related features
<a href="https://github.com/wso2/carbon-identity-providers.git">https://github.com/wso2/carbon-identity-providers.git</a>	This repository manages the Identity Provider(IDP) and Service Provider(SP) implementations and related resources.
<a href="https://github.com/wso2/carbon-identity-commons.git">https://github.com/wso2/carbon-identity-commons.git</a>	Carbon Identity Commons combines the common components for all Carbon Identity components.
<a href="https://github.com/wso2/carbon-secvault.git">https://github.com/wso2/carbon-secvault.git</a>	WSO2 Secure Vault enables the storage of encrypted passwords mapped to aliases, i.e., meaning that it is possible to use the aliases instead of the actual passwords in configuration files for better security.
<a href="https://github.com/wso2/carbon-identity-gateway.git">https://github.com/wso2/carbon-identity-gateway.git</a>	A Proof of Concept implementation of the Identity Framework
<a href="https://github.com/wso2/carbon-auth.git">https://github.com/wso2/carbon-auth.git</a>	Carbon Auth is a standard authentication platform for Carbon-based products.
<a href="https://github.com/wso2/identity-test-integration.git">https://github.com/wso2/identity-test-integration.git</a>	This repository contains all the tests needed to verify a working WSO2 IS integration.
<a href="https://github.com/wso2/samples-is.git">https://github.com/wso2/samples-is.git</a>	This repository contains applications and guides that demonstrate the features of WSO2 Identity Server.
<a href="https://github.com/wso2/carbon-consent-management.git">https://github.com/wso2/carbon-consent-management.git</a>	This repository contains the consent management feature.
<a href="https://github.com/wso2/identity-anonymization-tool.git">https://github.com/wso2/identity-anonymization-tool.git</a>	A Tool for removing/Replacing all identifiers matching given criteria
<a href="https://github.com/wso2/identity-tools.git">https://github.com/wso2/identity-tools.git</a>	This contains some tools to help in WSO2 IS tasks.
Continued on next page	

Table 6.1 – continued from previous page

Repository URL	Description
<a href="https://github.com/wso2/identity-rest-dispatcher.git">https://github.com/wso2/identity-rest-dispatcher.git</a>	Aggregates the API implementations from identity-api-user and identity-api-server and creates a single web app in order to expose the various API endpoints in WSO2 Identity Server.
<a href="https://github.com/wso2/identity-api-user.git">https://github.com/wso2/identity-api-user.git</a>	Implementation of a user-related API
<a href="https://github.com/wso2/identity-api-server.git">https://github.com/wso2/identity-api-server.git</a>	Implementation of a server related API
<a href="https://github.com/wso2/identity-apps.git">https://github.com/wso2/identity-apps.git</a>	End-user apps in WSO2 IS
<a href="https://github.com/wso2/identity-media.git">https://github.com/wso2/identity-media.git</a>	This is the identity content provider service. This service can be used to upload, download and delete the images of a service provider(application), identity provider and user.

We will not go in-depth on extension repositories since their name is auto-explanatory and is representative of its features, however, for reference, a list of all the extension repositories can be found in Section [A.1](#).

### 6.2.2 Tools

FindBugs [5] is a bug pattern detector for Java. FindBugs uses a series of ad-hoc techniques designed to balance precision, efficiency, and usability. One of the main techniques FindBugs uses is to match source code to known suspicious programming practices syntactically.

Findbugs divides the type of reports (bugs) it finds in 9 different categories:

- Bad practice
- Correctness
- Experimental
- Internationalization
- Malicious code vulnerability
- Multithreaded correctness
- Performance
- Security
- Dodgy code

FindBugs ranks reports from 1 to 20, 1 being the highest and 20 the lowest. This rank measures the severity of the problem. Findbugs also gives us the likelihood that a bug was flagged as a real bug; this is called priority. Priority can also be called confidence level and can have five levels: High, Medium, Low, Experimental, Ignore.

PMD [16] performs syntactic checks on program source code, but it does not have a dataflow component. In addition to some detection of clearly erroneous code, many of the “bugs” PMD looks for are stylistic conventions whose violation might be suspicious under some circumstances.

PMD divides the type of bugs it finds in 8 different categories:

- Best Practices
- Code Style
- Design
- Documentation
- Error Prone
- Multithreading
- Performance
- Security

PMD ranks reports from 1 to 5, one being the highest and five the lowest. This rank is called priority. We can use the following guidelines to assert the priority of the bug:

- 1 - Change absolutely required. Behaviour is critically broken/buggy.
- 2 - Change highly recommended. Behaviour is quite likely to be broken/buggy.
- 3 - Change recommended. Behaviour is confusing, perhaps buggy, and against best practices.
- 4 - Change optional. Behaviour is not likely to be buggy.
- 5 - Change highly optional. Nices to have, such as a consistent naming policy.

SonarQube [4], is an automatic code review tool to detect bugs, vulnerabilities, and code smells in code. It can integrate with existing workflows to enable continuous code inspection across branches and pull requests. SonarQube scanner is called SonarScanner and is the only feature of SonarQube we will be using. SonarScanner divides its reports into three types: bugs, vulnerabilities, and code smells. Like the other tools, SonarScanner also has five levels of severity associated (Blocker, Critical, Major, Minor and Info).

## 6.3 Implementation

### 6.3.1 FindBugs

In this implementation, we used FindBugs version 3.0.1. To filter all the test cases and sample code from our results, we developed the following FindBugs filter:

```

1 <FindBugsFilter>
2
3   <Match>
4     <Class name="~.*\.*Test.*" />
5     <Class name="~.*\.*scenario.*" />
6     <Class name="~.*\.*sample.*" />
7   </Match>
8
9 </FindBugsFilter>

```

Listing 6.1: FindBugs filter

In order to run FindBugs we just need to run the following command:

```

1 java -jar .\findbugs.jar -effort:max -xml -sortByClass -exclude .\
   filter.xml -project .\project.xml -progress -medium -output
   output.xml

```

Listing 6.2: Command used to Run FindBugs

The project.xml file can be generated by the FindBugs UI when configuring the project.

### 6.3.2 PMD

In this implementation, we used PMD version 6.26. Because the number of reports from PMD is so high as we will see in Section 6.4, we decided to filter the results to only reports from security, error prone and multithreading categories. Additionally, we filtered all the test cases and sample code. This process resulted in the following PMD ruleset:

```

1 <?xml version="1.0"?>
2 <ruleset name="myruleset"
3   xmlns="http://pmd.sourceforge.net/ruleset/2.0.0"
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:schemaLocation="http://pmd.sourceforge.net/ruleset/2.0.0 https://pmd.
   sourceforge.io/ruleset_2_0_0.xsd">
6   <description>My ruleset</description>
7
8   <exclude-pattern>.*Test.*</exclude-pattern>

```

```
9 <exclude-pattern>.*sample.*</exclude-pattern>
10 <exclude-pattern>.*scenario.*</exclude-pattern>
11
12 <rule ref="category/java/errorprone.xml" />
13 <rule ref="category/java/multithreading.xml" />
14 <rule ref="category/java/security.xml" />
15
16 </ruleset>
```

Listing 6.3: PMD ruleset

In order to run PMD we just need to run the following command:

```
1 .\pmd.bat -d <path to Repositories> -f csv -R .\ruleset.xml -l java
   -no-cache -r output.csv
```

Listing 6.4: Command used to Run FindBugs

### 6.3.3 SonarQube

In this implementation, we used SonarQube version 8.4.1. As previously said, SonarQube is not just a scanner like the other tools, so we have to follow a simple project setup in order to scan the project. An example for this can be found in SonarQube's documentation [4].

In order to scan, and because this is a large project, we used the following commands, in order, inside the product-is repository:

```
1 mvn clean install
2 mvn sonar:sonar
```

Listing 6.5: Commands used to scan the project with SonarQube

## 6.4 Results

### 6.4.1 FindBugs

FindBugs resulted in 736 reports, all found in the 8 packages shown in Table 6.2. We can see type for report per package in Table 6.3 and Table 6.4. FindBugs reported 0 security type of reports. Besides, in these two tables, we can evaluate that the package `org.wso2.carbon.identity` has the most reports and that most reports are Performance type. The `org.wso2.carbon.identity` package has a significant number of lines of code and is used in many features of the WSO2 IS.

In Figure 6.1 we can see the number of reports per priority, evaluating this table we can conclude that reports are most in the Normal to Low Priority, meaning that reports, in general, have a medium to low confidence level that is not a false positive.

In Figure 6.2 we can see the number of reports per rank, evaluating this table we can conclude that reports are mostly in the 16 to 20 Rank, meaning that reports, in general, are not critical problems, and correspond to small corrections that should be made.

Table 6.2: Packages found to contain problems

ID	Package
P1	<code>org.wso2.balana</code>
P2	<code>org.wso2.carbon.identity</code>
P3	<code>org.wso2.carbon.sp.mgt.workflow</code>
P4	<code>org.wso2.identity.integration.common.clients</code>
P5	<code>org.wso2.identity.integration.common.ui</code>
P6	<code>org.wso2.identity.integration.common.utils</code>
P7	<code>org.wso2.identity.integration.common.extension.server</code>
P8	<code>org.wso2.identity.integration.ui</code>

Table 6.3: Findbugs results per package - part 1

Package/Type	Dodgy code	Multithreaded	Internationalization	Bad Practice
P1	47	12	11	4
P2	62	0	0	7
P3	0	0	0	0
P4	2	0	1	2
P5	0	0	0	1
P6	0	0	0	0
P7	0	0	0	0
P8	0	0	0	1
Total	111	12	12	15

Table 6.4: Findbugs results per package - part 2

Package/Type	Malicious code	Performance	Experimental	Correctness
P1	15	8	0	3
P2	0	399	0	12
P3	0	2	0	0
P4	1	120	0	0
P5	1	10	1	0
P6	0	4	0	0
P7	0	2	0	0
P8	1	6	1	0
Total	18	551	2	15

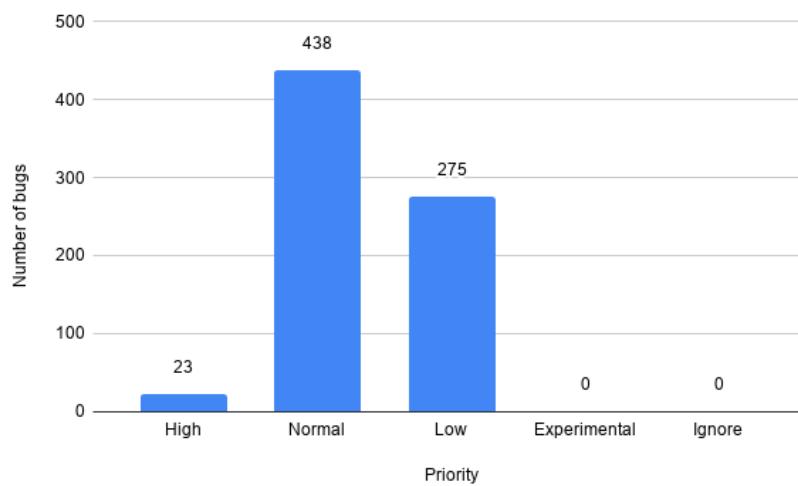


Figure 6.1: FindBugs number of reports per priority

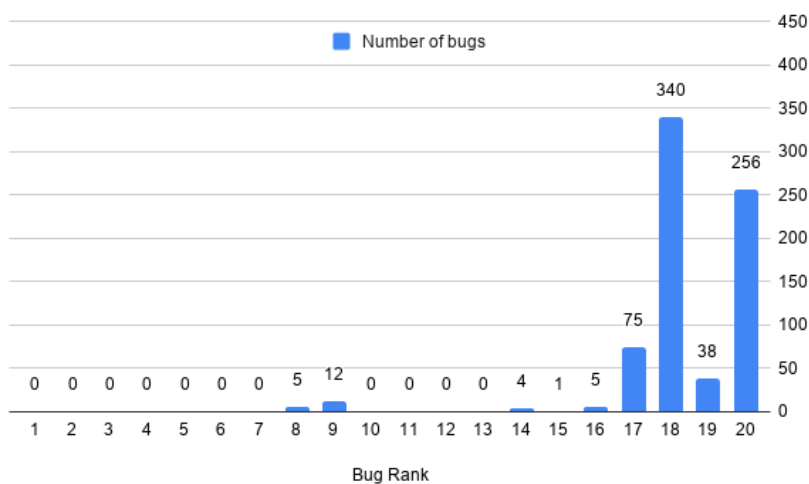


Figure 6.2: FindBugs number of reports per Rank



### 6.4.2 PMD

PMD resulted in 45456 reports, the top 10 packages with the most reports are shown in Table 6.5. As we previously said, we would only look at Error Prone, Multithreading and Security reports. PMD reported 0 Security issues, most of the reports being an Error Prone report. In Table 6.5, as we did in FindBugs results, we can see that the package org.wso2.carbon.identity has the most reports by far.

In Figure 6.3 we can see the number of reports per priority, evaluating this table we can conclude that reports are most in the 3 to 5 Priority, meaning that reports, in general, are not critical. However, 184 reports with Priority 1 is still a considerable amount.

Table 6.5: PMD top 10 packages with the most bugs

Package	Error Prone	Multithreading
org.wso2.carbon.identity	21725	1375
org.wso2.carbon.user	2434	147
org.wso2.carbon.registry	2161	147
org.wso2.balana	1340	56
org.wso2.charon3	1113	36
org.wso2.carbon.core	1074	115
org.wso2.security.tools and org.wso2.security.tool	908	59
org.wso2.identity	833	29
org.wso2.carbon.auth	636	50
org.wso2.carbon.is	615	36

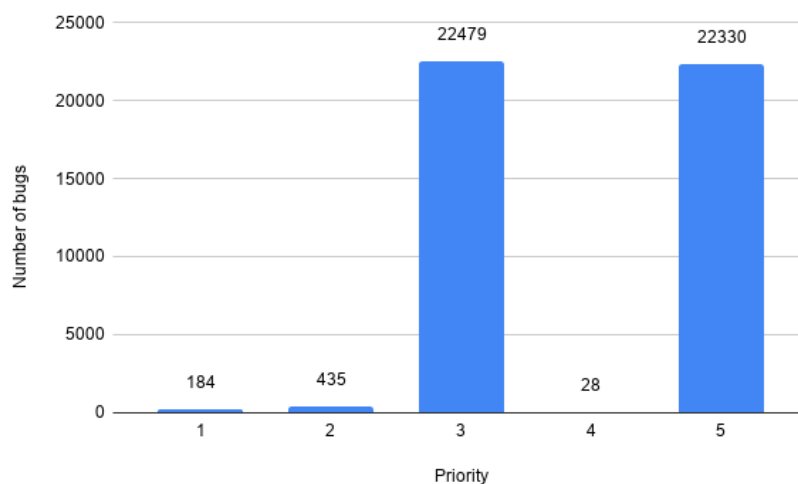


Figure 6.3: PMD number of reports per priority

### 6.4.3 Sonarqube

We can see SonarQube results summarised in Figure 6.4. However, after exporting this information with a SonarQube API, and filtering the results from tests, samples, and scenarios, we got 0 reports in WSO2 IS with SonarQube.

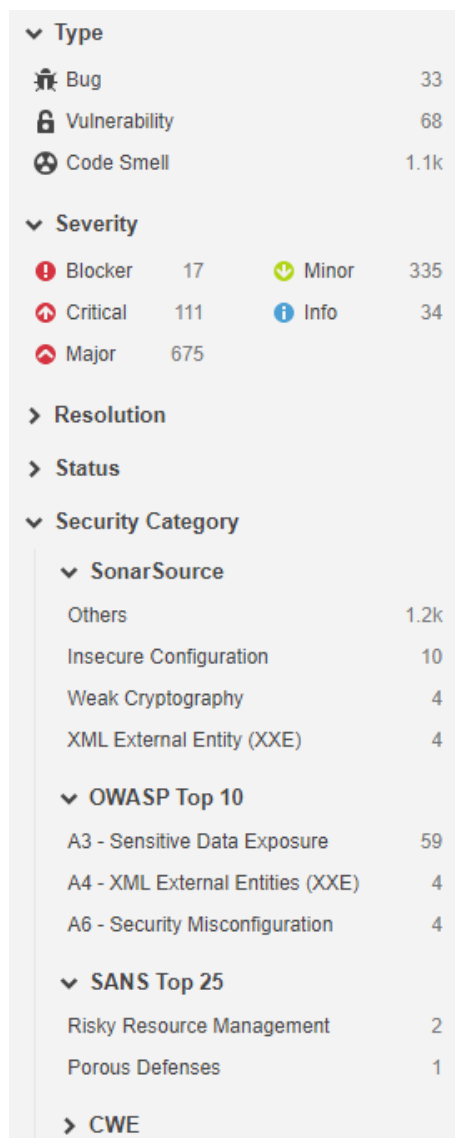


Figure 6.4: SonaQube results summarised

## 6.5 Discussion

We can see all the tools found 0 reports in a Security category in all three tools. These results are satisfactory; however, the 15 reports in FindBugs of Malicious code in org.wso2.balana, the package that implements XACML access policies, are concerning and should be reviewed. The

org.wso2.carbon.identity package also has somewhat worrying results, the sheer amount of reports in FindBugs and PMD, even if the types of reports do not imply the existence of bugs, should result in a review of this package.

Findbugs priorities are mostly between medium to low, so we can assume that there space for a lot of false positives. Findbugs has most of the reports ranks in 16 to 20; this indicates that most of the reports are not that concerning or alarming. PMD results further indicate this because almost half of the reports are of the priority 5 (Ignore priority ), and the other half are of priority 3 (not that concerning). However, 184 Priority 1 reports is still a concerning number, and these reports should be at least reviewed. SonarQube, having the scanner incorporated with the maven build, reported 0 problems after filtering for tests and samples. This further validates one of the conclusions of Chapter 5, WSO2 IS has all the tools to be a secure solution, and its security concerns are on the deployment and configuration of the overall product.

With these results, can we conclude that the WSO2 IS is a secure solution? No, even if all reports were accurate, it does not mean that a failure will occur or that a security breach is possible. However, a developer needs to be aware of common programming mistakes and how these flaws may compromise security. There will often be false positives for "flaws" that are not actually harmful, and false negatives where flaws go undetected. Static analysis tools serve as a helpful hand to remind and detect common vulnerabilities that may accidentally go unnoticed. However, source code analysers will not always point out and correct these issues. Developers should use these tools as an aid in the developmental process, but not depend on them. Creating a safe and functional code is the developer's task and cannot be replaced by static code analysis tools. Adopting secure coding techniques and regularly using various methods of security vulnerability detection will indeed reduce software security risks and improve efficiency for software developers.



## Chapter 7

# Conclusions

This dissertation discusses a plenitude of topics and approaches; In this section, we will revisit each one of these. We will revisit our problem, the obtained solutions to this problem, and the WSO2 IS Threat analysis and Security Audit. Additionally, we will present our conclusions, enumerate our contributions and future work ideas.

We first identified a common authentication problem in enterprises (Chapter 1). Systems using different protocols for communication means they process information differently, thus making these systems unable to communicate, IAM protocols being an excellent example of this problem. Different standard identity communication protocols were designed to solve this problem, SSO protocols being an excellent example of this. However, even though SSO protocols alleviate this problem, the different SSO protocols differ from each other, making the systems, again, unable to communicate.

Following the contextualisation of our problem, we reviewed what the state-of-the-art approaches are to solving this problem are (Chapter 3). We identified two types of solutions: A gateway able to translate protocols used as a middle-man in communication between two systems, and standard IAM solutions. We then understood that an Identity Protocol Translator is not a feasible solution and that solutions, like the WSO2 IS, solve our problem by unifying all these IAM features in a single place (Chapter 4).

WSO2 IS was the chosen IAM solution to proceed with our dissertation, now going forward with a Threat Analysis and a Security Audit with Static analysis tools (Chapters 5 and 6). This Threat Analysis revealed that WSO2 IS has all the tools in place to be a secure solution, but could be a target for attacks with a flawed system configuration. The Security Audit revealed that there are some problems to be reviewed, especially in `org.wso2.carbon.identity` and `org.wso2.balana` packages. A developer needs to be aware of common programming mistakes and how these flaws may compromise security.

In general, WSO2 IS is a well-thought solution, with all the state-of-the-art tools and protocols in place to be an extremely secure IAM solution, but this really depends on the deployment

configuration.

## 7.1 Contributions

All in all, we can highlight the following contributions:

- **Literature review**

We reviewed some open-source IAM solutions to understand the current state-of-art on IAM and FIdM.

- **WSO2 IS Threat analysis**

We conducted a Threat Analysis to the WSO2 IS solution, this can be used for further research on Threat Analysis methodologies and, as we recommend, future Threat Analysis made to specific WSO2 IS environments and deployments.

- **Securiy Audit using Static Analysis tools**

We gave one more example of how different Static analysis tools can produce different reports. We established that developers should use more than one of these tools as an aid in the developmental process. Furthermore, we found some concerning reports that should result in a code review.

## 7.2 Future Work

Our results are interesting, but only a first step into conceding the WSO2 IS as secure IAM solution. We believe many things can be considered for future work, namely:

- **Dynamic Analysis**

Dynamic analysis for the WSO2 IS to examine the application for potential security flaws during its runtime. This is expected to complement the static analysis tools.

- **Architecture review**

An architectural evaluation of security, evaluating the security of the WSO2 IS through its architectural design to expose the underlying security needs.

- **Penetration Testing**

A simulated cyber attack against a sample deployment of the WSO2 IS system to check for exploitable vulnerabilities. Information provided by the penetration test can be used to fine-tune security policies and patch detected vulnerabilities.

## Appendix A

# Appendix

### A.1 WSO2 IS Extension repositories

- <https://github.com/wso2-extensions/identity-outbound-auth-amazon.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-basecamp.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-dropbox.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-foursquare.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-github.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-instagram.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-inwebo.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-mailchimp.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-mepin.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-sms-otp.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-tiqr.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-wordpress.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-totp.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-x509.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-email-otp.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-yammer.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-inwebo.git>

- <https://github.com/wso2-extensions/identity-outbound-auth-passwordPolicy.git>
- <https://github.com/wso2-extensions/identity-extension-parent.git>
- <https://github.com/wso2-extensions/identity-agent-entitlement-proxy.git>
- <https://github.com/wso2-extensions/identity-agent-mobile-proxy-idp.git>
- <https://github.com/wso2-extensions/identity-agent-sso.git>
- <https://github.com/wso2-extensions/identity-local-auth-basicauth.git>
- <https://github.com/wso2-extensions/identity-local-auth-fido.git>
- <https://github.com/wso2-extensions/identity-local-auth-iwa-kerberos.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-oidc.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-openid.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-passive-sts.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-requestpath-basicauth.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-requestpath-oauth.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-saml-sso.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-iwa.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-mutual-ssl.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-saml2.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-signedjwt.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-thrift.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-oauth.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-openid.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-saml.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-sts.git>
- <https://github.com/wso2-extensions/identity-inbound-provisioning-scim.git>
- <https://github.com/wso2-extensions/identity-notification-email.git>
- <https://github.com/wso2-extensions/identity-notification-json.git>



- <https://github.com/wso2-extensions/identity-outbound-provisioning-google.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-salesforce.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-scim.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-spml.git>
- <https://github.com/wso2-extensions/identity-tool-samlssso-validator.git>
- <https://github.com/wso2-extensions/identity-user-account-association.git>
- <https://github.com/wso2-extensions/identity-user-workflow.git>
- <https://github.com/wso2-extensions/identity-user-ws.git>
- <https://github.com/wso2-extensions/identity-userstore-cassandra.git>
- <https://github.com/wso2-extensions/identity-userstore-ldap.git>
- <https://github.com/wso2-extensions/identity-userstore-remote.git>
- <https://github.com/wso2-extensions/identity-workflow-impl-bps.git>
- <https://github.com/wso2-extensions/identity-workflow-template-multisteps.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-linkedIn.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-twitter.git>
- <https://github.com/wso2-extensions/identity-oauth2-grant-jwt.git>
- <https://github.com/wso2-extensions/identity-data-publisher-oauth.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-facebook.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-google.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-windows-live.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-yahoo.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-symantecvip.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-mydigipass.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-reddit.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-office365.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-bitly.git>

- <https://github.com/wso2-extensions/identity-event-handler-notification.git>
- <https://github.com/wso2-extensions/identity-governance.git>
- <https://github.com/wso2-extensions/identity-event-handler-account-lock.git>
- <https://github.com/wso2-extensions/carbon-security-user-store-ldap.git>
- <https://github.com/wso2-extensions/carbon-security-login-module-jwt.git>
- <https://github.com/wso2-extensions/identity-endpoint-account-mgt.git>
- <https://github.com/wso2-extensions/identity-data-publisher-authentication.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-duo.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-duo.git>
- <https://github.com/wso2-extensions/identity-carbon-auth-rest.git>
- <https://github.com/wso2-extensions/identity-cloud.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-saml-cloud.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-token2.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-rsa.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-oidc-mobileconnect.git>
- <https://github.com/wso2-extensions/identity-userstore-onprem-agent.git>
- <https://github.com/wso2-extensions/identity-agent-onprem-userstore.git>
- <https://github.com/wso2-extensions/tomcat-extension-samlso.git>
- <https://github.com/wso2-extensions/identity-data-publisher-audit.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-cas.git>
- <https://github.com/wso2-extensions/identity-application-authz-xacml.git>
- <https://github.com/wso2-extensions/identity-extension-utils.git>
- <https://github.com/wso2-extensions/identity-metadata-saml2.git>
- <https://github.com/wso2-extensions/identity-inbound-provisioning-scim2.git>
- <https://github.com/wso2-extensions/identity-local-auth-iwa-ntlm.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-pinterest.git>

- <https://github.com/wso2-extensions/devicemgt-nfc-provisioning-android.git>
- <https://github.com/wso2-extensions/identity-oauth2-grant-kerberos.git>
- <https://github.com/wso2-extensions/identity-outbound-provisioning-scim2.git>
- <https://github.com/wso2-extensions/identity-client-scim2.git>
- <https://github.com/wso2-extensions/identity-oauth-addons.git>
- <https://github.com/wso2-extensions/identity-oauth-uma.git>
- <https://github.com/wso2-extensions/identity-x509-commons.git>
- <https://github.com/wso2-extensions/identity-api-appmgt.git>
- <https://github.com/wso2-extensions/identity-local-auth-limited-sessions.git>
- <https://github.com/wso2-extensions/identity-conditional-auth-functions.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-nuxeo.git>
- <https://github.com/wso2-extensions/identity-local-auth-api.git>
- <https://github.com/wso2-extensions/identity-fetch-remote.git>
- <https://github.com/wso2-extensions/identity-userstore-aws.git>
- <https://github.com/wso2-extensions/identity-client-lib-appauth.git>
- <https://github.com/wso2-extensions/identity-office365.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-casque-snr.git>
- <https://github.com/wso2-extensions/identity-migration-resources.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-cognito.git>
- <https://github.com/wso2-extensions/carbon-identity-saml-common.git>
- <https://github.com/wso2-extensions/identity-sdks-dotnet.git>
- <https://github.com/wso2-extensions/identity-samples-dotnet.git>
- <https://github.com/wso2-extensions/identity-application-authz-opa.git>
- <https://github.com/wso2-extensions/identity-tools-cli.git>
- <https://github.com/wso2-extensions/identity-tools-plugin-vscode.git>
- <https://github.com/wso2-extensions/identity-tools-debugger.git>

- <https://github.com/wso2-extensions/identity-sdks-android.git>
- <https://github.com/wso2-extensions/identity-samples-android.git>
- <https://github.com/wso2-extensions/identity-inbound-auth-jwtssso.git>
- <https://github.com/wso2-extensions/identity-samples-js.git>
- <https://github.com/wso2-extensions/identity-sdks-js.git>
- <https://github.com/wso2-extensions/identity-verification-evident.git>
- <https://github.com/wso2-extensions/identity-samples-spring-boot.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-naver.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-oauth2.git>
- <https://github.com/wso2-extensions/identity-outbound-auth-kakao.git>

# References

- [1] A Guide to WSO2 Identity Server. URL: <https://wso2.com/whitepapers/a-guide-to-wso2-identity-server/>. (Accessed on 06/08/2020).
- [2] Apache Syncope. URL: <https://syncope.apache.org/>. (Accessed on 08/24/2020).
- [3] CAS. URL: <https://apereo.github.io/cas/6.2.x/index.html>. (Accessed on 08/22/2020).
- [4] Code Quality and Security | SonarQube. URL: <https://www.sonarqube.org/>. (Accessed on 09/02/2020).
- [5] Findbugs™ - find bugs in java programs. URL: <http://findbugs.sourceforge.net/>. (Accessed on 09/02/2020).
- [6] Gluu Server 4.2 Docs. URL: <https://gluu.org/docs/gluu-server/4.2/>. (Accessed on 08/22/2020).
- [7] Identity and access management (iam) solutions | ibm. URL: <https://www.ibm.com/security/identity-access-management>. (Accessed on 08/24/2020).
- [8] Identity and access management – rsa securid suite. URL: <https://www.rsa.com/en-us/products/rsa-securid-suite>. (Accessed on 08/24/2020).
- [9] Keycloak. URL: <https://www.keycloak.org/>. (Accessed on 08/22/2020).
- [10] midpoint: Open source identity management software. URL: <https://evolveum.com/midpoint/>. (Accessed on 08/22/2020).
- [11] OECD Privacy Guidelines - OECD. URL: <http://www.oecd.org/internet/ieconomy/privacy-guidelines.htm>. (Accessed on 06/07/2020).
- [12] OpenAM 13.5. URL: <https://backstage.forgerock.com/docs/openam/13.5>. (Accessed on 08/22/2020).
- [13] OpenIDM 4. URL: <https://backstage.forgerock.com/docs/openidm/4/>. (Accessed on 08/22/2020).
- [14] Oracle identity governance home page. URL: <https://www.oracle.com/middleware/technologies/identity-management/governance.html>. (Accessed on 08/24/2020).
- [15] OWASP CSRFGuard. URL: <https://owasp.org/www-project-csrfguard/>. (Accessed on 09/16/2020).

- [16] PMD. URL: <https://pmd.github.io/>. (Accessed on 09/02/2020).
- [17] prabath/wso2is-repo-explorer: Repo explorer (rex) for wso2 identity server (is). URL: <https://github.com/prabath/wso2is-repo-explorer>. (Accessed on 09/02/2020).
- [18] Secure workforce identity with iam tools | okta. URL: <https://www.okta.com/workforce-identity/>. (Accessed on 08/24/2020).
- [19] Shibboleth. URL: <https://wiki.shibboleth.net>. (Accessed on 06/07/2020).
- [20] SimpleSAMLphp. URL: <https://simplesamlphp.org/>. (Accessed on 06/07/2020).
- [21] The Case for Open Source IAM. URL: <https://wso2.com/whitepapers/the-case-for-open-source-iam/>. (Accessed on 06/08/2020).
- [22] Unity IdM - Identity management and authentication (IAM) platform Unity IdM. URL: <https://www.unity-idm.eu/> (Accessed on 2020-02-02).
- [23] WSO2 Identity Server Documentation. URL: <https://is.docs.wso2.com/en/latest/>. (Accessed on 06/07/2020).
- [24] *Understanding authentication, authorization, and accounting*, chapter 2, pages 33–109. John Wiley Sons, Ltd, 2019.
- [25] Nadia Jemil Abdu and Ulrike Lechner. A threat analysis model for identity and access management. In *ICISSP*, 2016.
- [26] Hamda Hasan AlBreiki and Qusay H Mahmoud. Evaluation of static analysis tools for software security. In *2014 10th International Conference on Innovations in Information Technology (IIT)*, pages 93–98. IEEE, 2014.
- [27] Kim Cameron. The laws of identity. 2005. URL: <https://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf>. (Accessed on 06/10/2020).
- [28] David W Chadwick. Federated identity management. In *Foundations of security analysis and design V*, pages 96–120. Springer, 2009.
- [29] David W Chadwick. Federated identity management. In *Foundations of security analysis and design V*, pages 96–120. Springer, 2009.
- [30] Qingwen Cheng, Ping Luo, Andrew Patterson, and Rajeev Angal. Method and system for multi-protocol single logout, January 17 2012. US Patent 8,099,768.
- [31] C. Cowan. Software security for open-source systems. *IEEE Security Privacy*, 1(1):38–45, 2003.
- [32] ECB ECB. Recommendations for the security of internet payments. Technical report, Tech. Rep. January, 2013.
- [33] Dick Hardt et al. The OAuth 2.0 authorization framework. Technical report, RFC 6749, October, 2012.
- [34] David Hovemeyer and W. Pugh. Finding bugs is easy. *ACM SIGPLAN Notices*, 39:92–106, 2004.

- [35] John Hughes and Eve Maler. Security assertion markup language (saml) v2. 0 technical overview. *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, pages 29–38, 2005.
- [36] Rahma Mahmood and Qusay Mahmoud. Evaluation of static analysis tools for finding vulnerabilities in java and c/c++ source code. 05 2018.
- [37] R. Morgan, Scott Cantor, Steven Carmody, Walter Hoehn, and Ken Klingenstein. Federated security: The shibboleth approach. *EDUCAUSE Quarterly*, 27, 01 2004.
- [38] Michael Muckin. A threat-driven approach to cyber security methodologies , practices and tools to enable a functionally integrated cyber security organization. 2015.
- [39] Suvda Myagmar, Adam J. Lee, and William Yurcik. Threat modeling as a basis for security requirements. 2005.
- [40] N. Naik and P. Jenkins. Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect. In *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, pages 163–174, May 2017.
- [41] Frank Piessens, Bart De Decker, and Phil Janson. Interconnecting domains with heterogeneous key distribution and authentication protocols. In *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 66–79. IEEE, 1993.
- [42] Darren C Platt and Michael Scott Gile. Identity protocol translation gateway, May 2 2017. US Patent 9,641,512.
- [43] Nick Rutar, Christian B. Almazan, and J. Foster. A comparison of bug finding tools for java. *15th International Symposium on Software Reliability Engineering*, pages 245–256, 2004.
- [44] Nat Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. OpenID Connect Core 1.0 incorporating errata set 1. *The OpenID Foundation, specification*, 335, 2014.
- [45] STANDARDIZATION SECTOR and OF ITU. ITU-T. Baseline capabilities for enhanced global identity management trust and interoperability. Draft New Recommendation ITU-T X.1250.
- [46] STANDARDIZATION SECTOR and OF ITU. ITU-T. NGN identity management framework. Recommendation Y.2720.
- [47] Adam Shostack. *Threat modeling: Designing for security*. John Wiley & Sons, 2014.
- [48] Sean Simpson and Thomas Groß. A survey of security analysis in federated identity management. In *Privacy and Identity Management*, 2016.
- [49] Christopher Staite and Rami Bahsoon. Evaluating identity management architectures. In *Proceedings of the 3rd International ACM SIGSOFT Symposium on Architecting Critical Systems, ISARCS '12*, page 11–20, New York, NY, USA, 2012. Association for Computing Machinery.
- [50] Antonietta Stango, Neeli Prasad, and Dimitris Kyriazanos. A threat analysis methodology for security evaluation and enhancement planning. pages 262–267, 01 2009.

- [51] Dror Yaffe and Michael Gilfix. Multi-protocol authentication and authorization in computer network environments, April 13 2010. US Patent 7,698,443.
- [52] Dimitrios Zisis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation Computer Systems*, 28(3):583 – 592, 2012.