Architectural Design of a Conformative Authentication Service for Security Platforms

by

Mikael Hermansson

LIU-IDA/LITH-EX-A--13/004—SE

2013-01-31
Final thesis

Architectural Design of a Conformative Authentication Service for Security Platforms

by

Mikael Hermansson

LIU-IDA/LITH-EX-A–13/004-SE

2013-01-31

Supervisor: Ulf Kargén (IDA)
Jonas Elmqvist (Cybercom)

Examiner: Nahid Shahmehri
Abstract

Authentication services in security platforms often need to handle different types of systems which have various requirements regarding the authentication. These requirements can often interfere with each other and the issue here is that the authentication service often needs to be manually adjusted to comply with these requirements. Therefore there is a need for a flexible architectural design which enables changes and could open up for new emerging technologies and possibilities. This thesis presents an architectural design of a conformative authentication service based on SAML 2.0 to be used in security platforms. In this thesis a requirements analysis was performed and an architectural design was developed. The architectural design presented in this thesis is conformative in various aspects, e.g. usage of various authentication methods, versatile handling of attributes, handling of various SAML 2.0 profiles, possibilities to participate in various identity federations and handling of legacy systems not supporting SAML. In addition, an evaluation comparing the candidate architectural design presented in this thesis with a currently active architectural design was performed. This evaluation showed that the candidate architectural design was considered better for more usage scenarios.
Acknowledgements

First of all I would like to thank both Cybercom Group AB and the Department of Computer and Information Science (IDA) at Linköping University for giving me the opportunity to work with this thesis. I would especially like to thank my supervisor at Cybercom, Jonas Elmqvist, for his many ideas and support throughout this thesis. Then I would like to thank my examiner at IDA, professor Nahid Shahmehri, and my supervisor at IDA, Ulf Kargen, for their help with reviewing and improving this thesis. Finally, I would like to thank the other employees at Cybercom who have given valuable input to this thesis, especially Jonas Dehlin, Marcus Gustafsson, Åke Bengtsson and Magnus Nilsson.

Linköping, January 2013
Mikael Hermansson
Contents

1 Introduction 2
  1.1 Background ................................. 2
  1.2 Problem Description .......................... 3
  1.3 Objectives ................................... 4
  1.4 Method ....................................... 4
  1.5 Method Criticism ............................. 5
  1.6 Limitations ................................. 6
  1.7 Intended Audience ............................ 6
  1.8 Thesis Overview ............................ 6

2 Software Architecture 8
  2.1 Overview .................................... 8
  2.2 Architectural Structures and Views ............. 11
    2.2.1 Module Structures .......................... 11
    2.2.2 Component-and-Connector Structures .......... 11
    2.2.3 Allocation Structures ...................... 11
  2.3 Quality Attributes .......................... 12
  2.4 Architectural Styles .......................... 14
    2.4.1 Communicating Processes .................... 14
    2.4.2 Coupling-and-Cohesion ...................... 14
    2.4.3 Decomposition Style ........................ 15
    2.4.4 Information Hiding and Encapsulation ........ 15
    2.4.5 Layered Style .............................. 15
    2.4.6 Pipe-and-Filter Style ...................... 16
    2.4.7 Repository Styles ........................... 17
  2.5 ADL and UML ................................. 18
  2.6 Software Architecture Models and Methods ...... 18
    2.6.1 Attribute Driven Design ..................... 18
    2.6.2 ISO/IEC/IEEE 42010 ......................... 18
    2.6.3 Kruchten's 4+1 View Model ................... 19
    2.6.4 RM-ODP .................................. 21
  2.7 Evaluating Software Architectures ............... 21
    2.7.1 ALMA .................................... 22
    2.7.2 ATAM .................................... 22
CONTENTS

2.7.3 SAAM ........................................ 22
2.7.4 SBAR ........................................ 24

3 Security Architecture ........................................ 25
  3.1 Introduction to Security Architecture .................. 25
  3.2 Security Goals in Architectures ........................ 26
  3.3 Security Considerations ............................... 26
  3.4 Secure Design Patterns ............................... 26
    3.4.1 Directory .................................. 27
    3.4.2 Distributor .................................. 27
    3.4.3 Interceptor .................................. 27
    3.4.4 Layers ...................................... 27
    3.4.5 Trusted Third Party ......................... 28
  3.5 Software Architecture Security ....................... 28

4 Identity Management ........................................ 29
  4.1 Digital Identity Management .......................... 29
  4.2 Stakeholders ...................................... 30
  4.3 Credentials ...................................... 31
  4.4 Authentication and Authorization ...................... 31
  4.5 Single Sign-On .................................... 32
  4.6 Federated Identity Management ....................... 32

5 Electronic Authentication ................................... 34
  5.1 Overview ........................................ 34
  5.2 Tokens .......................................... 35
  5.3 Electronic Credentials .............................. 36
  5.4 Assertions ....................................... 36
  5.5 Authentication Methods .............................. 37
    5.5.1 Password Authentication ...................... 37
    5.5.2 One-time Password Authentication ............ 37
    5.5.3 Hardware Token Authentication .............. 38
    5.5.4 Software Token Authentication .............. 38
  5.6 Authentication Assurance Levels ..................... 38
  5.7 Authentication Protocols ............................ 38
  5.8 Authentication Threats ................................ 39
    5.8.1 Token Threats ................................. 39
    5.8.2 Authentication Protocol Threats ............. 40
    5.8.3 Additional Authentication Threats .......... 40

6 SAML .................................................... 41
  6.1 Introduction to SAML .................................. 41
  6.2 SAML Participants ................................... 42
  6.3 SAML Architecture ................................... 43
  6.4 SAML Assertions ................................... 44
  6.5 SAML Protocols .................................... 45
# CONTENTS

6.6 SAML Bindings ................................................. 46  
6.7 SAML Profiles ................................................. 47  
  6.7.1 Attribute Profiles ...................................... 47  
6.8 Security and Privacy in SAML ................................. 48  

7 Requirements Analysis ........................................ 49  
  7.1 Introduction ................................................. 49  
  7.2 Description of CESP ......................................... 50  
  7.3 Requirements List ........................................... 52  
    7.3.1 Internal Requirements ................................. 53  
    7.3.2 Customer requirements ................................. 54  
  7.4 Additional Security Requirements .......................... 56  
  7.5 OWASP Top 10 Analysis .................................... 59  
  7.6 Use Cases .................................................... 62  
    7.6.1 Use Case 1: Emergency and Rescue Enterprise ....... 62  
    7.6.2 Use Case 2: Umbrella Organization .................. 63  
    7.6.3 Use Case 3: Banking Service ......................... 64  
    7.6.4 Use Case 4: Company Federations .................... 64  
    7.6.5 Use Case 5: Large Identity Federations ............ 65  

8 Architectural Design ......................................... 68  
  8.1 Introduction ............................................... 68  
  8.2 Method and Rationale ..................................... 68  
  8.3 View Catalog .............................................. 69  
  8.4 System Overview .......................................... 70  
  8.5 Logical View ................................................ 71  
    8.5.1 Primary Presentation .................................. 71  
    8.5.2 Element Catalog and Context Description ........... 71  
    8.5.3 Rationale .............................................. 73  
  8.6 Process View ................................................ 73  
    8.6.1 Primary Presentation .................................. 73  
    8.6.2 Element Catalog and Context Description ........... 73  
    8.6.3 Rationale .............................................. 80  
  8.7 Development View .......................................... 82  
    8.7.1 Primary Presentation .................................. 83  
    8.7.2 Element Catalog and Context Description ........... 83  
    8.7.3 Rationale .............................................. 98  
  8.8 Physical View ............................................... 99  
    8.8.1 Primary Presentation .................................. 99  
    8.8.2 Element Catalog and Context Description ........... 99  
    8.8.3 Rationale .............................................. 101  
  8.9 Scenarios .................................................. 101  
    8.9.1 Primary Presentation .................................. 101  
    8.9.2 Element Catalog and Context Description ........... 101  
    8.9.3 Rationale .............................................. 104  
  8.10 Discussion ................................................ 105  

ix
## 9 Evaluation

9.1 Method and Rationale ................................................. 106
9.2 Prerequisites ......................................................... 107
9.3 Results ............................................................... 107
9.3.1 Describe Candidate Architectures ......................... 107
9.3.2 Develop Scenarios .............................................. 108
9.3.3 Perform Scenario Evaluations ............................... 109
9.3.4 Overall Evaluation ............................................. 119

## 10 Conclusions and Future Work  

10.1 Conclusions ......................................................... 121
10.2 Future Work ......................................................... 122
# List of Figures

1.1 Method Overview ........................................... 4  
2.1 Architectural concepts ................................. 10  
2.2 Layered Style ............................................. 15  
2.3 Layers With Sidecar ..................................... 16  
2.4 Pipe-and-Filter Style ................................. 16  
2.5 Shared-Data Style ...................................... 17  
2.6 Kruchten's 4+1 View Model .......................... 19  
2.7 SAAM Process Model ................................. 23  
5.1 Authentication Overview ............................... 35  
6.1 Example of SAML Participants ....................... 43  
6.2 SAML Concepts ......................................... 44  
7.1 CESP Components ...................................... 50  
7.2 Use Case 1, Emergency and Rescue Enterprise .... 63  
7.3 Use Case 2, Umbrella Organization ................. 64  
7.4 Use Case 3, Banking Service ......................... 65  
7.5 Use Case 4, Company Federations ................... 66  
7.6 Use Case 5, Large Identity Federations ............ 67  
8.1 System Overview ....................................... 70  
8.2 Logical View ........................................... 71  
8.3 Process Blueprint ...................................... 74  
8.4 Request and User Interface Process Flow .......... 75  
8.5 Validation Process Flow ............................. 76  
8.6 Configuration Process Flow ......................... 77  
8.7 Logging Process Flow ................................. 78  
8.8 Session Process Flow ................................. 79  
8.9 Response Generator Process Flow .................. 80  
8.10 Authentication Core Process Flow ................. 81  
8.11 Authentication Method Process Flow ............... 82  
8.12 Legacy Process Flow ............................... 83
LIST OF FIGURES

8.13 Attribute Process Flow . . . . . . . . . . . . . . . . . . . . . . . . . 84
8.14 Federation Process Flow . . . . . . . . . . . . . . . . . . . . . . . . . 85
8.15 Development View . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
8.16 Request and User Interface Module . . . . . . . . . . . . . . . . 87
8.17 Validation Module . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
8.18 SAML Profile Module . . . . . . . . . . . . . . . . . . . . . . . . . 88
8.19 Configuration Module . . . . . . . . . . . . . . . . . . . . . . . . . 89
8.20 Contents of Configuration Module Database Submodule . . . . 90
8.21 Authentication Core Module . . . . . . . . . . . . . . . . . . . . 91
8.22 Session Module . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 91
8.23 Authentication Method Module . . . . . . . . . . . . . . . . . . 92
8.24 Attribute Module . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
8.25 Attribute Processing Module . . . . . . . . . . . . . . . . . . . . 94
8.26 Federation Module . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
8.27 Legacy Module . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 96
8.28 Logging Module . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
8.29 Response Generator Module . . . . . . . . . . . . . . . . . . . . 98
8.30 Physical View . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100
8.31 Scenario 1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 102
8.32 Scenario 2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 103
List of Tables

7.1 Internal requirements ........................................ 54
7.2 Customer requirements ....................................... 56
7.3 OWASP Top 10 Overview ..................................... 60

9.1 List of Scenarios ............................................. 109
9.2 Scenario Evaluation for Current Active Architecture .... 114
9.3 Scenario Evaluation for Candidate Architecture .......... 118
9.4 Evaluation Summary Table .................................. 120
9.5 Evaluation Results Table .................................... 120
LIST OF TABLES
Chapter 1

Introduction

This chapter is an introduction to a master thesis (30 credit points) final report. The thesis was examined at the Department of Computer and Information Science (IDA) at Linköping University and is the final part of a 5 year degree leading to a Master of Science in Computer Science and Engineering. This thesis has been performed at Cybercom Group AB in Linköping. Cybercom Group AB is an IT company that delivers high quality business solutions, primarily in telecom management, connected devices, Internet services and security.

1.1 Background

There are a large amount of systems where security is of great importance. In these systems the confidentiality, integrity and availability of the internal applications and data needs to be assured. Typically this is achieved by implementing the system in a closed network environment where external users have no access to the system.

The increasing demand for service oriented applications has led to a need for opening up these systems to offer services to external users. This is not a trivial task if the security level of the system shall be preserved. Therefore there is a need for a security platform that handles the communication with the internal applications without compromising the confidentiality, integrity and availability of the internal applications and data.

To meet this need, Cybercom Group AB has developed an integration and security platform called Cybercom Enhanced Security Platform, CESP, which offers features such as authentication, access control and logging. Most customers use the platform to enable secure access to their systems. The platform is based on SAML 2.0, which is a standard for exchanging authentication and authorization data.

Authentication is needed to make sure that only trusted users can gain access to the system. The authentication service in security platforms like
1.2. PROBLEM DESCRIPTION

CESP needs to be able to handle different authentication methods and different kind of actors such as individuals or services. The authentication service needs to assure that the user has been identified, authenticated and assigned different attributes (characteristics which describe an identity, e.g. full name and date of birth). An assertion, which is a signed proof of this, is generated by the authentication service. This assertion can then be used to gain access to applications instead of performing a new authentication each time and also enables single sign-on between applications and organizations.

Many organizations also want to enable users from other organizations that they trust to securely access their internal applications and data without the need for additional authentications. This is called identity federations, which is an emerging technology that can offer both convenience and economic advantages, e.g. more efficient access management and reduced administration costs.

1.2 Problem Description

A security platform like CESP needs to be able to handle various types of systems and these systems often have a variety of different requirements regarding authentication which often can interfere with each other. Examples of such differing requirements could be different types of authentication methods, different ways to handle user attributes or compliance with different SAML profiles. The problem with this is that the authentication service often needs to be manually adjusted for each system which can lead to lots of manual work. These manual adjustments could also lead to the risk of introducing errors and possibly also increased security risks if the authentication service was not originally designed for handling these requirements. A more flexible authentication service could also open up for new emerging technologies and possibilities. To meet new customer requirements and increasing demands for flexibility but at the same time comply with the standards, there is a need to revise the architecture of the authentication service in CESP.

The thesis addresses the following problems:

• How can an architectural design of a conformative authentication service be developed based on a given set of requirements?

• How can versatile authentication be enabled in the same architectural design?

• Are there any problems with combining different authentication methods in the same authentication service?

• How can software architectures be evaluated?
CHAPTER 1. INTRODUCTION

1.3 Objectives

The objective of the thesis is to examine the possibility of designing a con-
formative authentication service from a given set of requirements without
compromising the security of the system. An evaluation of the architecture
shall also be performed.

1.4 Method

The work in this thesis was divided into four main phases: literature studies,
requirements analysis, architectural design and evaluation. An overview of
the thesis method structure is shown in Figure 1.1.

![Figure 1.1: Method Overview](image)

The first phase, literature studies was performed to get the background
information needed to create the architectural design for the authentica-
tion service. The literature studies were divided into five areas: Software
Architecture, Security Architectures, Identity Management, Electronic Au-
thentication and SAML.

The second phase of this thesis was the requirements analysis which
was performed to elicit and analyze the requirements for the architectural
design of the authentication service. The requirements analysis was per-
formed using semi-structured interviews with Cybercom employees working
with CESP. The results from the semi-structured interviews were summa-
rized as a requirements list and a set of use cases. The requirements list
was in addition evaluated by two Cybercom customers to make sure that
it conformed to their requirements. To gain an additional security perspec-
1.5 Method Criticism

The requirements elicitation was performed using semi-structured interviews with Cybercom employees. However, to provide a more complete picture of the requirements, all authentication service stakeholders should take part in the elicitation. During the requirements phase, to get an additional security perspective on the requirements the security requirements engineering framework described in [22] was used. However, all parts of this framework were not used and this could cause some aspects to be missing. A smaller framework would have been more suitable for the purpose of getting an additional security perspective of the authentication service requirements.

Kruchten’s 4+1 View Model [32] was used as a base for the architectural design. This model contains information about how to describe the correspondence between the different views. However, this part of the model was omitted. Excluding this part of the model could result in some aspects of the architectural design not being covered.

To perform the evaluation SAAM [30] was used. Using this model, all stakeholders should participate in the evaluation. During the evaluation in this thesis, only Cybercom employees working with CESP and the authentication service could participate. To get a more complete evaluation, all authentication service stakeholders (e.g., the customers) should have been participating in the evaluation. Another aspect which affects the result of the evaluation is that the scenarios covers requirements which were not known when the current active architecture was designed. This implies that the candidate architecture presented in this thesis have a great advantage compared to the current active architecture. Another evaluation approach which does not focus on these requirements would have been a more fair
CHAPTER 1. INTRODUCTION

approach. In addition, SAAM is a rather extensive model, a smaller model could have been more suitable for evaluating the architectures.

1.6 Limitations

This thesis only focuses on using SAML 2.0 for the exchange of authentication data. There are several other alternatives (e.g. OpenID and OAuth), but they have not been considered in this thesis. This thesis only covers the architectural design of an authentication service. Implementing the design is out of scope of this thesis since there is limited time.

1.7 Intended Audience

This thesis is intended for an audience with a reasonable background in computer science, software engineering or similar. Some basic knowledge in the area of information security is also advised.

1.8 Thesis Overview

Chapter 1: Introduction gives a description of the background and the purpose of this thesis.

Chapter 2: Software Architecture gives an introduction to software architecture, presents architectural styles and describes software architecture evaluations.

Chapter 3: Security Architecture presents a description of security architectures and various considerations and patterns useful for designing secure software architectures.

Chapter 4: Identity Management describes what digital identity management is all about and various considerations regarding digital identity management. The chapter also introduces concepts such as single sign-on and identity federation management.

Chapter 5: Electronic Authentication presents a description of what electronic authentication is and describes various concepts related to electronic authentication.

Chapter 6: SAML describes the Security Assertion Markup Language (SAML), what it is all about and how it is structured.

Chapter 7: Requirements Analysis gives a description of how the requirements analysis was performed and presents the results of the
1.8. THESIS OVERVIEW

requirements analysis.

Chapter 8: Architectural Design presents the architectural design of the authentication service.

Chapter 9: Evaluation describes the architectural evaluation that was performed and the results of the evaluation.

Chapter 10: Conclusions and Future Work gives conclusions drawn from this report and discusses future work related to this report.
Chapter 2
Software Architecture

This chapter gives an introduction to software architectures and various related concepts, e.g. quality attributes, structures and views. Various architectural styles are presented and a description of software architecture evaluation is given.

2.1 Overview

A software architecture can be seen as a high-level description of a software system which describes the elements in the software and their relationships. There exist lots of different definitions of software architecture since it is a growing and rather young discipline. In [5] software architecture is given the following definition: “The software architecture of a program or a computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationship among them.”. This definition implies that the architecture defines the elements in the software and how they relate to each other and does not include things that do not concern their interaction. In other words, the architecture is an abstraction which only includes details about how elements use, are used, relate or interact. In most cases elements interacts using interfaces. Interfaces divide the information about an element into a public and a private part. The architecture is only concerned with the public part, the private parts concerning internal implementations are not part of the architecture at all. [5]

Another definition of software architecture is given by the IEEE 1471-2000 standard [26]: “The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.” This definition emphasizes that software architecture is not only a description of the system at a certain point but also includes principles guiding the design and evaluation of the software system.
2.1. OVERVIEW

Software architectures are highly coupled to the organizational and business context in which they are used and to the business and organizational requirements. The desire to satisfy the organizational and business goals highly affects the design decisions. Software architecture flows from the experiences in the organization and the experiences that the architect has gathered. The software architecture is a development product that gives high return on investment in the form of quality, schedule and cost. This is since it is the foundation for the rest of the product’s lifecycle which includes development, integration, testing and modification. Analyzing the architecture and finding errors early in the product life cycle is more effective compared to the other development activities since the architectural decisions affect so many other parts of the product’s lifecycle, and if it is later shown that the architecture was wrong then it cannot be changed so easily. Architecture also sets the foundation for re-use, not only for re-use of components but also re-use of the whole architecture which can lead to families of products being developed and new business opportunities. [5]

What is the difference between design and architecture? Architecture is design but not all design is architecture. Many design decisions are left out of the architecture to be decided by designers and implementers [15]. The architecture creates constraints for the following activities which creates the details of the system. The main purpose of the architecture is to make sure that the quality and behavioral requirements are satisfied and that the business goals are fulfilled [15]. These requirements and goals are achieved in the architecture by making architectural decisions about the system structure. Compared to design, the architecture is often concerned with issues of longer range while design is more about producing the right answer and the expected functionality [5]. The architecture is often faced with lots of competing and conflicting influences and demands from both the organizational and technical environment which often aren’t even concerned with the expected functionality. These demands are often as important as the explicit demands even though they are seldom written down [5]. There are many stakeholders that influence the architecture and they often have different concerns which are important for them and sometimes these concerns can be conflicting [5]. These conflicting concerns are something that the architect will need to handle.

To elicit the requirements various techniques can be used, for example use cases. Often it is not possible to include all requirements from the stakeholders and tradeoffs will have to be made. Sometimes certain requirements are not noticed until parts of the architecture are constructed and then additional tradeoffs may have to be made.[5]

Software architecture is often used to separate the overall structure of the system, often in terms of subsystems and interfaces which are separated from the internal details of the subsystems [21]. A system consists of several structures and no single part can be said to be the architecture alone [5]. A structure could for example be a system partitioned into implementation
CHAPTER 2. SOFTWARE ARCHITECTURE

units which means that the focus lies on the system’s functionality. Another structure could be one which describes how elements interact with each other during runtime which has focus on how the functions are carried out. A single structure can’t be said to be the architecture alone unless the system is trivial [5]. A structure conveys information about the architecture, which consists of a combination of several structures. This implies that the architecture can consist of various types of elements and various types of interactions and also various types of contexts. The behavior of the elements that can be observed by other elements is also said to be a part of the architecture [5].

Between a simple sketch and a complete architecture are several stages which can be very useful. Some important concepts regarding this are: reference model, architectural pattern and reference architecture. The relations between these architectural concepts are shown in Figure 2.1.

![Figure 2.1: Architectural concepts](image)

Architectural patterns are common descriptions of elements, their relations and constraints describing how they shall be used. A common example of an architectural pattern is the client-server pattern. Innumerable architectures that use the client-server pattern exists, but they are all different from each other. Architectural patterns often bring certain quality attributes, for example performance or availability which is often the reason why a particular pattern is chosen. [5]

A reference model divides a problem into parts that cooperate to solve the assignment. The reference model divides both the functionality and the dataflow. Common examples that have useful reference models are for example compilers and database management systems. [5]

Reference architectures map a reference model onto software elements and their data flows. Here the different software elements cooperate to implement the functionality and a software element can include parts of the functionality or several functions. [5]
2.2 Architectural Structures and Views

Modern systems are often complex and difficult to understand as a whole and therefore we most often focus our attention to one or a small number of the systems’ structures. This means that to be able to communicate appropriately about the architecture there is a need to clarify which structures we are communicating about and which view of the architecture that is currently discussed. Having multiple views in an architecture helps stakeholders handle system complexity by separating the various concerns into separate views [3]. The terms structure and view are related and are often used interchangeably. According to [5] a view consists of a representation of a set of elements and the relations among them as they are written and read by the stakeholders. A structure on the other hand is the elements itself as they actually exist. Architectural structures can in general be divided into three groups: module structures, component-and-connector structures and allocation structures.

2.2.1 Module Structures

In a module structure the elements are modules which represent areas of functional responsibility. The elements are units of implementation and represent a code-based way of viewing the system. A module structure also describes the relations to other modules, which other modules that can be used and which other parts of the system that each module uses. Some common module structures are: decomposition, uses, layered and class [5].

2.2.2 Component-and-Connector Structures

In a component-and-connector structure the elements are runtime components and connectors [5]. Connectors handle the communication among components. Components are often described as self-contained objects with well-defined interfaces that can be used in various environments and connectors are used to join components and to encapsulate protocols between the components [21]. The component-and-connector structures describe which the major executing components are and their interactions [5]. They also describe how data is moved through the system and different parts that can be run in parallel. Some common component-and-connector structures are: client-server, concurrency, process and shared data [5].

2.2.3 Allocation Structures

Allocation structures show the relationship between the elements and the external elements in the environment where the software is created and executed. The structures shows things like which processor an element executes on, which files that store the elements during the development process and
CHAPTER 2. SOFTWARE ARCHITECTURE

how elements are assigned to development teams. Some common allocation structures are: deployment, implementation and work assignment. [5]

2.3 Quality Attributes

For most systems quality attributes like security, performance, availability, scalability etc. are as important as making sure that the system performs the correct actions. For example, results could be useless if it takes too long time to perform the actions or if the results are revealed to the competitors [15]. These quality attributes are typically addressed by the architecture but must be considered throughout the whole development process [5]. No quality attributes are exclusively dependent on the architectural design or exclusively dependent on the implementation. The quality attributes are concerned making sure that both the big picture and the details are correct. This means that architecture is critical to achievement of quality attributes but architecture itself is not capable of achieving the quality attributes, it only provides the foundation [5].

There exists a lot of different definitions of various quality attributes [5] and this poses some problems from an architects point of view. The definitions are not complete, e.g. when reasoning about a quality attribute like modifiability one cannot state that an architecture is entirely modifiable. The architecture can be modifiable with respect to a specific set of changes but not modifiable to another set of changes. Another problem which the different definitions brings is that the communities which developed the definitions often have their own vocabulary, which means that the same occurrence can be using different names [5]. The lack of standardized definitions can therefore cause misunderstandings and communication issues. To avoid these issues it is important to make sure to document the architecture in a comprehensive way. When documenting software architecture it is important to make sure to describe the quality attributes that drove the design, the decisions made to meet these quality attributes and arguments about the quality attributes [15]. The architecture documentation will serve as a basis for analysis which makes it possible to determine if quality attributes are achieved. Quality attributes can be shown in the architecture in various ways, e.g. the major design decisions and architectural styles used have quality attributes associated with them [15].

Some of the most common quality attributes are: availability, modifiability, performance, security, testability, usability and scalability. Some of these quality attributes can be both software quality and system quality attributes since both hardware and software could be needed to achieve them [21].

Availability is concerned with system failure and the results of a system failure. System failures occur when the system is not providing the services it should provide according to it’s specification and can affect both the users and other systems [5]. Availability is about how failures can be
2.3. QUALITY ATTRIBUTES

detected and prevented, about how long it is acceptable for a system to be out of service and what should happen when a failure occurs. System availability can be increased by using several tactics which mainly are divided into three categories. The categories are fault detection (includes, e.g. ping/echo, heartbeat and exceptions), fault recovery (includes e.g. voting, active/passive redundancy and shadow operation) and fault prevention (includes e.g. transactions and process monitors) [5].

Modifiability is about the ability to make changes to a software after it has been deployed. There exist various reasons for changing the software, e.g. correct remaining errors, fix performance issues and changes in requirements [21]. The main concerns of modifiability is: what can change, when is the change made an who makes it [5]. Unexpected requirements changes could require large modifications and therefore the software should be designed for changes and take evolution into account [21]. To increase the system modifiability several tactics are presented in [5], these tactics are split into three categories. These categories are: localize modifications (includes e.g. anticipate expected changes and generalize the module), prevent ripple effects (includes e.g. information hiding and restricted communication paths) and defer binding time (includes configuration files and component replacement) [5].

Performance is generally concerned with how long it takes for the system to respond when an event occurs [5]. Various performance goals exists, e.g. throughput and response time [21] and these goals can be measured at design time using various methods. Performance is however generally considered to be rather complicated to analyze and measure since there exists so many various events and arrival patterns, e.g. events can occur by user inputs, inputs from other systems and within the system. When an event is received the system either processes that event or the processing is blocked, which leads to two types of response time usages: resource consumption and blocked time [5]. To enhance the performance the time of these activities needs to be reduced.

Security is a quality attribute that measures the systems ability to resist unauthorized usage. Attacks are attempts to breach security, e.g. unauthorized data access or modification or denial of service attacks. According to [5] security can be characterized as providing nonrepudiation, confidentiality, integrity, assurance, availability and auditing. There exist various tactics for achieving security, and generally they can be divided into three main categories: resisting attacks, detecting attacks and recovering from attacks [5].

Testability is about to which extent a software is testable [21] and is often described by how easy it is to demonstrate the softwares faults through testing. To be able to test software it is essential that it is possible to control the internal states, the inputs and to be able to observe the output [5]. Tactics for achieving testability have the goal of making testing easier and includes among others: management of input & output and internal
monitoring.

Usability considers how easy it is for the user to accomplish a task and the level of support that the system provides for this task [5]. Several usability tactics exists and the one which is most closely related to the architecture is to separate the user interface from the rest of the system to provide easier modification [5] of the user interface to meet the users demands and needs.

Scalability is about how easy it is for the system to grow after initial deployment [21]. Scalability is concerned with both system and software issues. From a system perspective, a centralized system limits the scalability scope while a distributed system offers possibilities for much more scalability. From the software view the architecture need to be designed to help making growth possible. Using a distributed component-based approach helps to increase scalability compared to a centralized design [21].

2.4 Architectural Styles

According to Garlan and Shaw [19] an architectural style defines a family of systems of components and connectors in terms of their structural organization. Architectural styles are often considered to be the building blocks of software architectures and includes a description of the components and connectors that can be used, various constraints and a description of benefits and drawbacks [15].

2.4.1 Communicating Processes

Communicating Processes is a type of component-and-connector structure where each component can execute as a separate process [15]. There exist several variants of this style. One of the most common variants for systems using substructures is to run the top-level components as separate processes and let the subcomponents execute in the parent’s process [15]. Other variants are e.g. using threads instead of processes. Communicating processes can often be used to describe which portions of the system that can work in parallel and to analyze reliability and performance.

2.4.2 Coupling-and-Cohesion

The notations of coupling and cohesion was introduced by Stevens et al. [45] in the context of structured design when dividing computer systems into modules. Coupling is concerned with the degree that one module relates to the other modules and cohesion is concerned with the degree of which the elements of a module belongs together. Coupling and cohesion are often used as metrics to evaluate software systems. Typically low coupling between the modules and high cohesion within the modules are desired and considered a property of a well-structured software system and good design.
2.4. ARCHITECTURAL STYLES

2.4.3 Decomposition Style

The Decomposition Style consists of modules (the components) and the decomposition (a type of part-of relation) relations among them. The Decomposition Style is used for decomposing a system into its implementation structure and organizes the system into modules and submodules with individual responsibilities. Various criteria are used for decomposition of a module into submodules, e.g. the information-hiding design principle encapsulates changeable parts of the system into individual modules to enhance modifiability [15]. The constraints put on the Decomposition Style is that modules can only have single parents. The Decomposition Style presents the responsibilities of a subsystem in a system and can be used for e.g. work assignment and analyzing the consequences of changes.

2.4.4 Information Hiding and Encapsulation

The principle of encapsulation and the principle of information hiding are often used interchangeably and refers to method of hiding the internal representation of a class or a module to shield it from misuse and enhance the reusability and modifiability.

2.4.5 Layered Style

![Diagram of Layered Style]

Figure 2.2: Layered Style

The Layered Style hierarchically divides the software into units that groups modules that offer related services. The Layered Style is represented by a layer diagram and is one of the most commonly used styles in software architecture [15]. An example of the Layered Style is shown in Figure 2.2. Each layer provides a cohesive set of services that communicates through interfaces in a strict ordering relation. The relation used in the Layered Style is: allowed to use, which is a variant of the depends-on relation. Since true layered systems promotes modifiability and portability many architectures
CHAPTER 2. SOFTWARE ARCHITECTURE

strive to be shown as layered even if they are not [15]. According to Clements et al. [15] a valid layer system cannot allow a layer to use the services of a higher layer, since allowing upward usage destroys desirable properties that the Layered Style brings. Some other desirable properties that the Layered Style brings is increased abstraction, enhancement and reuse [19]. However the Layered Style also have some disadvantages, e.g. all systems are not easily structured in a layered manner and finding right abstraction levels can be hard [19]. There exists some variations of the Layered Style regarding which lower layers that can be used, some layering schemes allow a layer to use the services of any lower layer while some only allow the nearest lower layer. Various notations for the Layered Style exists in the literature, e.g. stack notation (shown in Figure 2.2), segmented layers, rings, layers with sidecar (shown in Figure 2.3) etc. Layers are often used in security architectures and this usage is further described in Section 3.4.4.

Figure 2.3: Layers With Sidecar

2.4.6 Pipe-and-Filter Style

The Pipe-and-Filter Style consists of a single type of components, the filter and a single type of connector, the pipe. A filter handles the transformation of a the input data it receives trough one ore more pipes and forwards the results trough one ore more output pipes [15]. An example of the Pipe-and-Filter Style is shown in Figure 2.4. Several filters are often connected and the filters executes incrementally [19]. Filters can also exist in parallel and then execute concurrently. Filters are typically used when the processing can be divided into a set of separate steps and therefore the filters don’t need to know the identity of the upstream or downstream filters [15]. Filters can be used to improve reuse since filters are independent, improve throughput using parallelization and simplify reasoning about behavior.

Figure 2.4: Pipe-and-Filter Style
2.4. ARCHITECTURAL STYLES

2.4.7 Repository Styles

In repository styles one of the component types are called repositories and consists of large collections of persistent data that other components read data from or write data to. Access to repositories are often handled by Database Management Systems (DBMS) that provides interfaces to handle the data operations. The DBMS often provides services like atomic transactions, security, concurrency control and integrity [15]. When repository styles are used the repository component often consists of both the DBMS and the data repository. Some examples of repository styles are: Shared-Data Style and Blackboard Style. In Blackboard Style the data repository may notify the other components about data changes and are mainly used in systems requiring complex interpretations of signal processing [19]. In the Shared-Data Style the components that access the data are responsible for initiating the communication with the repository. This style is further explained in the following section.

Shared-Data Style

The Shared-Data Style (shown in Figure 2.5) have one or more shared-data storage components and data access components that reads or writes the shared-data. The shared-data storage components provides data persistence, concurrent access by transaction management, access control, distribution and caching of data [15]. A benefit of using the Shared-Data Style is that it supports modifiability by decoupling the data producers from the data consumers [15]. Architectural quality attributes often associated with this style are e.g. performance, security, availability and scalability. Data could for example be replicated into multiple stores to enhance performance and availability by redundancy [15].

![Shared-Data Style Diagram](image-url)

Figure 2.5: Shared-Data Style
CHAPTER 2. SOFTWARE ARCHITECTURE

2.5 ADL and UML

An Architecture Description Language (ADL) is a formal language intended for representation of software system architectures. ADLs present a way to describe and analyze architectural designs [39] and often includes a formal syntax. Most of the ADLs were created to fulfill specific needs for specific domains and therefore only gained wider adoption within their specific areas. Unified Modeling Language (UML) which is a general modeling language commonly used within software engineering have for a long time not been accepted as an ADL but is now not only accepted as an ADL but have also become a de facto standard notation for software system architecture [39].

2.6 Software Architecture Models and Methods

Software architectures can be developed and described using various models and methods. Some of the most popular models and methods are: Attribute Driven Design, ISO/IEC/IEEE 42010, Kruchten’s 4+1 View Model and RM-ODP which are described in the following subsections.

2.6.1 Attribute Driven Design

Attribute Driven Design (ADD) is a method for designing architecture that focus on fulfilling both functional and quality attributes [5]. The ADD method is focused on the earliest stages of the design process and is based on recursive decomposition [4]. The decomposition process is based on a set of quality attribute scenarios which the software needs to satisfy. The ADD method uses three different kinds of views: module view, component and connector view and deployment view [4]. These views represents the first step of the design process and is therefore rather coarse-grained and more detailed design decisions are needed to make the architecture ready for implementation [5]. The ADD method is designed to handle uncertainty in requirements and depends on “architectural drivers” which are combinations of functional, quality and business requirements that needs to be defined in a reasonable level [4].

2.6.2 ISO/IEC/IEEE 42010

ISO/IEC/IEEE 42010 is a ISO standard named Systems and Software engineering - Architecture description which is based on two main ideas: a framework for architecture description and statements of the information that needs to be included in a ISO/IEC/IEEE 42010 compliant architecture description [29]. In ISO/IEC/IEEE 42010 the documentation of software architecture is based on views and an architecture description consists of
2.6 SOFTWARE ARCHITECTURE MODELS AND METHODS

one or more views. Requirements for an ISO/IEC/IEEE 42010 architecture description includes [29]:

- Identification and overview information.
- Identification of the stakeholders and their concerns.
- Definitions for all architecture viewpoints.
- Architecture views and architecture models for all architecture viewpoints.
- Architecture correspondences and correspondence rules.
- Architecture rationale decisions.

2.6.3 Kruchten’s 4+1 View Model

The 4+1 View Model of Architecture was introduced in 1995 by Philippe Kruchten [32] and is a model for describing software architecture using a set of concurrent views. Kruchten’s 4+1 View Model is an architecture definition model which describes the architecture from multiple viewpoints to simplify the complexity of the system [41]. The model is based on multiple views that separately represent the different concerns of the various stakeholders. The model also separately handles functional and non-functional requirements and proposes a notation for how to describe each view. The 4+1 View Model of Architecture consists of four views (logical view, process view, development view, physical view) plus a fifth view (scenarios) that tie the other four views together. The use of the fifth view (scenarios) has been one of the major success factors of the 4+1 View Model which has lead to its widespread adoption [41]. A visual description of Kruchten’s 4+1 View Model is shown in Figure 2.6. For each view, a set of elements

![Kruchten's 4+1 View Model Diagram](image)

Figure 2.6: Kruchten’s 4+1 View Model

which consists of components, containers and connectors are presented by Kruchten [32], however the 4+1 View Model is generic and other notations can be used.
CHAPTER 2. SOFTWARE ARCHITECTURE

The logical view conveys the functional requirements and represents the functionality provided to the end-user. The system is divided into abstractions described as objects or object classes. The approach used and described by Kruchten [32] is to describe the logical view as class diagrams or class templates.

The process view describes the architecture as a network of communicating processes and gives the concurrency and synchronization aspects of the system. In this view non-functional requirements like performance, scalability and availability are also addressed. The Process View is often described as a set or processes and their relations, and can be described at various levels of detail. The processes are executable units that can be controlled (e.g. started, stopped and reconfigured). To handle load and availability concerns these processes can be replicated [32].

The development view handles the module organization of the software. The software is divided into modules and submodules which can be developed by teams or individual developers. Often the development view is divided into subsystems that are organized in layers. The development view is often used as a basis for planning, cost evaluation, allocation of work teams, progress monitoring and various other management activities. The recommended approach for the Development View is to use a layered style where each layer has well-defined responsibilities [32].

The physical view describes the mapping between the software and the underlying hardware and also shows distribution properties. Mostly non-functional requirements like reliability, performance, availability and scalability are handled by the physical view [32].

The scenarios connects the other four views and shows how they work together and helps the stakeholders to get a better understanding of the other views. This view is redundant with the other four views (hence +1), however it has two main purposes: to help discover new architectural elements during the design and for validation and illustration after the design is complete [32]. In addition, the scenarios provides a way to communicate and reason about the architectural decisions. The scenario view is sometimes also called the use case view and according to [15] the use case view is described as architecturally important portions of the use case model.

The elements of the various views are connected to elements in other views and the information flow is from top to bottom and from left to right (as shown in Figure 2.6). This is the original notation from [32], several variations of this flow exists, e.g. [41] describes the information flow as bidirectional since they argue that as the system evolves, the information can flow in both directions. The relations between the elements and the mappings between the different views can be combined and use for various purposes, e.g. as measurable proofs that the system meets it’s requirements [41].
2.7. EVALUATING SOFTWARE ARCHITECTURES

2.6.4 RM-ODP

The Reference Model for Open Distributed Processing (RM-ODP) is a framework that creates architectures that supports integration of distribution, interworking and portability [28]. RM-ODP is based on formal description techniques to specify the architecture [28] and describes the architecture from five viewpoints [41]:

- The enterprise viewpoint (business model).
- The information viewpoint (information flow).
- The computational viewpoint (component-based software elements).
- The engineering viewpoint (distributed nature).
- The technology viewpoint (mapping of component details to technologies).

RM-ODP also provides additional support for architecture analysis by using eight distribution transparencies: Access, Failure, Location, Migration, Relocation, Replication, Persistence and Transaction [41]. These transparencies masks critical properties from the view and thereby we can assume that the property is correctly implemented [41]. Using these transparencies makes it easier to validate the remaining visible parts of the architecture.

2.7 Evaluating Software Architectures

Software architecture evaluations determine properties of software architectures by analyzing them and verifies architectural decisions compared to problem statements and requirements. The evaluations also determine the satisfactions of quality attributes and helps to find tradeoffs between them since the quality attributes often affect each other in a positive or negative manner. Performing software architecture evaluations has many benefits and the major advantages are that they tend to increase quality, control cost and decrease risks [5].

Software evaluations are typically performed either before implementation (early evaluation) or after implementation (late evaluation). Early evaluation are often based on the specifications and the descriptions of the software architecture while late evaluation often is based on metrics, e.g. coupling and cohesion of components. Abowd et al. [1] propose to categorize the evaluations into two types: qualitative evaluation and quantitative evaluation. Qualitative evaluations use qualitative questions to measure quality. Examples of questioning techniques are: scenarios, questionnaires and checklists [1]. Quantitative evaluations focus on quantitative measurements that address specific quality attributes.
CHAPTER 2. SOFTWARE ARCHITECTURE

The qualitative technique that have been most widely used are scenario-based methods, which evaluate the software architectures abilities with respect to a set of scenarios and offers a systematic way to investigate the software architecture and are flexible and simple [3]. Scenarios describes a single interaction that a stakeholder has with the system and the scenario-based methods determines if an architecture can perform the scenario or not. Some of the most common scenario-based evaluation methods are ALMA, ATAM, SAAM and SBAR which are described in the following subsections.

2.7.1 ALMA

Architecture-Level Modifiability Analysis (ALMA) is is a scenario-based evaluation method that is developed to address the modifiability quality attribute [7]. The ALMA method was initially developed and tested for business information systems only [27]. ALMA is divided into five main steps: software architecture description, scenario elicitation, scenario evaluation and interpretation of results [7]. The outcomes of an ALMA evaluation consists of: impact estimates for each scenario, a modifiability model and a scenario generation stopping criterion [27]. ALMA focuses on modifiability, and this focus can be divided into three main goals: maintenance prediction, risk assessment and software architecture comparison [7].

2.7.2 ATAM

Architecture Tradeoff Analysis Method (ATAM) is a comprehensive scenario-based method to evaluate software architectures according to satisfaction of quality attributes [5]. ATAM especially provides a way to evaluate software architectures with respect to competing quality attributes [42] and is based on SAAM (see section 2.7.3) [27]. The ATAM evaluation is carried out in 9 steps and is considered to be a more complex method compared to SAAM [42]. The ATAM evaluation focuses on the most important quality attribute requirements, which means that the evaluation will not tell if all the requirements of an architecture is will be met [5]. ATAM is a robust evaluation method that focuses on the high-priority scenarios and on finding areas of risk in an architecture [5].

2.7.3 SAAM

Software Architecture Analysis Method (SAAM) is a method used for analyzing the properties of software architecture. Originally the method was developed to compare competing architectures and was introduced by Kazman et al. [30] in 1994. It has with increased experiences evolved and therefore there exist variations in the steps involved in SAAM [31].

The inputs to the SAAM method mainly consists of business drivers, architecture descriptions and quality requirements. The outputs consists of
2.7. EVALUATING SOFTWARE ARCHITECTURES

quality sensitive scenarios and estimated efforts required to fulfill the scenarios. Some recognized business drivers for software architectures are the quality attributes [30]. According to Kazman et al. [31] the scenarios differ widely in both size and complexity. The scenarios are brief description of the desired use of a system, and they are typically a sentence long. Kazman et al. [31] also emphasizes that not all scenarios are impacted by the architecture, some depend on code level or hardware level aspects and some cannot be evaluated at architectural level. The scenarios are divided into direct scenarios and indirect scenarios. Direct scenarios are those that are directly supported in the architecture and indirect scenarios are those that requires changes to the architecture. The purpose of the SAAM is not to criticize any architecture, but rather about providing a way to decide which architecture that is best suited for an organization’s needs [30]. A visualization of the SAAM process model is shown in Figure 2.7 and consists of the following steps [31]:

1. **Describe candidate architecture(s).** This candidate architecture(s) shall be described so that it is well-understood by the parties involved in the evaluation. Both the computation and data components shall be described.

2. **Develop scenarios.** The scenarios that are elicited shall describe the activities the system must support and changes that are likely to be made to the system. It is important to capture all uses of the system and tasks that are relevant to various roles, e.g. end-user and system administrator.

3. **Perform scenario evaluations.** For each indirect scenario the various changes to the architecture that are needed shall be listed together with a estimate of the cost of performing the change. Changes are either new components or connections that need to be introduced or
CHAPTER 2. SOFTWARE ARCHITECTURE

that changes need to be made to the specifications of existing components or connectors. This stage should result in a summary table that lists all scenarios (both indirect and direct). For the indirect scenarios the impact or changes shall be described. The tabular summary is especially useful when comparing alternative architectures to find out which architecture that best supports a set of scenarios.

4. **Reveal scenario interaction.** Scenario interaction is when two indirect scenarios require changes to the same components or connections. Revealing scenario interactions therefore consists of identification of scenarios that affect common components and is a way to measure how well an architecture supports separation of concerns. For each component all indirect scenarios that affect it should be determined.

5. **Overall evaluation.** The final step is overall evaluation which is about weighting each scenario according to their relative importance to produce an overall ranking. The weighting should be performed by all stakeholders involved in the evaluation and is a way to reflect about the importance of the quality attributes that the scenario represents.

2.7.4 SBAR

Scenario-Based Architecture Reengineering (SBAR) is a method for reengineering software architectures [6] and the method is used to evaluate operational quality attributes of an architecture [42]. The reengineering of an software architecture is often started by major changes in the requirements and these changes are often concerned with software quality attributes [6]. The input to an SBAR evaluation consists of an updated requirements document and the existing architecture and the output is an improved architecture [6].
Chapter 3

Security Architecture

This chapter explains what security architecture is and describes security goals and security considerations. Various secure design patterns are presented and the area of software security architecture is described.

3.1 Introduction to Security Architecture

Security architecture is a design that handles requirements (e.g. authentication and authorization) and risks (e.g. risks of a particular environment) and determines which security controls should be used [47]. The term security architecture is broad and generally consists of the following elements: network, host, applications, information, software, hardware, databases and physical [2]. There are no general security architecture solutions which will work across organizations and with changing infrastructure. This implies that security architectures need to be able to adapt changes both in technologies and strategies [2]. In software architecture, the term “security” can often do more harm than good since there exist various definitions which are often conflicting [40]. This implies that the architecture must provide specific security guidance to the other stakeholders [40]. Software security architecture is something that needs to be handled as an iterative process by decomposing complex problem areas to get an understanding of the details and then building up the design according to the security goals [40].

Various security components focuses on hardening systems against threats while often excluding other goals. Secure design patterns (described in Section 3.4) focuses on both strong security and good architecture and therefore brings balance to the architecture [41].

Security Assertion Markup Language (SAML) is an open standard for exchanging security information (authentication and authorization data) and can be used as a base for software security architectures. SAML provides opportunities for authentication, encryption, management of digital rights and federated architectures and is further described in Chapter 6.
CHAPTER 3. SECURITY ARCHITECTURE

3.2 Security Goals in Architectures

Software architects are often given the goals of designing future-proof systems, which implies that the system should have the flexibility to adapt to changes of various natures [41], e.g. technology changes, volume growth, introduction of new interfaces and changes in the threat landscape. These goals increase the complexity of the system and to handle this complexity the systems are often divided into multiple levels of abstraction. The architect often creates interfaces between subsystems to minimize the impact of changes to subsystems onto the whole system. This way of designing the architecture enables the architect to hide design decisions about the implementation of subsystems from the other parts of the system. However adding security to the architecture often has the negative impact of collapsing the hierarchy of abstraction by promoting low-level decisions to a higher level [41]. Therefore security need to be implemented according to structured security architecture guidelines to avoid creating tension in the design.

3.3 Security Considerations

Designing for failure, i.e. making a design that can handle and recover from failures, is a fundamental principle in security design. In most systems the focus is on the basic flow of the system since it is these flows which provides business value, however from a security perspective it is important to focus on the exceptional and the alternative flows since these often become the attack vectors [40].

The use of distributed systems is increasing and these systems are often used in ways not intended by the original architectural design, e.g. old legacy systems get connected to the web, data and messages are used in ways not intended by the protection mechanisms [40]. A security countermeasure for this type of security problem is to move the security mechanisms closer to the assets, e.g. using technologies like encryption for the data elements to constrain access to persistent data [40].

3.4 Secure Design Patterns

Secure Design Patterns have the purpose to avoid introducing vulnerabilities and to mitigate consequences of vulnerabilities [17]. Secure design patterns exists at various levels ranging from high-level architectural patterns to implementation patterns, however this section will focus on secure design patterns suitable for architectural design of software systems. Architectural security patterns have to goal of enabling valid communication in a way which is as transparent as possible, while at the same time blocking all invalid communication [41]. This section will introduce some of the secure
3.4. SECURE DESIGN PATTERNS

design patterns for architectures.

3.4.1 Directory
Directories are databases which are read frequently and written infrequently. The use of directories have increased tremendously in the recent years which have led to a widespread acceptance and many directory enabled products [41]. The original X.500 directory definition is rather complex and this has led to the development of a simplified protocol called Lightweight Directory Access Protocol (LDAP) which is the de facto standard on the Internet today [41]. Directories are important security components for security architectures since they allow enterprises to partition users (e.g. customers, employees and partners). The directories are also important in security architectures since they often handle security information. This security information could be user profiles, user attributes (e.g. roles and permitted services) and user group information [41]. Various commercial directory products exist and they often support secure administration.

3.4.2 Distributor
The distributor pattern is a more abstract architectural structure that takes a communication stream and separates it into multiple streams based on some criteria [41]. Distributors can be either symmetric (identical outgoing streams) or asymmetric (separating outgoing streams based on e.g. protocol and QoS rules) [41]. Distributors are often used in security architectures at specific network choke points (e.g. for load balancing) [41]. In these situations it is important to use the same security strategy at all outgoing locations to avoid attacks of the type that uses wrapping of one type of communication into another type to bypass security checks.

3.4.3 Interceptor
Interceptors are a type of man-in-the-middle security components [41] that can add, delete or modify the communication between communicating parties. Interceptors are often used in security architectures and can be combined to supply complex decision support, and can by that implement complex security actions [41].

3.4.4 Layers
The general layered style for software architectures is described in Section 2.4.5. The layer pattern is one of the most widely used patterns in security architectures and is used for various purposes. Using layers in security architectures provides a way to separate the different layers using clearly defined interfaces which helps to hide implementation details in one level from the other levels [41]. Layers are used to separate functionality, e.g.
CHAPTER 3. SECURITY ARCHITECTURE

secure transport, secure infrastructure and security services [41]. Layers in security architectures are often even more strict than layers in the general sense. This strict type of layering prohibits interaction of a layer with anything else than the direct underlying layer [41]. Regarding security, the most important attribute of a layer is modifiability since this provides possibilities for adding additional security to an existing software without affecting the code of other parts of the software [41].

3.4.5 Trusted Third Party

A trusted third party is an entity that is trusted by all other entities involved in secure communications [41]. In the initialization phase all participants agree to trust the third party, and this trust includes information received, decisions and validations. There exist some variants of a trusted third party, e.g. inline (acts as a proxy), online (real time interaction) and offline (processed when available) [41]. Various examples of trusted third parties exists e.g. Public Key Infrastructures (PKIs). PKI is one of the most used methods for handling public encryption keys. In PKIs the Certificate Authority (CA) is a trusted entity which is used to sign documents, the Registration Authority (RA) is the trusted entity which handles the proof of identity for certificate requests. PKIs also includes Certificates Revocation List (CRL) servers which handles a list of revoked certificates and the Online Certificate Status Protocol (OCSP) handles real time verification of certificates. Another variant of trusted third party is the PGP cryptosystem where users sign each other’s certificates.

3.5 Software Architecture Security

Security have for a long time been presented as an independent property of a software system rather than a real system feature, it is only recently that security has become a principle that is integrated into the design of the system [41]. Garlan and Shaw [19] provides some actions that an architect should perform when designing secure software, e.g. decomposing the system into subsystems (where each subsystem has a specific architectural style), choose security components that matches the styles (by implementing the required security properties), add security constraints and choose communication security components that enforce required security properties. The choice of architectural style therefore drives the choices of security components and their relations to other parts of the system.

Additional principles for secure software architecture includes e.g. proper identity management (described in Chapter 4), authentication and authorization (described in Section 4.4 and in Chapter 5).
Chapter 4

Identity Management

This chapter presents the concept of digital identity management and explains related terms, e.g. credentials, authentication, authorization, single sign-on and federated identity management.

4.1 Digital Identity Management

A steadily increasing security concern in information systems is identity management. Security breaches as a result of flaws in identity management are increasing and the security attacks often originate from external sources. These attacks can cause business disruption and cause large financial damage. Identity management is needed to open up services, applications and resources in information systems and it is crucial to secure the identity through authentication of users before granting access to resources within companies and organizations. [44]

When information assets shall be protected there are essentially three main requirements that need to be considered: confidentiality, integrity and availability. Confidentiality is about protecting the information from unauthorized disclosure, integrity is about protecting the information from modifications that could affect the validity of the data and availability is about making sure that the information is available when needed by the users. [49]

A digital identity can be defined as “the digital representation of the information known about a specific individual or organization” [8]. Digital identities can be used for various purposes and could include attribute information like a name or Social Security Number (SSN). The digital identities can also include identifiers like user names or pseudonyms to be used when interacting with the user in the digital world.

In the context of digital identity management another definition of identity presented in [8] is “information about an entity that is sufficient to identify that entity in a particular context”. Related to this definition they conclude that an identity consists of three parts: identifiers, credentials and
CHAPTER 4. IDENTITY MANAGEMENT

attributes. Here identifiers are some form of data to identify the user, e.g. an account name or an Uniform Resource Identifier (URI). Credentials are data that provides evidence for the claims made about the identity, e.g. usernames & passwords, digital certificates and SAML assertions (SAML is described in chapter 6). Characteristics that describes an identity are called attributes, e.g. full name, date of birth, roles and activity records. [8]

Digital identity management poses many problems for companies and organizations. Often the companies and organizations have customized identity management solutions for their legacy systems, which have different security mechanisms and different infrastructure. Therefore users are often required to have different user accounts with different credentials for each system. As the number of applications and services increases there is a need for a single identity that can be used in multiple systems and infrastructures. Single-identity solutions however require extensive security integration. Things that need to be taken into account are, e.g. management of encryption keys, certificates and how the legacy systems shall be able to share authentication mechanisms. [44]

There are many security vulnerabilities and threats related to identity management, which not only affect a single system since identities are managed across system boundaries. Some examples of security vulnerabilities and threats include denial-of-service attacks on parts handling identity management, man-in-the-middle attacks which steal identity information, session hijacking, spoofing using faked or stolen accounts, replay attacks using stolen identities etc. Therefore it is of great importance for security architects to understand the problems regarding user identities. [44]

4.2 Stakeholders

In identity management solutions there are different parties that have interest in the digital identities. Solutions for identity management therefore need to take these parties’ views into consideration. According to [8] these parties can be grouped into four different stakeholder types: subjects, identity providers, relying parties and control parties.

The subjects are usually the individuals whose identity attributes are collected and used for various purposes. For the subjects, privacy and protection from misuse of the identity and the attributes are the most important aspect since this could cause harm the individuals [8]. Another big risk for the subjects is failure to complete an operation due to insufficient trust from the relying party [35].

The parties that supply the subject with their identities are the identity providers. An identity provider has four basic tasks: to generate and assign identity attributes to subjects, to bind identity attributes of a subject to other identity attributes of the subject, to create assertions about identity attributes and to supply credentials for identity attribute and identity assertions. [8]
4.3 CREDENTIALS

The parties that require proper credentials in order to provide services to the users are called relying parties. An important aspect for a relying party is to determine to which extent they can trust the provided credentials. Another important aspect is that they should be able to verify the credentials with the issuer. [8]

The control parties are the stakeholders who need to access identity information, e.g., logs of transactions that requires identity information. There exist various reasons for the need to access this identity information, one example could be forensic investigations. The control parties have one main requirement and that is the support for audibility. [8]

Stakeholders can often play more than one role, e.g., relying parties can handle the identity management duties themselves, but this leads to subjects having different usernames and passwords for each service. Using separate identity providers can provide economic benefits since the relying parties can focus more on their main services and relying parties can share identity provider. Sharing identity providers can also make it possible to provide more advanced identity management services, e.g., stronger authentication and single sign-on. [8]

4.3 Credentials

Credentials are a collection of identity assertions and attributes about a specific subject. The credentials are often issued by the identity provider. When the relying party shall decide if the credentials shall be trusted or not the issuer plays a major role since the integrity and sometimes also the validity of the contents of the credentials is attested by the identity provider [8]. There exist several different classifications for credentials. One example provided by [11] is to classify the credential into the following types: validated credentials (validated then digitally signed), authenticated credentials (digitally signed but not validated), raw credentials (digitally signed by the subject itself and not validated).

4.4 Authentication and Authorization

The credentials are used to authenticate and authorize users. The process of authentication is often considered to consist of two parts, identification and authentication. Identification is the part where the user identity is provided to the system and authentication is the part where the users identity is validated. Even though the user claims a specific identity by providing credentials, it doesn’t necessarily mean that this claim is true. [49]

Authorization is the process of determining if an already identified and authenticated user is allowed to access information in a specific way [49]. Before the authorization can take place the identification and authentication
CHAPTER 4. IDENTITY MANAGEMENT

must have been successfully performed. The combination of authentication and authorization is also called access control [20].

4.5 Single Sign-On

Single sign-on (SSO) is an authentication process which enables access control to multiple systems based on the initial authentication of the user. Thus, users only need to provide their credentials once. When used within the same enterprise it is called enterprise SSO (ESSO), when used across multiple enterprises it is called multidomain SSO and when used by clients interacting with web applications across the web it is called web-based SSO.

Single sign-on allows a user to enter his/her credentials only once during a session, which allows the user to access multiple services without having to be prompted for authentication multiple times[8]. The use of single sign-on also eliminates the need for users to manage multiple credentials (e.g. multiple passwords) and increases productivity among system administrators since the number of accounts they need to manage is significantly reduced [48].

Currently most SSO deployments are ESSO, but achieving even this level of SSO is cumbersome since many legacy systems implement their own customized authentication solutions. For these solutions the SSO system needs to translate the initial authentication credentials into credentials that can be used by the legacy application. SSO can be implemented using various approaches and current standardization efforts are promoting more advanced approaches using federated identities. [8]

4.6 Federated Identity Management

Today a typical enterprise has many online services which they provide to employees, partners, contractors and customers. Since the collaborations have become more distributed, there is a need for handling digital identities in a more distributed way across various domains [43]. Federated identity management provides dynamic distribution of identity information and delegation of identity tasks across security domains.

Handling of digital identities raises issues and challenges in various environments (e.g. in an enterprise context or on the Internet), especially when the digital identities represents humans. Federated identity provides solutions to many problems that are shared by the environments and SSO is often the first federated identity functionality which is added [37]. When humans are involved in the federated interactions, then the interactions often involves four different components: the user (has a specific digital identity), the user agent (could be a web browser or application which passively or actively mediates the interaction), the relying party (also called service
4.6. **FEDERATED IDENTITY MANAGEMENT**

provider which offloads authentication to a third party) and the identity provider (handles authentication) [37].

Federated identity management can offer a lot of benefits but also imposes security and privacy risks as well as costs. A new risk which is introduced by federated identity is about crossing security domains. In the best of worlds, all stakeholders should make sure that their communication channels are secured against man-in-the-middle attacks, replay attacks and session hijacking and other types of attacks. This is typically performed using Secure Sockets Layer / Transport Layer Security (SSL/TLS) (further described in Section 5.7) using mutual authentication, however this part is often overlooked by developers. Also some security functionality like auditing could conflict with privacy issues like keeping the users true identity private. [37]

Federated identities enhances the portability of the digital identities but also creates new architectural challenges which have both security and privacy issues [37]. Federated identity management enables users to interact with various service providers or web sites which have established trust relationships, by signing on only once [43]. As an example a company employee could log in to the enterprise network, perform the time reporting, and then use the car rental services from their business partner to order a car, without having to login again. This type of scenario is called cross-domain single sign-on (SSO) and is the most widely used application area for federated identity management.

There exists various protocols for handling federated identities and some common are: Security Markup Language (SAML), OpenID and InfoCard. SAML is further described in Chapter 6 and is an XML-based language for exchanging security information which conveys authentication information in the form of assertions about subjects [43].
Chapter 5

Electronic Authentication

This chapter explains what electronic authentication is and describes various related concepts, e.g. tokens, electronic credentials, assertions, authentication methods and authentication protocols. Various threats related to electronic authentication are also presented.

5.1 Overview

The process of establishing confidence in the identity of a user in an information system is called electronic authentication. The information system can use the authenticated identity to determine what actions the user is authorized to perform. An overview of the basic components in an electronic authentication system is shown in Figure 5.1. The electronic authentication process begins with a registration where the user registers or is supplied with a secret (called a token, see section 5.2) and a credential (see section 5.3). An example of a registration is when a user signs up for an e-mail account and is given a password. The credential (e.g. e-mail address) binds the token to a name and sometimes also to other attributes (e.g. social security number, street address etc.). When a user can successfully demonstrate ownership and control of a token to a verifier (e.g. successful log on attempt at webmail) the verifier can verify the identity of the user. The verifier sends an assertion (see section 5.4) to the relying party (unless the verifier is the relying party). The assertion includes a name, some identifier or other attributes verified at the registration. The relying party can then use this authentication information to perform access control. The authentication process therefore only provides an identity and verified attributes, not what actions the user is authorized to perform or what access privileges the user has. [12]
5.2 Tokens

Tokens are something that the user holds and controls and can be used to authenticate the identity of the user. In electronic authentication the authentication is often performed using a system over a network and therefore the secret token must be protected. The token can be e.g. a cryptographic key that is encrypted with a password. [12]

To confirm the user identity the information generally comes from one or more of the following factors [9] [12] [48]:

- What you know (e.g. a password)
- What you have (e.g. smart-card)
- What you are (e.g. fingerprint)
- Where you are (e.g. in a specific building)

The number of factors that is used for authentication is often used for categorizing authentication systems [12]. The more factors that are used, the greater the level of protection [48]. Many information systems, such as e-mail services, uses single-factor authentication which means that only one factor (only one token, e.g. a password) is used for authentication. Two-factor authentication uses two of these factors which means that a user must provide two types of tokens (e.g. credit card and PIN-code in an ATM) in order to be authenticated. Multiple-factor authentication is sometimes also
CHAPTER 5. ELECTRONIC AUTHENTICATION

called strong authentication [49]. In the authentication process, multiple factors can be presented to the verifier or multiple factors can be used to protect the secret that is presented to the verifier. The secrets are often public key pairs or shared secrets.

Public key pairs are asymmetric keys and consists of a public key and a private key. A verifier knowing the user’s public key can verify the user’s identity by proving that the user has control over the private key (which is used as a token) [12].

Shared secrets are typically symmetric keys or passwords. Passwords are considered to be categorized as something you know and symmetric keys as something you have, but can be used in similar authentication protocols. Passwords often have lower resistance to network attacks and are vulnerable to key loggers and “shoulder surfing”, which means that passwords and symmetric keys have differences in their authentication properties. [12]

5.3 Electronic Credentials

Electronic credentials have the purpose of binding a name and the attributes of a user to a token. There exists various types of credentials today and new ones are continuously being developed. An electronic credential shall include information that makes it possible to recover the records that was stored in the registration of the credential and the name that is linked to the user. Credentials can be either general or targeted at a specific verifier. Some common types of electronic credentials are: X.509 public key certificates (binds identity to public key) and X.509 attribute certificates (binds attributes to an identity or a public key). [12]

The electronic credentials are often stored in directories (e.g. LDAP) or databases. The credentials can be stored as digitally signed objects, which means that the directory or database can be an untrusted entity. Another solution is that the directory or the database is a trusted entity. This trusted entity shall authenticate itself to the verifier or the relying party and therefore the credentials could be stored as unsigned data. [12]

5.4 Assertions

Assertions are used to send information about a user or the authentication process from the verifier to the relying party. During an electronic authentication session the verifier must verify that the user holds and controls the token. The user authenticates it’s identity using the token and an authentication protocol. This process is called Proof of Possession (PoP) and is when possible designed so that the verifier learns nothing about the token from the communication. It is undesirable for the verifier to learn shared secrets unless the verifier was the entity that registered the token. When the verifier and the relying party are separate entities the verifier must send
5.5. AUTHENTICATION METHODS

The results of the authentication to the relying party. The object that is created to carry this information is called an assertion. [12]

The source of the assertion and the assertion properties are used by the relying party to decide if the assertion shall be trusted. As a minimum the assertions include a name and some information to make it possible to gather the registration records. Two examples of assertions are SAML assertions and cookies. SAML assertions are assertions described using the Security Assertion Markup Language (see chapter 6) and can be digitally signed to ensure the integrity.

Cookies can both be used as assertions and pointers to assertions. Cookies are strings placed in the users web browser and can only be used by websites in the same domain as the one that placed the cookie. [12]

5.5 Authentication Methods

Authentication can be carried out in various ways and one of the most important decisions when designing secure systems is to choose appropriate authentication method(s). When using electronic tokens for authentication one important consideration is the number of factors that are used (number of factors is described in section 5.2). All authentication methods contains one or more authentication factors. Using more factors gives higher protection but often at the cost of a more cumbersome authentication process for the users.

5.5.1 Password Authentication

Authentication based on passwords requires the user and the verifier to share a password, so if the user provides identification information and the correct password then the verifier can successfully authenticate the user. Static passwords are one of the oldest and most widespread authentication methods [49]. The trust model for password authentication is based on the assumption that only the user and the verifier knows the password. Often the verifier stores the password in an authentication database where it should be protected in a secure way [49].

5.5.2 One-time Password Authentication

Passwords have many security issues, e.g. should be long, complex and hard to remember. By using authentication based on one-time passwords some of these issues can be avoided. One-time passwords are passwords that are generated and used only once. One-time password authentication can be both software based and hardware based [41]. This could be hardware devices that generate a one-time password to use for authentication [12] or services that send SMS messages containing one-time passwords valid only
CHAPTER 5. ELECTRONIC AUTHENTICATION

for a specific time period. One-time passwords can be used for together with e.g. a password to provide two-factor authentication.

5.5.3 Hardware Token Authentication

Hardware token authentication uses hardware devices which contains cryptographic keys. The cryptographic keys in the devices are protected and authentication is performed by the user providing a proof that he has the hardware device and has control of the key [12]. This means that hardware token authentication often uses two factors (something you have and something you know) [41]. Some of the most common hardware tokens used today are smart-cards and USB tokens.

5.5.4 Software Token Authentication

Authentication using software tokens is often based on cryptographic keys. The authentication is performed by the user proving both possession and control of the cryptographic key. The cryptographic keys are often stored on disk or some other digital media and should be encrypted using a key derived from the activation process [12]. This key (typically a password) is only known to the user and is used to activate the software token.

5.6 Authentication Assurance Levels

The authentication process shall according to NIST 800-63 Electronic Authentication Guideline [12] provide the relying party with sufficient information about the assurance level. For this they have defined requirements for four levels of assurance (abbreviated LoA). These requirements are supplements to the definitions of assurance levels provided by OMB M-04-04 [10]. Level 4 provides the highest level of assurance and Level 1 provides the least assurance. The definitions and requirements for each level can be found in [10] and [12] and is not further detailed in this thesis report. Various other definitions and requirements for LoA exists, however those provided by 800-63 and M-04-04 are often seen as the foundation and most other definitions and requirements are based on them.

5.7 Authentication Protocols

Authentication protocols are defined by NIST 800-63 Electronic Authentication Guideline [12] as “the sequence of messages between a claimant and a verifier that enables the verifier to verify that the claimant has control of a valid token to establish his/her identity”. Some of the most widely used authentication protocols are: Public Key Infrastructure (PKI) and Secure Sockets Layer (SSL).
5.8. AUTHENTICATION THREATS

Public Key Infrastructure is one of the most well known security components and have many standards around it, e.g. PKCS, X.500 and OCSP. PKI uses asymmetric cryptology to securely exchange data. The public key is bound to the identifying credentials using the certificate [41]. The certificate is signed by the trusted third party called a Certificate Authority (CA). The CA ensures the validity and integrity of the certificate [41].

SSL [18] was originally developed by Netscape and is now available as an open-community standard called Transport Layer Security (TLS) [16] and provides secure communication between clients and servers. SSL has gained large success in providing secure web communications and enables developers to add strong security to applications that uses sockets [41]. The initial use for SSL was for securing communication between web servers and web clients but nowadays many applications protocols have been SSL-enabled [41]. SSL/TLS uses PKI to exchange keys for the symmetric encryption that is used in the communication.

5.8 Authentication Threats

Generally authentication threats can be divided into threats against the authentication protocol and threats against tokens or threats that reveal confidential information. Threats against tokens are generally the most serious attacks since then the attacker can use the token to act as the token owner.

5.8.1 Token Threats

If an attacker gets hold of the token then the attacker can act as the token owner. Threats to tokens are generally divided into three categories [12].

- **Theft** - The first category is that something you have can be stolen or cloned.

- **Disclosure** - The second category is that something that you know can be revealed to an attacker.

- **Replication** - The third category is that something you are can be replicated.

To prevent these types of token threats there exists several mitigation techniques. E.g. the use of multiple-factors can raise the difficulty of the attacks. Another technique is to use system and network controls to prevent an attacker from gaining access to systems [12]. To prevent duplication of stolen tokens, various types of physical protection can be used, e.g. mechanisms providing tamper resistance and detection [12]. Another mitigation technique is to use complex passwords which could reduce the chance for successful attacks where password tokens are used.
5.8.2 Authentication Protocol Threats
There exists various threats against authentication protocols, e.g. eavesdroppers that observes and analyzes protocol runs. Another threat is impostors which could act as a claimant or a relying party to obtain sensitive information. In addition hijackers could take over already active sessions to obtain sensitive information or generate invalid information. Another threat to authentication protocols are man-in-the-middle attacks where the attackers place themselves in the path of the protocol communication to obtain sensitive information. To prevent authentication protocol threats, the protocols should be designed so that exposure of long-term authentication secrets are limited. [12]

5.8.3 Additional Authentication Threats
Beyond token threats and authentication protocol threats there are several other threats in the context of electronic authentication. These other types of attacks include e.g. malicious code attacks that could compromise tokens [12]. Other threats also include insider threats and various type of out-of-band attacks like social engineering or “shoulder surfing” [12]. Another type of attack is about downgrading authentication strength to fool a claimant to use a less secure protocol [49].
Chapter 6

SAML

This chapter describes what Security Assertion Markup Language (SAML) is, how it is structured and explains various SAML concepts, e.g. assertions, protocols, bindings and profiles.

6.1 Introduction to SAML

Information about electronic authentication can be communicated in various ways, e.g. using Security Assertion Markup Language (SAML) [36]. SAML is an XML-based framework for exchanging security information between subjects. The security information is exchanged using signed SAML assertions that can be trusted by applications across different security domains [25]. The development of the SAML standard was started in 2001 [34] by the standards organization OASIS (the Organization for the Advancement of Structured Information Standards) and in this thesis SAML V2.0 was used.

Previously many products have claimed to support web-based SSO (see Section 4.5). They have often relied on cookies in the user’s web browser to maintain the authentication information. Due to the fact that browser cookies are never transmitted across different domains the authentication information will not be available to other domains. To be able to support multi-domain SSO (MDSSO) these products would typically use proprietary solutions to transfer the authentication information between different domains. The use of proprietary solutions often limits the possibilities for cooperation since business partners often have heterogeneous environments. SAML solves the MDSSO problem by providing a representation for transferring user information without restrictions in domains. [25]

In other words, SAML provides SSO for heterogeneous applications and is a more cost-effective solution that avoids the interoperability problems of the proprietary solutions. [44]
CHAPTER 6. SAML

To be able to establish a cooperative application environment the applications must have a common language to exchange authentication information and they must also have a common understanding of who the user in the exchange is. These users often have separate local identities for each cooperating party and each security domain which they interact with. SAML provides a way to create a shared user identity which makes it possible to share information about the user across domains. When this agreement is made the user is said to have a federated identity. This reduces the need for services to independently collect and maintain identity data and reduces identity management costs. [25]

SAML also provides platform neutrality by separating the security framework away from specific vendor implementation and platform architectures. In addition SAML pushes the responsibilities for identity management from the service providers to the identity providers, which is often more consistent with the business model of the identity providers. [36]

6.2 SAML Participants

In a SAML exchange there are at least two participants, a SAML asserting party and a SAML relying party. Often there is a user involved who is running a web browser or a SAML-enabled application. The SAML assertions are created by the asserting party (sometimes also called a SAML authority) and then received and used by the relying party. When a SAML request is made, the party that makes the request is called a SAML requester and the party that receives the request is called a SAML responder. The willingness of a relying party to rely on an assertion from the asserting party depends on if there is a trust relationship established between them. [25]

The SAML entities can act as various SAML roles which specify which services and protocol messages they will use and what kind of assertions they will create and consume. An example is that to support multi-domain SSO (MDSSO), SAML has defined two roles called identity provider (IdP) and service provider (SP). For the case of attribute queries, SAML has defined roles called attribute authority and attribute requester. [25]

The terms subject and principal are often used interchangeably and refers to a entity that can be authenticated within a specific security domain and which something can be asserted about. A typical SAML assertion could contain information like the name of the subject, the e-mail address of the subject, a statement about the authentication result and which authentication mechanism that was used. [25]

An example of some SAML participants involved in a SAML exchange is shown in Figure 6.1.
6.3 SAML Architecture

SAML is built upon different components that are put together to enable various use cases (see Figure 6.2). These components allow communication about identity, authentication, attributes and authorization information between parties that have a trust relationship.

- **SAML assertions** are usually created by an asserting party when a request is received and the structure of the SAML assertions are defined by the SAML assertion XML schema.

- To make the SAML requests and responses the entities use **SAML protocols** which are defined by the SAML protocol XML schema.

- **SAML bindings** describe how lower level communication protocols (e.g. HTTP or SOAP) are used to transfer SAML protocol messages.

- **SAML profiles** are used to satisfy particular use cases (e.g. web-browser SSO) and typically sets constraints on SAML bindings, protocols and assertions.

SAML also includes attribute profiles which is a type of profiles that only put constraints on how attribute information is exchanged [25]. Another concept that is present in SAML is **metadata** which describes how SAML
parties can exchange configuration information. The configuration information could be things like the different supported roles, supported bindings, information about how encryption and signing shall be performed etc. The metadata is also defined by its own XML schema. In addition, SAML also provides the concept of *authentication context* which is a way to provide information about the type and strength of a particular authentication of a user. This information is included in or referred from the SAML assertion. A service provider can also use authentication contexts in requests to identity providers to make sure that a user is authenticated with specific set of requirements. Authentication contexts are separated into different classes which each is defined by an own XML schema. [25]

### 6.4 SAML Assertions

SAML assertions are produced by a SAML authority and can be described as a package of information that can contain statements made by a SAML authority. The SAML standard specifies three types of assertion statements: *authentication*, *attribute* and *authorization decision*. An assertion consists of an outer structure and an inner structure. The outer structure is generic and provides information that is common to all assertion statements. It can contain both required and optional information. The outer section can include information like issuer, signature, subject and conditions. The inner elements contain the authentication, attribute, authorization decision or user defined statements. [14]

- In order to assert that a subject was authenticated at a specific time and with a specific authentication mechanism, a SAML authority creates a *authentication assertion*. The authentication assertion statement contains the elements: AuthenticationMethod (e.g. password),

![Diagram of SAML Concepts](image)
6.5. SAML PROTOCOLS

AuthenticationInstant (e.g. timestamp) and possibly SubjectLocality (e.g. domain name and IP address). [44]

- An attribute assertion is created by a SAML authority to assert that an association has been made between the subject and a specific set of attributes. The elements that an attribute assertion statement contains is: Attribute, AttributeName, AttributeNamespace and AttributeValue. [44]

- Authorization decision assertions describe a response to a request for a specific subject to access a specific resource. The response could be permit, deny or indeterminate. [44]

6.5 SAML Protocols

SAML protocols are sets of request-response pairs defined by an XML schema. SAML defines a number of protocols which makes it possible for service providers to request assertions, ask for authentication of subjects, request logout of related sessions (Single Logout) and handling of identifiers and their mappings. [14]

- The authentication request protocol is used when subjects want to request assertions containing authentication statements. The subject sends an authentication request message to the SAML authority and requests that the authority returns a response message containing the assertion. These assertions could optionally include other statements, e.g. attribute statements. [14]

- The single logout protocol provides possibilities for almost simultaneous logout for all sessions that are linked to a specific subject. The single logout protocol can be used for logout directly from a session, from the authority or due to timeouts or other reasons. [14]

- The assertion query and request protocol provides a way to request assertions by assertion ID or to query for assertions based on a subject and a statement type. Querying for assertions can be used for both new and existing assertions. [25]

- Artifacts are small pieces of data that are used to send SAML protocol messages by reference instead of by value. The artifact resolution protocol can provide the possibility to receive both requests and responses as artifacts and can be used together with various SAML binding protocols (see section 6.6). The artifacts must provide means for the receiver to determine who sent the artifact. The artifact resolution protocol is mostly used for messages that have size constraints and to enable communication by a secure channel to avoid the need for signatures. [14]
CHAPTER 6. SAML

- The name identifier management protocol makes it possible to change value and format of a name identifier that refers to a subject. The protocol request can be performed by either service providers or identity providers and also allows for termination of name identifiers. [25]

- The name identifier mapping protocol enables mapping of name identifiers to each other. For example, if two service providers have different name identifiers for the same subject, an identity provider can provide the service providers with identifiers that can be used to refer to the subject in integration scenarios. [25]

6.6 SAML Bindings

Using lower level communication protocols (e.g. HTTP or SOAP) to transfer SAML request-response message exchanges is handled using SAML bindings. The purpose of the bindings is to make sure that SAML-conforming software which is implemented independently shall be able to interoperate. SAML v2.0 defines the following six bindings (based on [13] and [25]): HTTP Redirect Binding, HTTP POST Binding, HTTP Artifact Binding, SAML SOAP Binding, Reverse SOAP (PAOS) Binding and SAML URI Binding. In the standard, guidelines for how to specify additional bindings are also included.

- The HTTP Redirect Binding describes how SAML protocol messages can be transferred using HTTP redirect messages, which means that the SAML protocol messages can be transmitted inside URL parameters. URL lengths are in practice limited and therefore the HTTP Redirect Binding involves message encoding. The HTTP Redirect Binding is intended for use when the requester and the responder needs to communicate using a web agent (typically a web browser).

- The HTTP POST Binding describes how to transport SAML protocol messages using an HTML form control (base64-encoded). This binding is intended for use when the communication between the requester and the responder needs to be handled through a web agent.

- To transport SAML requests or SAML responses by reference using artifacts (described in Section 6.5), the HTTP Artifact Binding is used. When using the HTTP Artifact Binding there are two encoding methods which can be used, either to encode the artifact as a URL parameter or to encode the artifact in an HTML form control.

- The SAML SOAP Binding conveys how to use SOAP 1.1 to send and receive SAML protocol messages. SOAP is a lightweight protocol which uses XML to provide a messaging framework. SOAP makes it possible to exchange messages over various underlying protocols.
6.7 SAML Profiles

SAML Profiles describe how SAML assertions, SAML protocols and SAML bindings are combined and constrained to support particular usage scenarios. The profiles are developed to enhance the interoperability for the particular scenarios [25]. SAML V2.0 defines a total of eight profiles: Web Browser SSO Profile, Enhanced Client and Proxy (ECP) Profile, Identity Provider Discovery Profile, Single Logout Profile, Assertion Query/request Profile, Artifact Resolution Profile, Name Identifier Management Profile and Name Identifier Mapping Profile [24]. This set of profiles are the ones that are defined by SAML, however additional profiles can be developed [24]. Two common types of additional profiles are Interoperability Profiles and Deployment Profiles which are developed to satisfy various interoperability scenarios (e.g. federation configurations) and deployment scenarios. These types of additional scenarios are often developed since SAML leaves out a lot of options. This are options like where to use signing and encryption and how PKI should be used. When different deployments uses different options this could create problems, Interoperability Profiles and Deployment Profiles are created to avoid this type of problems.

- One of the most commonly used SAML profiles is the Web Browser SSO Profile. The Web Browser SSO Profile describes how to perform single sign-on using standard web browsers. To achieve the single sign-on the Web Browser SSO Profile uses the Authentication Request Protocol in combination with the HTTP Redirect, HTTP POST and HTTP Artifact bindings [25].

- Another common SAML profile is the Enhanced Client Proxy (ECP) Profile. This profile describe how specialized clients can use proxies to perform single sign-on using SOAP or Reverse-SOAP (PAOS) bindings [25].

6.7.1 Attribute Profiles

SAML also defines an additional type of profile called Attribute Profile. Attribute Profiles describe how SAML attributes shall be mapped into other
attribute representations [44]. SAML defines five Attribute Profiles: Basic Attribute Profile, X.500/LDAP Attribute Profile, UUID Attribute Profile, DCE PAC Attribute Profile and XACML Attribute Profile [24].

6.8 Security and Privacy in SAML

In an information technology context, privacy is often considered with both the user’s possibilities to control how identity data is shared and used, and functionality that avoids that the user’s actions at multiple service providers are linked in an inappropriate way. SAML have incorporated several mechanisms to support privacy. One of these privacy mechanisms is establishment of pseudonyms between identity providers and service providers. Another privacy support that SAML provides is the use of one-time transient identifiers, which makes service providers unable to recognize a user as the same user that previously logged in. [25]

To detect and protect against various security attacks and threats, SAML includes several security mechanisms. The main method for security between the relying party and the asserting party is to have a trust relationship. This trust relationship can for example be set up using a Public Key Infrastructure (PKI), which is also recommended in the SAML standard [25]. As a general recommendation, SAML recommends the usage of HTTP over SSL 3.0 or TLS 1.0 when integrity and confidentiality are needed [25]. Message-level confidentiality is about making sure that the SAML messages only can be read by the desired recipients and to provide message-level confidentiality. SAML V2.0 generally recommends the usage of XML Encryption [23]. Message-level data integrity is the ability to confirm that the message that is received is unaltered since it was sent and SAML V2.0 generally recommends XML Signature for message-level data integrity [23]. However, any method which provides persistent verification that the XML message is unaltered is considered sufficient for message-level data integrity [23].
Chapter 7

Requirements Analysis

This chapter explains how the requirements analysis have been conducted and presents the results.

7.1 Introduction

Requirements analysis is about understanding the customers’ fundamental problems and goals and modeling the desired behavior. There exists a variety of requirements modeling notations, for example plain text specifications or use cases. These models help to give a deeper understanding of the requirements and their relationships by revealing what questions should be asked; models that contain holes reveal unknown or ambiguous parts. While the models are developed the requirement areas that the development team and the customers has missed/overlooked can also be revealed. [33]

In section 7.2 Description of CESP a description of Cybercom Enhanced Security Platform, CESP, is presented to give a description of the context that the authentication service is a part of. Both requirements from Cybercom and their customers are described and modeled to get a clear picture of how the requirements will affect the architectural design.

The requirements elicitation was performed using semi-structured interviews and question sessions with Cybercom employees and the results of these are presented in section 7.3 Requirements List. To elicit and analyze additional security requirements, parts of the security requirements engineering framework presented in [22] were used and the results are presented in section 7.4 Additional Security Requirements. To get a web application security perspective on the requirements and make sure that the most critical web application security risks are identified, the OWASP Top 10 were reviewed and analyzed. The result of this analysis is presented in section 7.5 OWASP Top 10 Analysis. As a final step on the requirements analysis the requirements were modeled into use cases. Four different use cases are presented to give different perspectives of how the authentication service is
supposed to be used and to give both a visual representation and textual representation. These use cases are presented in section 7.6 Use cases.

7.2 Description of CESP

Cybercom Enhanced Security Platform, CESP, is an integration and security platform which provides previously closed systems with an additional security layer to be able to provide services to external users. To be able to do this in a secure way CESP enables communication with internal applications and data without compromising the confidentiality, integrity and availability of the applications. The platform consists of seven parts (shown in Figure 7.1): Authentication, Access Control, Logging, Control and Audit, Secure Communication, Service Management and Notification. [46]

![Figure 7.1: CESP Components](image)

The authentication part of CESP is called CESP-ID and makes sure that only trusted users can get access to the system. CESP-ID is an SAML identity provider and provides authentication for various actors, both physical persons and services. The actors can be authenticated using different authentication methods and assigned different attributes retrieved from various attribute sources. CESP-ID generates an assertion that is a signed proof of the authentication and can be used to gain access (e.g. access to new applications or subsequent access to applications) instead of performing a new authentication each time. CESP-ID also provides single sign-on between both applications and organization. [46]
7.2. DESCRIPTION OF CESP

Access control is handled by CESP-Access which makes sure that authorized users only can access data or services from the SAML service providers that they have right to see and use. CESP-Access uses Attribute Based Access Control (ABAC) where access is granted based on the user attributes and access policies. The rights to access data or applications are checked by an authorization engine which uses SAML assertions that contains attributes. A Policy Enforcement Point (PEP) makes the decision if a request for an activity should be performed or rejected. If the request should be performed it is sent as an XACML Request Context to the Policy Decision Point (PDP) which takes decision if the request should be accepted or rejected. This decision is based on the attributes of the requester and the attributes of the resource that is requested. The decision is sent back to the PEP as an XACML Request Context and the service can then get the decision from the PEP. [46]

The central logging service in CESP is called CESP-Log and is responsible for the logging. All components in CESP are designed to generate and send encrypted and signed log records to CESP-Log where they are stored in a protected repository. The components in the system have local log agents which buffers the log data if the central log agent would become unavailable. The central logging service handles different types of logs that have well-defined content, where the structure of the log contents can be determined by an XML-schema. [46]

CESP-Analyzer is the part of CESP that handles control and audit for all activities in the system. CESP-Analyzer continuously analyzes the logs from CESP-Log and this can be handled in real-time to provide alerts when unwanted behavior is detected. CESP-Analyzer also provides functionality to generate various types of reports and schedule automatic creation of log reports. [46]

The part of CESP that handles secure communication between networks segments within the system is called CESP-Link. This functional component provides a way to securely exchange and access services and also provides transparent communication. Transparent communication enables communication between components without them knowing each other’s location. [46]

To provide administration for all components and services CESP includes CESP-Admin which is a separate management module. CESP-Admin is available through a web interface and among other things provides possibility to: manage servers, add and remove services, add and update workflows, update settings for services, produce status reports and diagnostics. [46]

The notification part of CESP is called CESP-Notify and is a notification service that provides transparent notifications. It provides functions like signing, guaranteed delivery and secure/unsecure delivery. CESP-Notify is built upon the publish-subscribe messaging pattern which enables the sender to notify the receivers without knowing who they are or in which way they should be notified. [46]
CHAPTER 7. REQUIREMENTS ANALYSIS

7.3 Requirements List

In this thesis the requirements elicitation was performed using semi-structured interviews and question sessions with Cybercom employees. The semi-structured interview form was chosen since it provides more flexible communication during the interviews and new questions that rise during the interview can be brought up and discussed. Only Cybercom employees working with CESP was interviewed in this study since the timeframe was limited. From the semi-structured interviews and the question sessions a list of elicited requirements was prepared and the requirements were grouped according to their origin to provide traceability. Here requirements originating from Cybercom are labeled internal requirements and requirements originating from customers are labeled customer requirements. This list of elicited requirements was then sent to two of the customers that use CESP to be evaluated, they were especially asked to look for missing requirements and requirements that were misunderstood.

This section presents the final result of these requirement elicitation and analysis steps. At first a prioritized list of the five most important quality attributes and a list of the main functions of the authentication service are presented, the contents of these two lists originate from Cybercom requirements and are not something that comes from their customers in the first place. Further on the specific requirements lists originating from Cybercom and the customers are presented. The requirements are marked with a priority level where priority level one (1) is requirements that the architecture shall include. Priority level two (2) is requirements that the architecture should include and priority level three (3) are requirements that would be good to have in the architecture.

Prioritized Quality Attributes

1. Security
2. Availability
3. Performance
4. Redundancy
5. Scalability

Functions that the architecture shall include

F1. Handling of requests (login & logout).
F2. Handling of validation of SAML 2.0 profiles.
F3. Handling of authentication of users.
F4. Handling of user attributes and attribute sources.
7.3. REQUIREMENTS LIST

F5. Handling of configuration management.
F6. Handling of logging.

Requirement Priority Levels

- Level 1: Requirements that the architecture shall include.
- Level 2: Requirements that the architecture should include.
- Level 3: Requirements that would be good to have in the architecture.

7.3.1 Internal Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1.1</td>
<td>The architecture shall comply with the SAML 2.0 standard.</td>
<td>1</td>
</tr>
<tr>
<td>R1.2</td>
<td>The architecture shall provide possibility to act as an SAML 2.0 IdP, enable identification of actors and make it possible to produce SAML 2.0 assertions as verification.</td>
<td>1</td>
</tr>
<tr>
<td>R1.3</td>
<td>The architecture shall make single sign-on possible.</td>
<td>1</td>
</tr>
<tr>
<td>R1.4</td>
<td>The architecture shall make sure that all credentials transferred to the authentication service are transmitted over encrypted communication channels.</td>
<td>1</td>
</tr>
<tr>
<td>R1.5</td>
<td>Authentication using electronic IDs shall support validation of certificates using Trusted Security Server (TSS).</td>
<td>1</td>
</tr>
<tr>
<td>R1.6</td>
<td>The architecture shall support an additional step to be performed after the user credentials are received by the authentication service and before the actual authentication takes place.</td>
<td>1</td>
</tr>
<tr>
<td>R1.7</td>
<td>The architecture shall perform validation of SAML requests.</td>
<td>1</td>
</tr>
<tr>
<td>R1.8</td>
<td>The architecture shall be able to perform validation according to various SAML profile rules.</td>
<td>1</td>
</tr>
<tr>
<td>R1.9</td>
<td>The architecture shall make login possible by using a web interface.</td>
<td>1</td>
</tr>
<tr>
<td>R1.10</td>
<td>The architecture shall make it possible to use login timeouts to discourage brute force attacks.</td>
<td>1</td>
</tr>
<tr>
<td>R1.11</td>
<td>The architecture shall make it possible to limit the amount of failed login attempts and perform appropriate actions when the limit is reached.</td>
<td>1</td>
</tr>
<tr>
<td>R1.12</td>
<td>The architecture shall provide possibility to use persistent identifiers (obfuscate the user identity) to enhance user privacy so that the users activities cannot be linked between different service providers.</td>
<td>1</td>
</tr>
</tbody>
</table>

Continued on next page
CHAPTER 7. REQUIREMENTS ANALYSIS

Table 7.1 – continued from previous page

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1.13</td>
<td>The architecture shall provide possibility to use transient identifiers (one-time-use) to enhance user privacy so that a service provider won’t be able to recognize the user as the same user that previously logged in.</td>
<td>1</td>
</tr>
<tr>
<td>R1.14</td>
<td>The architecture shall make it possible to configure the authentication service using metadata according to the SAML V2.0 Metadata specification.</td>
<td>1</td>
</tr>
<tr>
<td>R1.15</td>
<td>The architecture shall make it possible to load SAML V2.0 Metadata from an external source.</td>
<td>1</td>
</tr>
<tr>
<td>R1.16</td>
<td>The architecture shall provide possibility for administrators to manage all configuration settings using an administration user interface.</td>
<td>1</td>
</tr>
<tr>
<td>R1.17</td>
<td>The architecture shall provide support for load balancing and failovers.</td>
<td>1</td>
</tr>
<tr>
<td>R1.18</td>
<td>The Architecture shall provide support for sessions exclusively stored in databases.</td>
<td>2</td>
</tr>
<tr>
<td>R1.19</td>
<td>The architecture shall make it possible to change settings during runtime, i.e. saving settings in a database.</td>
<td>2</td>
</tr>
<tr>
<td>R1.20</td>
<td>The architecture shall provide support for using different SAML 2.0 profiles.</td>
<td>2</td>
</tr>
<tr>
<td>R1.21</td>
<td>The architecture shall provide support for using SAML 2.0 interoperability profiles.</td>
<td>2</td>
</tr>
<tr>
<td>R1.22</td>
<td>The architecture shall provide support for using SAML 2.0 deployment profiles.</td>
<td>2</td>
</tr>
<tr>
<td>R1.23</td>
<td>The architecture shall provide support for configuration using SAML 2.0 attribute profiles (attribute profiles gives definitions of how specific attributes are described).</td>
<td>3</td>
</tr>
<tr>
<td>R1.24</td>
<td>The architecture shall provide support for using metadata-ui to configure e.g. login-pages.</td>
<td>3</td>
</tr>
<tr>
<td>R1.25</td>
<td>The architecture shall make it possible for organizations to make identity federations to provide single sign-on over the organizational boundaries.</td>
<td>3</td>
</tr>
<tr>
<td>R1.26</td>
<td>The architecture shall support the new standard for electronic IDs (Eid2.0) in Sweden.</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7.1: Internal requirements

7.3.2 Customer requirements
### 7.3. REQUIREMENTS LIST

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2.1</td>
<td>The architecture shall make it possible to configure allowed authentication methods for each relying party.</td>
<td>1</td>
</tr>
<tr>
<td>R2.2</td>
<td>The architecture shall make it possible for an relying party to tell the authentication service which authentication method to use for a specific user.</td>
<td>1</td>
</tr>
<tr>
<td>R2.3</td>
<td>The architecture shall provide Single Logout according to SAML 2.0 Single Logout Profile.</td>
<td>1</td>
</tr>
<tr>
<td>R2.4</td>
<td>The architecture shall support hardware token authentication using smart cards containing certificates.</td>
<td>1</td>
</tr>
<tr>
<td>R2.5</td>
<td>The architecture shall support both CRL (Certificate Revocation List) and OCSP (Online Certificate Status Protocol) to check for revoked certificates.</td>
<td>1</td>
</tr>
<tr>
<td>R2.6</td>
<td>The architecture shall support two-factor authentication using smart cards combined with a PIN number.</td>
<td>1</td>
</tr>
<tr>
<td>R2.7</td>
<td>The architecture shall support authentication against LDAP, AD or internal databases using username and password.</td>
<td>1</td>
</tr>
<tr>
<td>R2.8</td>
<td>The architecture shall support token based one-time-password (OTP) authentication using SMS OTP.</td>
<td>1</td>
</tr>
<tr>
<td>R2.9</td>
<td>The architecture shall support authentication using the current standard for electronic IDs in Sweden.</td>
<td>1</td>
</tr>
<tr>
<td>R2.10</td>
<td>The architecture shall provide possibility to manage authentication with an assurance level corresponding to LoA 3 and be able to convey information about which LoA that is used.</td>
<td>1</td>
</tr>
<tr>
<td>R2.11</td>
<td>The architecture shall provide attribute retrieval services for relying parties.</td>
<td>1</td>
</tr>
<tr>
<td>R2.12</td>
<td>The architecture shall support retrieval of attributes from certificates.</td>
<td>1</td>
</tr>
<tr>
<td>R2.13</td>
<td>The architecture shall support gathering of user attributes from internal attribute sources.</td>
<td>1</td>
</tr>
<tr>
<td>R2.14</td>
<td>The architecture shall support gathering of user attributes from external attribute sources using web service requests.</td>
<td>1</td>
</tr>
<tr>
<td>R2.15</td>
<td>Gathering of user attributes from internal sources shall support gathering user attributes from LDAP or AD.</td>
<td>1</td>
</tr>
<tr>
<td>R2.16</td>
<td>The architecture shall provide the possibility to temporary inactivate relying parties.</td>
<td>1</td>
</tr>
<tr>
<td>R2.17</td>
<td>The architecture shall make it possible for relying parties to use encapsulated flow (iframe).</td>
<td>1</td>
</tr>
<tr>
<td>R2.18</td>
<td>The architecture shall provide audit logging.</td>
<td>1</td>
</tr>
<tr>
<td>R2.19</td>
<td>The architecture shall provide performance logging.</td>
<td>1</td>
</tr>
</tbody>
</table>
7.4 Additional Security Requirements

To gather additional security requirements and to get an additional perspective of the requirements the security requirements engineering framework described in [22] was used for requirements elicitation and analysis. In the framework, four stages for the security requirements engineering process are defined. The framework is rather extensive and therefore only three out of four parts were used.

The first stage of the framework is called Identify Functional Requirements and the only requirement that the framework has on this stage is that a representation of the system context shall be produced. The second stage is called Identify Security Goals and this stage requires that candidate assets are identified. A harm analysis should be done for each of the identified assets and should finally result in a set of identified security goals. Harm is generally caused by the negation of confidentiality, integrity or availability. The third stage is called Identify Security Requirements and here the security requirements are defined as constraints of the functional requirements that are required to fulfill the security goals. The fourth stage is

Table 7.2: Customer requirements
7.4. ADDITIONAL SECURITY REQUIREMENTS

called Verification of the System and consists of two steps. The first step is about creating formal arguments to prove that the security requirements are fulfilled and the second step is about creating a set of structured informal arguments to support the assumptions about the system that was made in the formal arguments. This fourth stage of the framework was not performed in this thesis due to the limited timeframe.

Stage 1: Identified Functional Requirements

The main functional requirements was identified in section 7.3 Requirements List and consists of:

F1. Handling of requests (login & logout).
F2. Handling of validation of SAML 2.0 profiles.
F3. Handling of authentication of users.
F4. Handling of user attributes and attribute sources.
F5. Handling of configuration management.
F6. Handling of logging.

Stage 2: Identify Security Goals

In this stage the candidate assets was identified and a list of possible threats that could occur was prepared. Based on these threats a list of the most important security goals was identified.

Identified Candidate Assets

- User input credentials (for example username / password)
- Stored authentication information (for example username / password hash)
- Attributes
- Assertions
- Configuration information
- Logs

List of Possible Threats on Assets

- The user input credentials can be read by an unauthorized user.
- The user input credentials can be manipulated.
CHAPTER 7. REQUIREMENTS ANALYSIS

- The stored authentication information can be read by an unauthorized user.
- The stored authentication information can be manipulated.
- The stored authentication information can be erased.
- The stored authentication information can become unavailable.
- The attributes can be read by an unauthorized user.
- The attributes can be manipulated.
- The attributes can be erased.
- The attributes can become unavailable.
- The assertions can be read by an unauthorized user.
- The assertions can be manipulated.
- The configuration information can be read by an unauthorized user.
- The configuration information can be manipulated by an unauthorized user.
- The configuration information can be erased.
- The configuration information can become unavailable.
- The logs can be read by an unauthorized user.
- The logs can be manipulated.

List of Identified Security Goals

1. The input credentials from the users are to be kept confidential.
2. The integrity of the input credentials from the users is to be preserved.
3. The stored authentication credentials are to be kept confidential.
4. The integrity of the stored authentication credentials are to be preserved.
5. The stored authentication credentials are to be available.
6. The attributes are to be kept confidential.
7. The integrity of the attributes is to be preserved.
8. The attributes are to be available.
9. The assertions are to be kept confidential.
7.5. **OWASP TOP 10 ANALYSIS**

10. The integrity of the assertions is to be preserved.
11. The configuration information are to be kept confidential.
12. The integrity of the configuration information is to be preserved.
13. The configuration information is to be available.
14. The logs are to be kept confidential.
15. The logs are not be modified undetectably.

**Stage 3: Identify Security Requirements**

By including the security goals as constraints on the functional requirements a list of security requirements was identified.

**SR1.** The architecture shall handle requests (login & logout) only if the credentials from the user is kept confidential and the integrity of the credentials from the user is preserved.

**SR2.** The architecture shall handle authentication of users while keeping the stored credentials and assertions confidential, preserving the integrity of the stored credentials and the assertions and keeping the stored credentials available.

**SR3.** The architecture shall handle user attributes and attribute sources while keeping attributes confidential, preserving the integrity of the attributes and keeping the attributes available.

**SR4.** The architecture shall handle configuration management while keeping configuration information confidential, preserving the integrity of the configuration information and keeping the configuration information available.

**SR5.** The architecture shall handle logging without compromising the confidentiality or the integrity of the logs.

### 7.5 OWASP Top 10 Analysis

The Open Web Application Security Project (OWASP) Top 10 provides a summary of the most critical web application security risks facing an organization today (an overview is shown in Table 7.3).

<table>
<thead>
<tr>
<th>Nr</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Injection</td>
</tr>
<tr>
<td>2</td>
<td>Cross-Site Scripting (XSS)</td>
</tr>
<tr>
<td>3</td>
<td>Broken Authentication and Session Management</td>
</tr>
</tbody>
</table>
CHAPTER 7. REQUIREMENTS ANALYSIS

<table>
<thead>
<tr>
<th>Nr</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Insecure Direct Object References</td>
</tr>
<tr>
<td>5</td>
<td>Cross-Site Request Forgery (CSRF)</td>
</tr>
<tr>
<td>6</td>
<td>Security Misconfiguration</td>
</tr>
<tr>
<td>7</td>
<td>Insecure Cryptographic Storage</td>
</tr>
<tr>
<td>8</td>
<td>Failure to Restrict URL Access</td>
</tr>
<tr>
<td>9</td>
<td>Insufficient Transport Layer Protection</td>
</tr>
<tr>
<td>10</td>
<td>Unvalidated Redirects and Forwards</td>
</tr>
</tbody>
</table>

Table 7.3: OWASP Top 10 Overview

The goal of OWASP Top 10 is to raise awareness about web application security and in this thesis the OWASP Top 10 was used to get a web application security perspective on the requirements. It was used as an additional foundation for analyzing the requirements to make sure that the most critical web application security risks have been assessed and to discover additional areas where potential security risks exist. The OWASP Top 10 constantly changes and in this thesis the 2010 version was used for the analysis. [38]

The first security risk introduced in OWASP Top 10 is **Injection**. Injection flaws occur when untrusted data is sent to an interpreter, often as a query or command. Example flaws can be SQL and LDAP injections and can cause the interpreter to execute unwanted commands. This is something that needs to be handled for the authentication service by for example validating the input from the user as well as the internal & external data used for authentication to keep untrusted data away from commands and queries. This vulnerability is related to e.g. R1.7 “The architecture shall perform validation of SAML requests” and R1.8 “The architecture shall be able to perform validation according to various SAML profile rules”. This means that in the authentication service architecture it is important to make sure that the validation part is performed as early as possible.

The second security risk introduced is **Cross-Site Scripting (XSS)** which is something that is preferably handled in the implementation of the authentication service and therefore not a part of the architectural design.

**Broken Authentication and Session Management** is the third security risk introduced in OWASP Top 10. Custom authentication and session management schemes are often used and it’s hard to build them correctly. If not built correctly an attacker can use flaws in the authentication or the session management functions to steal user identities. This could be things like revealed account information, passwords or session IDs. Broken authentication and session management is absolutely something that needs to be considered for the authentication service architecture since authentication is the main purpose of the service. Therefore the business value of the affected data and the impact of public exposure of a vulnerability of this kind would be huge. To prevent broken authentication and session management, the
7.5. **OWASP TOP 10 ANALYSIS**

Main assets that need to be protected are the credentials and the session IDs. They need to be stored using appropriate encryption or hashing and should not be able to be guessed or overwritten by using weak management functions. Credentials should only be sent over secure connections and session IDs shall never be disclosed and should have timeouts. The Broken Authentication and Session Management flaws are related to requirement R1.4 “The architecture shall make sure that all credentials transferred to the authentication service are transmitted over encrypted communication channels”. Another architectural decision that originates from the Broken Authentication and Session Management threats is to make sure to store credentials and session information in an appropriate way using encryption.

**Insecure Direct Object References** is another security risk that is introduced. This risk is concerned with an attacker that is authorized to use the system, and who can change a parameter value that directly refers to another object, and by that gain access to objects that the user is not authorized for. This is something that needs to be considered during the implementation of the authentication service and is therefore not part of the architectural design.

The fifth security risk introduced is **Cross-Site Request Forgery (CSRF)** which is about attackers sending forged requests that use vulnerabilities that makes it is possible to predict details of activities. This is something that needs to be considered during the implementation of the authentication service and is therefore not part of the architectural design.

The OWASP Top 10 also introduces **Security Misconfiguration** which an attacker can use in various ways to gain unauthorized access or information about the system. This could be things like using flaws in unpatched software, using default accounts that are not disabled or improperly configured security settings. An important architectural prevention is to make sure that the application architecture provides strong network security and separates components which lowers the risk for security misconfigurations.

The seventh security risk introduced is **Insecure Cryptographic Storage** which is mainly not about attackers breaking the crypto, but rather about using other methods to gain access to sensitive information. This could for example be unsafe handling of keys and storage, getting cleartexts of data in various ways and the use of unsalted hashes. However the largest flaw in this area is not using encryption at all for sensitive data. Therefore an important question that needs to be handled in the architectural design of the authentication service is which areas that store sensitive information and therefore need to be protected by encryption. Using encryption for storing sensitive data is not part of the requirements in the requirements list and therefore considered an additional security concern needed to be handled by the architectural design.

**Failure to Restrict URL Access** is another security risk introduced in the OWASP Top 10, this security risk is mainly about implementation and configuration and is therefore not considered a part of the architectural design.
CHAPTER 7. REQUIREMENTS ANALYSIS

The ninth security risk introduced is Insufficient Transport Layer Protection. Without proper transport layer protection an attacker could monitor the network traffic to capture sensitive data and account information. Applications often use SSL/TLS during authentication but forget to use it in other parts of the communication with the user. This could be communication where the applications request and receive information saved in session cookies. These session cookies could then be used to hijack the session. Insufficient transport layer security is absolutely something that needs to be addressed in the architectural design of the authentication service. This risk is related to requirement R1.4 “The architecture shall make sure that all credentials transferred to the authentication service are transmitted over encrypted communication channels”. In addition, the insufficient transport layer security risk also indicates that the authentication service architecture should use transport layer protection like SSL/TLS for all network traffic and only support strong algorithms.

The last security risk introduced is Unvalidated Redirects and Forwards and is concerned with redirects and forwards that could be used by an attacker for malicious activities if the requests and forwards are not validated. This is something that needs to be considered during the implementation of the authentication service and is therefore not part of the architectural design.

7.6 Use Cases

This section analyses the requirements by providing four different use cases that provide both visual and textual descriptions of different possible scenarios where the authentication service is used. The use cases show the interaction between the different actors and the system.

7.6.1 Use Case 1: Emergency and Rescue Enterprise

A nurse at a healthcare center in Sweden wants to order an ambulance from the emergency and rescue enterprise so she connects to the ambulance ordering application. To make sure that she is who she claims to be she needs to login and prove her identity. Therefore the ambulance ordering system connects to the authentication service and the nurse is prompted to login using her smart-card and a PIN-number. The authentication service connects to an external web service to validate the identity and another external web service to gather the user attributes. The authentication service then creates an assertion that contains the necessary authentication information and sends it to the ambulance ordering application to be used for access control in the next step.

When the nurse has completed the ambulance ordering she wants to logout. She uses the logout function of the ambulance ordering application
7.6. USE CASES

Figure 7.2: Use Case 1, Emergency and Rescue Enterprise

which sends a logout request to the authentication service. The authentication service makes sure that logout is performed correctly and then sends a confirmation to the ambulance ordering application and the nurse.

7.6.2 Use Case 2: Umbrella Organization

An employee at a small company that is a part of the umbrella organization wants to check his e-mail. Therefore he uses his web browser to connect to the umbrella organization webmail application. To make sure that he is who he claims to be he needs to authenticate and prove his identity. Therefore the webmail application connects to the authentication service and the employee is prompted to enter his username and password. The authentication service connects to an external LDAP directory to validate the identity and gather user attributes. The authentication service then creates an assertion that contains the necessary authentication information and sends it to the webmail application to be used for access control in the next step.

Now the employee at the small company decides that he wants to perform time reporting. Therefore he uses his web browser to connect to the umbrella organization time report application. This time no authentication is needed since he is already authenticated using the authentication service. When the employee is finished with the time reporting he wants to logout. He uses the Single Logout functionality in the time report application and that sends a request to the authentication service that all applications shall be
logged out. When the authentication service has correctly logged out the employee from all active applications he receives a confirmation.

### 7.6.3 Use Case 3: Banking Service

A bank customer wants to login to the bank system using her Swedish electronic ID. She visits her bank’s web site and presses the login button. To be able to login using her Swedish electronic ID she is redirected to the authentication service to prove her identity. The authentication service prompts her to authenticate using her Swedish electronic ID and then connects to the Trusted Security Server (TSS). TSS is a Cybercom product for validating electronic IDs and is used to validate the certificate included in her electronic ID. The authentication service then creates an assertion that contains the authentication information needed for access control in the next step. When she has finished using the banking system she presses the logout button which sends a logout request to the authentication service. The authentication service makes sure that logout is performed correctly and then sends a confirmation to the bank customer.

### 7.6.4 Use Case 4: Company Federations

A user that has authenticated with company A wants to access company B’s secret application. Company B trusts company A and therefore want
7.6. USE CASES

Figure 7.4: Use Case 3, Banking Service

to make use of the previous authentication to avoid an additional authentication step. Therefore when the user tries to access company B’s secret application the authentication service first connects to company A’s authentication service. Since the user has already authenticated there, the already existing authentication information is used so that no additional authentication is needed. When the user has finished using company B’s secret application he uses the logout function which sends a logout request to company B’s authentication service. Company B’s authentication service makes sure that the logout is performed correctly and sends back a confirmation to the user.

7.6.5 Use Case 5: Large Identity Federations

In this use case we have a large federation among schools in a country. One of the schools provides a math application that the other schools want to use. A student in another school in the federation has heard about math application and wants to use it. To do this he needs to authenticate to prove that he is a student that is part of this identity federation. When he connects to the math application he is provided a list of available identity providers. To always be able to provide a list of available identity providers, the application connects to the central discovery service of the identity federation and retrieves the list. The student chooses to use the Identity Provider (IdP) at his local school which in this case is the authentication service in
CHAPTER 7. REQUIREMENTS ANALYSIS

Figure 7.5: Use Case 4, Company Federations

CESP. To perform the authentication, the student is prompted to enter his username and password, which is verified against a local database in the authentication service. The authentication service then creates an assertion that contains the necessary authentication information and sends it to the math application to be used for access control in the next step.

When the student has finished using the math application for this time he wants to logout. He uses the logout function on the math application which sends a logout request to the authentication service. The authentication service makes sure that logout is performed correctly and then sends a confirmation to the math application and the student.
7.6. USE CASES

Figure 7.6: Use Case 5, Large Identity Federations
Chapter 8

Architectural Design

This chapter describes the architectural design of the authentication service and explains methods and rationale for the design decisions.

8.1 Introduction

The architectural design of the authentication service is based on the requirements analysis (see Chapter 7) and is described using the 4+1 View Model of Architecture (see Section 2.6.3). The design method and rationale is described in Section 8.2 and is followed by an catalog of the different views in Section 8.3. To provide the reader with a brief overview of the design of the authentication service architecture an additional system overview is described in Section 8.4.

8.2 Method and Rationale

For the architectural design the 4+1 View Model of Architecture (described in Section 2.6.3) have been chosen as the architectural definition model. This model uses four different views (logical view, process view, development view and physical view) plus additional scenarios and has been chosen since it is a model that has been around for a long time, is well proven and stable. The reason for choosing a method with multiple views is that it helps the stakeholders manage the complexity of the authentication service by separating the different concerns into separate views.

Another approach that would have been possible is to use Attribute-Driven Design (described in Section 2.6.1), which is an architectural definition model which uses only three different views (module view, component and connector view, deployment view) [5] for the architectural design. The ADD model was not considered sufficient to provide a more comprehensive picture of the architecture’s components and also to ensure that misunder-
8.3 View Catalog

The architectural design contains the five main views from The 4+1 View Model of Architecture plus an additional system overview. The system overview is described in Section 8.4 and gives a brief overview of the authentication service. The system overview can be interesting for all stakeholders. The logical view (Section 8.5) describes how the authentication service is divided into classes which has communicational and behavioral obligations and shows the end-user perspective of the architecture. The different processes that the authentication service is divided into during execution is explained in the Process View (Section 8.6) and is very useful for integrators. The Development View (Section 8.7) describes how the authentication service

standings do not arise. Another reason for choosing the 4+1 View Model of Architecture is that it provides graphical description of the views. UML is a standardized modeling language that is widely used within the software community and provides a lot of benefits since it’s often known by many stakeholders. UML has also become a de facto standard notation for documenting software architectures in the industry [39] and therefore it has been chosen as a base for the descriptions of the architectural views. Other Architecture Description Languages (ADLs) (described in Section 2.5) could have been chosen for the architectural design but the fact that UML is a visual language was considered a great advantage compared to other ADLs, which often are non visual and quite restrictive. Another approach that could have been used is to combine UML and some other ADL. That approach were discarded due to the limited time frame of this thesis. There are many other methods and models which could have been used to describe the architectural design, e.g. the approach proposed by ISO/IEC/IEEE 42010 (described in Section 2.6.2), The Reference Model for Open Distributed Processing (described in Section 2.6.4) etc. However the 4+1 View Model of Architecture was considered more well proven and was therefore considered more appropriate for the authentication service architecture considering that security is the most important quality attribute. The 4+1 View Model of Architecture contains approaches for describing the correspondence between the different views. This part of the model have been omitted due to the limited time frame of the thesis.

To make the architectural design of the authentication service as conformative as possible, the major architectural design approach chosen was information-hiding-based decomposition style (described in Section 2.4.3) which provides high modifiability [15]. The architectural design was also based on the shared-data-style (described in Section 2.4.7) to support modifiability and enhance the performance and availability [15] of the authentication service. XML has been chosen for configuration data storage and communication since it provides a platform-independent description of data and allows dynamic configuration.
CHAPTER 8. ARCHITECTURAL DESIGN

is organized in layers and modules that can be used by development teams, testers and managers to divide the workload. A description of the mapping between the software and the underlying hardware is explained in the Physical View (Section 8.8) and is especially useful for system engineers. The Scenarios (Section 8.9) give use cases that describe how the elements of the other views (logical, process, development and physical) work together and is useful for all all stakeholders and evaluators. To document the views, the approach for documenting views described in [15] have been used as a base and been modified to suit the need of the architectural design of the authentication service.

8.4 System Overview

Figure 8.1: System Overview

A brief overview of the design of the authentication service architecture is shown in Figure 8.1 on page 70. This system overview is only intended for the reader to quickly grasp the overall structure of the architecture and omits many important aspects of the architecture. The complete architecture is described in the following sections.
8.5 LOGICAL VIEW

8.5 Logical View

The logical view shows how the authentication service is divided into classes with communicational and behavioral responsibilities.

8.5.1 Primary Presentation

The primary presentation of the logical view gives an overview of the functionality that the authentication service provides to the end-users and is shown in Figure 8.2 on page 71.

8.5.2 Element Catalog and Context Description

The logical view is derived from the requirements analysis and consists of a total of 10 class categories and their usage relations. Usage relations for the Configuration Service and the Logging Service have deliberately been omitted since it would presumably make it more difficult to grasp the overall structure of the logical view. All other classes can load separate configurations from the configuration service and therefore there is a usage relation between all other class categories and the configuration service. The logging
service shall provide logging functionality to all the other class categories (amount of logging can be configured) and therefore there is a usage relation from all other classes to the logging service.

The Request and User Interface Service provides handling of SAML requests from user agents and relying parties. The service provides possibilities to present a web user interface to the user and also provides possibilities to handle direct SAML requests using SOAP and HTTP. The request and user interface service uses the validation service for validation of SAML request and therefore only handles the management of the requests.

The Validation Service provides validation of the SAML requests that is received from the Request and User Interface Service. The main functionality of the Validation Service is SAML profile validation, but the service also provides additional validation (e.g. time-validation and signature validation).

The Configuration Service works as a configuration repository for the other class categories, where they can gather the latest configuration updates that is connected to their functionality. The Configuration Service also includes an administration user interface where administrators can make changes to the authentication service configuration.

The authentication service provides both audit and performance logging to the end-users through the Logging Service which handles logging requests from all the other class categories. The logging requests are processed by the Logging Service and sent to a central logging service (temporary storage is provided by the Logging Service if the central logging service is unavailable).

The Authentication Core Service handles the processing of requests and provides possibilities to combine e.g. various authentication methods and attributes. The Authentication Core also allows usage of different federation settings and possibilities to interact with legacy systems.

Relying parties often want to use various authentication methods and also be able to combine authentication methods, this is made possible through the Authentication Method Service which is used by the Authentication Core Service.

The Legacy Service makes it possible to use and communicate with legacy systems that does not support SAML. The legacy service is only used by the Authentication Core Service.

Federation functionality is provided by the Federation Service, which allows the authentication service to be part of an identity federation. This includes functionality like using a Central Metadata Register (CMR) for configuration of the identity federation that the authentication service is a part of. A CMR is a central register where metadata definitions are stored to provide consistency among organizations. The Federation Service is only used by the Authentication Core Service.

The Attribute Service provides the end-users functionality to handle various attributes and attribute sources. Additional functionality provided by the Attribute Service is processing of attributes, e.g. converting personal
numbers to a specific format. The Attribute Service is only used by the Authentication Core.

Generation of the various resulting SAML responses that are sent to the user agents or the relying parties are handled by the Response Generation Service which is used by the Authentication Core Service.

8.5.3 Rationale

The logical view was divided into several separate class categories to make a clear separation of the different functional responsibilities of the authentication service. The logical view could also have been divided into various classes instead of class categories, however this would have made the logical view much more complicated since the amount of classes would have been much greater than the amount of class categories. The approach of using class categories also provides abstraction and encapsulation which provides a clear picture of the functional responsibilities included in the authentication service.

8.6 Process View

The process view shows the different processes that the authentication service is divided into during execution and describes some aspects of non functional requirements, e.g. concurrency, scalability and performance.

8.6.1 Primary Presentation

The primary presentation of the process view consists of the process blueprint shown in Figure 8.3 on page 74. The process blueprint shows an overview of the different processes that exist within the authentication service and message relations between the processes. The main architectural style used to design the structure of the primary presentation of the process view is the Communicating Processes Style (described in Section 2.4.1).

8.6.2 Element Catalog and Context Description

The process view is structured into a set of 11 processes. These processes communicate using messages and their message relations are shown in the primary presentation except for the message relations to the configuration process and to the logging process. These message relations have deliberately been left out since including them in the blueprint would make it much more difficult to grasp the overall process structure. All the other processes have a message relation to the logging process and a bidirectional message relation to the configuration process to make authentication service as conformative as possible by being able to request and receive updated configuration settings and rules.
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.3: Process Blueprint

The request and user interface process is a periodic process (a cyclical process which is running over and over again) that handles the input requests and user interface generation. When an complete SAML request is received it it is forwarded to the Validation Process. The process flow of the request and user interface process is described in Figure 8.4 on page 75.

The validation process is designed according to the interceptor pattern (described in Section 3.4.3) to provide additional security by validating the input requests. The validation process is configured to be able to validate SAML profiles and there is also an additional option to perform an additional customized step before the validated request is sent to the authentication core process. The process flow of the request and validation process is described in Figure 8.5 on page 76.

Configuration of the authentication service is handled by the configuration process. The configuration process is a periodic process that handles the administration user interface that administrators can use to manage all configuration settings. All other processes can also send messages to the configuration process to receive updated configuration settings. The process flow of the configuration process is described in Figure 8.6 on page 74.
8.6. PROCESS VIEW

Figure 8.4: Request and User Interface Process Flow

77. The logging process handles logging request messages from all the other processes. When a log request message is received by the logging process it is sent to an central logging service, however if the central logging service is unavailable the logs are stored in a temporary logging storage (signed and encrypted). The process flow of the logging process is described in Figure 8.7 on page 78.

Handling session requests and updates are the purpose of the session process. The session process has a bidirectional message relation with the authentication core process and a message relation to the logging process. The messages from the authentication core process is of two types, either a
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.5: Validation Process Flow

request for an active session or an update session request. The process flow of the session process is described in Figure 8.8 on page 79.

The response generator process handles generation of SAML responses. A bidirectional message relation is provided with the authentication core process to make sure that the authentication core process can send the information needed for generation of SAML responses and to make sure that the Response Generator Process can return the result of the SAML response generation so that the authentication core can send an update to the session process about the result. An alternative solution here would be to have a message relation to the session process as well, but this was discarded since it would provide higher coupling between the processes which is considered good practice to avoid. The process flow of the response generator process is described in Figure 8.9 on page 80.
8.6. PROCESS VIEW

Requests from the validation process is sent to the authentication core process which decides which actions are to be performed. The authentication core process have bidirectional message relations to the authentication method process, the legacy process, the attribute process and the federation process which provides possibilities for the authentication core process to use the services provided by these processes. The process flow of the authentication core process is described in Figure 8.10 on page 81.

The authentication method process handles the different authentication methods. A bidirectional message relation is provided with the authentication core process to receive requests and return the results from the authentication methods execution. The authentication method process is designed according to the interceptor pattern (described in Section 3.4.3) to provide additional security. The process flow of the authentication method process is described in Figure 8.11 on page 82.
 Handling of legacy systems without SAML support is performed by the legacy process. The legacy process has bidirectional message relation to the authentication core process to be able to receive requests for the legacy systems and to be able to reply with the results of the different actions to the authentication core. The legacy process is designed according to the interceptor pattern (described in Section 3.4.3) to provide additional security. The process flow of the legacy process is described in Figure 8.12 on page 83.

The attribute process handles retrieval of attribute request on the behalf of authentication core process. This means that there is a bidirectional message relation between the attribute process and the authentication core process to exchange attribute requests and the attribute results. Attribute processing is also included as an additional part of the execution of the attribute process. The attribute process is designed according to the interceptor pattern (described in Section 3.4.3) to provide additional security. The process flow of the attribute process is described in Figure 8.13 on page 84.

Possibilities to be included in different identity federations are handled by the federation process by loading appropriate federation settings (from a Central Metadata Register (CMR) or local federation rules) and returning settings needed by the authentication core process. These returned federation settings are needed by the authentication core to be able to perform its actions in a federated environment. Therefore there is a bidirectional mes-
8.6. PROCESS VIEW

Figure 8.8: Session Process Flow

There is a relationship between the authentication core process and the federation process. The federation process is designed according to the interceptor pattern (described in Section 3.4.3) to provide additional security. The process flow of the federation process is described in Figure 8.14 on page 85.
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.9: Response Generator Process Flow

8.6.3 Rationale

The process view was structured into a process blueprint and a set of process flows (one flow for each process) to provide a brief overview of the process structure and to supplement it with a more detailed flow description for
8.6. PROCESS VIEW

Figure 8.10: Authentication Core Process Flow

each process which provides more depth. This approach was chosen to provide a runtime description of the authentication service and is a form of Component-and-Connector structure (described in Section 2.2.2).
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.11: Authentication Method Process Flow

8.7 Development View

The development view describes the software module organization and how the software is packaged into components and submodules that can be developed and tested by individual developers or teams of developers.
8.7. DEVELOPMENT VIEW

Figure 8.12: Legacy Process Flow

8.7.1 Primary Presentation

The primary presentation of the development view (shown in Figure 8.15 on page 86) gives an overview of how the authentication system is structured into different modules with individual responsibilities and their dependences to the other modules. In the primary presentation all dependences related to the configuration module and the logging module have deliberately been omitted since it would probably make it more difficult to grasp the overall structure of the development view.

8.7.2 Element Catalog and Context Description

The development view is divided into four layers with well-defined responsibilities:
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.13: Attribute Process Flow

- The **Front Layer** includes Request and User Interface Module, Validation Module and Authentication Core Module and handles incoming SAML requests and outgoing SAML responses.

- The **Domain Layer** handles domain processing and includes Response Generator Module, Session Module, Authentication Method Module, Attribute Module, Federation Module, Legacy Module and Attribute Processing Module.

- The **Data Layer** handles all data storage and includes credential storage, configuration storage, attribute storage and session storage.

- The **Cross-Cutting System Layer** handles configuration, logging
8.7. DEVELOPMENT VIEW

Figure 8.14: Federation Process Flow

and SAML profiles and includes Configuration Module, Logging Module and SAML Profile Module.

All modules implements the same interface to use the logging module (to enhance reusability) which enables the modules to send logging requests according to a specific format to the Logging Module.

The Request and User Interface Module is shown in Figure 8.16 on page 87 and handles incoming request communication with the clients (both user agents and relying parties). The module is responsible for handling various
CHAPTER 8. ARCHITECTURAL DESIGN

SAML request according to the active SAML Profile Rules retrieved from the SAML Profile Module. Another responsibility of the request and user interface module is to generate web interfaces that can be customized using metadata retrieved by the interface between the Configuration Module and the Request and User Interface Module. The interface between the Request and User Interface Module and the Validation Module will make it possible for the Request and User Interface Module to send a SAML request to the Validation Module to perform validation according to various rules.

The Validation Module is shown in Figure 8.17 on page 87 and handles validation of incoming SAML requests. The interface to SAML Profile
8.7. DEVELOPMENT VIEW

Module will make it possible for the validation module to load SAML Profile Rules from the SAML profile module. Other validation rules can be loaded through the interface with the configuration module (XML schema rules, time rules). This interface also provides a collection of rules for the additional customized step that can be performed before the result of the
validation is sent to the Authentication Core Module. The interface between
the Validation Module and the Authentication Core Module will make it pos-
sible for the Validation Module to send both the validation result (signed
with an XML signature) and the SAML request to the Authentication Core
Module for further processing.

Handling of SAML profiles, SAML deployment profiles and SAML in-
teroperability profiles is performed by the SAML Profile Module which
is shown in Figure 8.18 on page 88. The interface to the Validation Module
will make it possible for Validation Module to load the SAML rules that
are needed to perform the validation. The Configuration Module stores the
SAML rules and the interface between the SAML Profile Module and the
Configuration Module will make it possible for the SAML Profile Module to
load these rules. The interface to the Response Generator Module will make
it possible for the Response Generator Module to load the SAML rules that
are needed to generate the SAML responses. The SAML Profile Module
also has an interface to the Authentication Core Module which will make it
possible for the Authentication Core Module to load the SAML rules needed
for authentication and attribute actions that will be performed. Several sub-
modules are included in the SAML Profile Module, one is the SAML Pro-
file Processing Submodule which handles all processing of data that relates
to the basic SAML Profile definition. Deployment Profile Processing Sub-
module handles all processing of SAML Rules that is related to deployment profiles and Interoperability Processing Submodule handles all processing of SAML Rules that is related to interoperability profiles. The SAML Profile Communication Submodule handles the communication between the other modules and the submodules and also handles processing of which parts of the input that need to be processed by the different profile submodules.

The **Configuration Module** handles all configuration storage for the authentication service and is shown in Figure 8.19 on page 89. The storage in the Configuration Module is based on the Shared-Data Style (described in Section 2.4.7). Handling of the administration user interface is also performed by the Configuration Module. The administration user interface will work as a target system / relying party, which means that it will use the authentication and access control services of the security platform. To enable easy configuration management of all configuration settings in the authentication service the administration user interface will be a web interface. The administration interface will among other things make it possible for administrators to choose which attribute that will be sent to specific applications. The Validation Submodule of the Configuration Module will handle validation of incoming configuration changes from the administration user interface. The Configuration Storage Submodule handles the physical...
storage of all the configuration settings (which will be stored in a distributed database to enhance availability, redundancy and scalability). A further decomposition of the contents of the Configuration Storage Database is shown in Figure 8.20 on page 90. Configuration rules will be stored as XML to provide extensibility. A local copy of the Central Metadata Register (CMR) settings will be saved in the Configuration Storage Module to ensure high availability. The Configuration Storage Module also includes Relying Party Configuration which among other thing will include which attribute that will be sent to the specific applications.

The processing of SAML requests is handled by the Authentication Core Module which is shown in Figure 8.21 on page 91. The Authentication Core Module will decide which of the domain layer modules (Federation Module, Authentication Module, Attribute Module, Legacy Module) it will use to process the SAML request and send necessary data to them for further processing. The Session Module will be used by the Authentication Core Module to check for active sessions and to create and update sessions. When the Authentication Core has finished processing the SAML request the result will be sent to the Response Generator Module. The interface to the SAML Profile Module allows loading of the rules for the SAML profiles needed for process decision and also the rules needed in the Authentication Method Module, Federation Module, Legacy Module and Attribute Module so that these modules don’t need to have interfaces to the SAML Profile Module. The solution to load all rules to the Authentication Core Module was chosen to lower the coupling between the modules.

The Session Module handles retrieval, update, creation and deletion of sessions and is shown in Figure 8.22 on page 91. Input to the Session Module is received from the Authentication Core Module which also gets the results of the session operations. The Session Module includes an interface
8.7. DEVELOPMENT VIEW

Figure 8.21: Authentication Core Module

Figure 8.22: Session Module
CHAPTER 8. ARCHITECTURAL DESIGN

to the Session Storage which will use a distributed database to enhance availability, redundancy and scalability. Session data is for security reasons encrypted and saved on the server side and not at the user side. Different programming platforms have different ways to save session objects on servers but since this architecture provides the opportunity to use load balancers and failovers (described in Section 8.8) it could cause availability issues if session data is saved directly on each individual server, therefore, a shared distributed database will be used to store session data.

Handling of execution of different authentication methods is performed using the Authentication Method Module which is shown in Figure 8.23 on page 92. The Authentication Method Module depends on the Authentication Core Module to provide it with information about which authentication method(s) that shall be used and what input that shall be used for them. The different authentication methods are designed to be submodules and can be extended with additional authentication methods. The authentication methods are extended by receiving new authentication methods from the Configuration Module. Each time the Authentication Method Module gathers configuration settings it also checks for new authentication methods. The interface with the Configuration Module also provides possibilities to update rules for existing authentication methods. Both external and internal credential storages can be used and is configured using the Configuration Storage which provides the Authentication Method Module with information about e.g. communication type and access addresses. All credential storages
shall be encrypted for security reasons. The authentication methods that are included in the architecture as submodules are SMS-OTP, smart-card, username and password, LDAP and AD. Additional submodules for using the current standard for electronic IDs in Sweden (EID Submodule) and the new standard for electronic IDs (Eid2.0) in Sweden (EID 2.0 Submodule) are also included in the Authentication Method Module. The Authentication Core Submodule also includes a submodule called Certificate Control Submodule which handles authentication using certificates and can validate the certificates using Trusted Secure Server (TSS) (A Cybercom Product), Certificate Revocation List (CRL) which is provided by the Certificate Authority (CA) (can be saved for the time that the CRL is valid) and by OCSP requests to OCSP Responder.

Figure 8.24: Attribute Module

The Attribute Module handles the retrieval of attributes and is shown in Figure 8.24 on page 93. Attributes can be retrieved from internal or external attribute sources of various types and the retrieval of attributes is configured using the rules received from the Configuration Module. All attribute storages shall be encrypted for security reasons. Attribute requests are received from the Authentication Core which also receives the resulting attributes. The Attribute Module has a submodule that is called Attribute Request Analyzer Submodule which is responsible for analyzing the attribute request to determine which attribute source to use and if additional attribute processing is needed.
CHAPTER 8. ARCHITECTURAL DESIGN

Sometimes additional attribute processing is needed. This processing is handled by the Attribute Processing Module which is shown in Figure 8.25 on page 94. This additional processing could be e.g. personal numbers that are stored in various formats and need to be processed into a single common format or employee roles that need to be processed into profession categories. Rules for this attribute processing are received from the Configuration Module. The Attribute Processing Module has a submodule called Attribute Data Analyzer Submodule which is responsible for analyzing the input from the Attribute Module to determine what processing rules that are applicable and a submodule called Attribute Data Processing which is responsible for performing the actual attribute processing actions.

The Federation Module is responsible for handling all configuration for participation in identity federations and is shown in Figure 8.26 on page 95. The interface to the Central Metadata Register (CMR) makes it possible for the Central Metadata Handling Submodule to load metadata from the CMR (data signed by the federation operator who handles the federation) and to send updated configuration data to the CMR (information about the IdP, the target systems and the attribute services that are provided). The interface to the Authentication Core Module makes it possible for the Federation Module to receive configuration data for the identity federation that the Authentication Core Module has gathered from the Configuration Module. The results of the federation processing operations and a local copy of the CMR is sent back to the Authentication Core Module which in turn forwards it to the Configuration Module for storage to ensure high availability. The validation of the received metadata is performed in a separate submodule called Metadata Validation Submodule in order to not have to go the extra detour by the Validation Module when the metadata shall be
8.7. DEVELOPMENT VIEW

Figure 8.26: Federation Module

used and also when the local copy of the CMR shall be stored in the Configuration Module. When requests to the authentication service is initiated by a relying party and the authentication service is part of an identity federation, then the federation module is responsible for checking if the relying party is trusted according to federation rules.

Handling of authentication and attributes for legacy systems that does not have support for SAML will be performed by the Legacy Module which is shown in Figure 8.27 on page 96. The rules that configure the Legacy Module (what legacy systems to use and which communication type that is used) is received through the interface with the Communication Module and instructions for the actual operations to perform and the input to use will be received from the Authentication Core Module. The Legacy Module have interfaces to both internal and external credential storages and attribute storages that can be used to perform the operations. All credential storages and attribute storages shall be encrypted for security reasons. Processing of the legacy requests are handled by the Legacy Request Processing Submodule and the results are then sent back to the Authentication Core Module.

All logging in the authentication service will be handled by the Logging Module which is shown in Figure 8.28 on page 97. All the other modules implements an interface to the Logging Module which will make them able to send logging requests according to a specific XML format to the Logging Module. The Logging Module processes the logging requests according to
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.27: Legacy Module
8.7. DEVELOPMENT VIEW

Figure 8.28: Logging Module

rules received from the Configuration Module. The interface with the Central Logging Service will make it possible for the Logging Module to send the processed log requests in XML format (signed and encrypted) to the Central Logging Service. Connection type and rules for the Central Logging Service will be received from the Configuration Module. The Logging Module offers three types of logging: audit logging, performance logging and error logging. These different types of logging are handled by individual submodules called Audit Logging Submodule, Performance Logging Submodule and Error Logging Submodule. It is important to make sure that the logs are maintained in a trusted way. It is particularly important for the audit trails since they are often attack targets themselves [5], therefore all storage and communication-handling logs will be both signed and encrypted.

The Response Generator Module is responsible for handling generation of SAML responses and is shown in Figure 8.29 on page 98. The interface from the Authentication Core Module will make it possible for the Response Generator Module to receive the results of the authentication operations. Response generation depends on which SAML profiles that are used and the interface to SAML Profile Module will make it possible for the Response Generator Module to receive these SAML profile rules. The Input Analyzer Submodule analyzes the results received from the Authentication Core Module to decide what type of SAML response to generate and the Response Generation Submodule performs the generation. The Response Module will also be responsible for providing the SAML assertion with the LoA that was used (if applicable). Signing is performed by the Signing Submodule and here signing using a Hardware Security Module (HSM) is
 optional. The SAML response is sent to the appropriate receiver (relying party or user agent) and the results of the response generation will also be returned to the Authentication Core Module to enable session updates.

8.7.3 Rationale

A combination of different module structures (described in Section 2.2.1) have been chosen for the development view. Both decomposition style, layered style and allowed to use relations have been used to provide an as extensive description as possible. Module decomposition is based on the information hiding principle (which also is good for security purposes) where parts of the system that typically will be independent are collected and encapsulated into individual modules, which is good for testability and to make it possible to change the implementations of one module without affecting the other modules. To make the authentication service as conformative as possible the approach to divide the development view into modules and submodules seemed to be the best suitable approach. Using a layered style was chosen to enhance the security awareness among the authentication service stakeholders. The layered style was also chosen to separate the different domains of the authentication service from each other to provide lower coupling between the modules. Layered approaches are common in security architectures and also enhances the modifiability [41], therefore, the layered style was used as a base for the development view. Audit logging is
8.8 PHYSICAL VIEW

provided by the Logging Module to enhance the security quality attribute of the authentication service and is used both as an prevention technique (causing fear of punishment) [5] and to maintain an audit trail to identify an attacker. The reason for choosing each module to have an interface to the logging module (instead of handling the logging from only Request and User Interface Module, Validation Module, Authentication Core Module and Configuration Module) is to be able to provide as accurate and specific log requests as possible.

8.8 Physical View

The physical view shows the mapping of the software to the underlying hardware and takes into account some non-functional requirements like availability, reliability and scalability.

8.8.1 Primary Presentation

The primary presentation of the physical view gives a brief description of how the hardware environment where the authentication service executes shall be structured and is shown in Figure 8.30 on page 100. The figure gives an overview of how the hardware topology is structured and the communication between the various parts of the hardware environment.

8.8.2 Element Catalog and Context Description

The physical view is divided into 5 main areas: Client Communication & Firewall, Load Balancer & Failover, Authentication Service Replication, Internal Storage Replication and External Services. Each of these areas describes important aspects of the physical environment of the authentication service and they all have important obligations.

The Client Communication and Firewall area includes transport layer structure for both relying parties and user agents. Communication with the authentication service can be performed using HTTP or using SOAP. To provide message integrity and message confidentiality all communication with the authentication services shall be performed using SSL/TLS. To be able to control which incoming and outgoing data traffic that shall be allowed, a firewall shall be used. The firewall shall work as a network bridge to enhance security by limiting access [5] between the external and the internal network.

To ensure high availability and high scalability, the Load Balancer and Failover area includes support for both load balancing and failover. Failover is provided by using duplicate servers which are communicating heartbeat messages to each other at regular intervals. If the failover stops receiving heartbeats it makes sure to handle all communication from the clients. Load
balancing is provided by spreading the different request from clients onto several replicated authentication service systems.

The **Authentication Service Replication** area provides replicated copies of the authentication service system, which means that if one of them stops working all the others will still be available to serve client requests. Also if one of the authentication service system copies gets high load, then the load balancer can send the new requests to another copy of the authentication service.

The various data storages within the authentication service (e.g. configuration, credential, attributes and session) shall provide replication by using
8.9. SCENARIOS

distributed databases. In the primary presentation this is represented by the Internal Storage Replication area. Distributed databases shall be used since it provides a way to keep data consistent between the different authentication service system copies and also provides higher availability and scalability for the data that the authentication service systems are relying on.

The External Services area represents the various external services that the authentication service systems are using to perform authentication actions (e.g. external credential storage, external attribute storages, Central Metadata Registers, TSS, OCSP responders and central logging service).

8.8.3 Rationale

This informal representation of the physical view have been chosen over a more formal representation using e.g. UML to be able to quickly provide the reader with a clear picture of the architecture’s physical appearance. This representation style was also chosen to avoid the physical view being too messy. The load balancer part of the physical view shall implement the distributor pattern (described in Section 3.4.2) which is a security architecture pattern used to separate concerns.

8.9 Scenarios

The scenarios are the view that tie the other four views together and are instances of more general use cases.

8.9.1 Primary Presentation

The primary presentation of the scenarios are divided into two scenarios: Scenario 1 shown in Figure 8.31 on page 102 and Scenario 2 shown in Figure 8.32 on page 103.

8.9.2 Element Catalog and Context Description

Scenario 1 (authentication and additional attributes) which is shown in Figure 8.31 on page 102 consists of the following steps:

1. The Request and User Interface Service detects an incoming SAML authentication request trough a direct request and sends it to the Validation service.
2. The Validation Service sends a request to the Configuration Service to gather new update validation configurations.
3. The Configuration Service transmits the updated validation configuration to the Validation Service.
CHAPTER 8. ARCHITECTURAL DESIGN

Figure 8.31: Scenario 1

4. The Validation Service performs validation of the SAML authentication request and transmits the authentication request to the Authentication Core Service.

5. The Authentication Core Service sends a request to the Configuration Service to gather new updated authentication core configurations.

6. The Configuration Service transmits the updated authentication core configuration to the Authentication Core Service.

7. The Authentication Core Service processes the Authentication Request to determine session status, what authentication method and additional services that are needed and first transmits an authentication method request to the Authentication Method Service.

8. The Authentication Method Service sends a request to the Configuration Service to gather new updated authentication method configurations.

9. The Configuration Service transmits the updated authentication method configuration to the Authentication Method Service.

10. The Authentication Method Service performs the authentication actions and transmits the results to the Authentication Core Service.
8.9. SCENARIOS

11. The Authentication Core Service has detected that additional attributes are needed and sends an attribute request to the Attribute Service.

12. The Attribute Service sends a request to the Configuration Service to gather new updated attribute configurations.

13. The Configuration Service transmits the updated attribute configuration to the Attribute Service.

14. The Attribute Service performs the attribute actions and transmits the results to the Authentication Core Service.

15. The Authentication Core Service puts together the authentication method result and the attribute result and sends a request for response generation to the Response Generation Service. Response Generation Service generates and sends the appropriate SAML response to the receiver.

Scenario 2 (part of a large identity federation + authentication) which is shown in Figure 8.32 on page 103 consists of the following steps:

1. The Request and User Interface Service receives an incoming SAML authentication request through the web interface and sends it to the Validation Service.

2. The Validation Service sends a request to the Configuration Service to gather new updated validation configurations.

Figure 8.32: Scenario 2

1. The Request and User Interface Service receives an incoming SAML authentication request through the web interface and sends it to the Validation Service.

2. The Validation Service sends a request to the Configuration Service to gather new updated validation configurations.
3. The Configuration Service transmits the updated validation configuration to the Validation Service.

4. The Validation Service performs validation of the SAML authentication request and transmits the authentication request to the Authentication Core Service.

5. The Authentication Core Service sends a request to the Configuration Service to gather new updated authentication core configurations.

6. The Configuration Service transmits the updated authentication core configuration to the Authentication Core Service.

7. The Authentication Core Service processes the authentication request and identifies that the authentication service is a part of a identity federation and therefore sends a federation request to the Federation Service.

8. The Federation Service updates data from a Central Metadata Register (CMR) and returns the federation configurations that are needed by the Authentication Core Service.

9. The Authentication Core Service further processes the Authentication Request and identifies that authentication need to be performed by the Authentication Method Service and therefore sends a request to the Authentication Method Service.

10. The Authentication Method Service sends a request to the Configuration Service to gather new updated authentication method configurations.

11. The Configuration Service transmits the updated authentication method configuration to the Authentication Method Service.

12. The Authentication Method Service performs the authentication actions and transmits the results to the Authentication Core.

13. The Authentication Core Service sends the authentication results to the Response Generation Service which generates and sends the appropriate SAML response the receiver.

8.9.3 Rationale

As recommended for the 4+1 View Model of Software Architecture (described in Section 2.6.3) the Logical View has been used as a base for the scenarios. The various logging requests to the Logging Service have been omitted from the scenarios since they would probably only make it more complicated to grasp the structure of the scenarios. Other details have also been deliberately omitted to avoid making the scenarios unnecessary complicated.
8.10 Discussion

As shown in this chapter, this architectural design for the authentication service is conformative in various aspects, e.g. usage of different authentication methods, usage of different SAML profiles, possibilities to communicate with legacy systems not supporting SAML, possibilities to participate in various identity federations and possibilities to use an administration interface which makes the authentication service highly configurable.

Regarding the issue of providing versatile authentication for the authentication service, no obstacles with combining various authentication methods could be found. Neither the literature studies, the requirements analysis or the architectural design gave any indications that combining various authentication methods could pose any problems. This applies if it is possible to configure which authentication methods that are allowed to be used, and by that be able to provide a lowest security level. For authentication services where high security is important multiple-factor authentication (see Chapter 5) is recommended.

When it comes to using legacy systems that does not support SAML, this is not recommended since it could pose security issues and also increases the complexity of the authentication service to an unnecessarily high level. Since using legacy systems was a part of the requirements for this particular authentication service it was included in the architectural design. However, if this had not been a requirement then it would have been avoided.
Chapter 9

Evaluation

This chapter describes the evaluation performed to compare the current active architecture with the candidate architecture presented in this thesis.

9.1 Method and Rationale

Scenario-based Architecture Analysis Method (SAAM) (described in section 2.7.3) was chosen for the evaluation of the architectural design. SAAM is a qualitative scenario-based evaluation method and it was chosen since it is a widely used and validated method which was developed to compare candidate architectures.

An alternative method that also was considered was the Architecture Tradeoff Analysis Method (ATAM) (described in section 2.7.2) which has evolved from SAAM, however ATAM is a much more complex method compared to SAAM. ATAM is more complex compared to SAAM since it introduces more steps in the evaluation process and also extends the other steps which makes the method more time consuming. Since the timeframe of the evaluation in this thesis is limited, the size of a SAAM evaluation was considered more suitable to evaluate the architectural design and compare it to the current active architectural design. Another reason for choosing SAAM over ATAM is that SAAM was developed to help compare candidate architectures while ATAM has the focus on comparing competing quality attributes.

Another alternative method that also was considered was the Architecture-Level Modifiability Analysis (ALMA) (described in section 2.7.1). The ALMA method was not chosen to evaluate the architectural design and compare it to the current active architectural design since it is a method that only focuses on the modifiability quality attribute. SAAM focuses on various quality attributes and to get a more comprehensive evaluation SAAM was considered to be a more suitable method compared to ALMA.
9.2 Prerequisites

In addition, Scenario-Based Architecture Reengineering (SBAR) (described in section 2.7.4) was also considered for the evaluation. SBAR focuses on reengineering of a software architecture where major changes in the requirements often is the starting point. The SBAR method takes updated requirements document and the existing architecture as inputs and produces an improved architecture as output. Since the goal of this evaluation was to compare the current active architecture with the candidate architecture presented in this thesis the SAAM method was considered to be more suitable compared to SBAR.

The evaluation was performed according to the five steps of the SAAM process (described in Section 2.7.3), however due to the limited timeframe, the fourth step (Reveal Scenario Interaction) was omitted. The evaluation was conducted together with several Cybercom employees who have been working with the authentication service and the security platform. Unfortunately it was not possible to include any customers or other stakeholders in the evaluation and this could affect the result.

9.3 Results

This section is structured according to the steps of the SAAM process (described in Section 2.7.3) and describes the results from the execution of each step. The fourth step (Reveal Scenario Interaction) was omitted in this evaluation due to the limited timeframe.

9.3.1 Describe Candidate Architectures

Both the architectural design of the current active architecture and the candidate architecture presented in this thesis were presented to all the
CHAPTER 9. EVALUATION

participants of the evaluation in a similar manner.

9.3.2 Develop Scenarios

A list of scenarios which are activities that the authentication service must support, as well as future changes that are likely to be needed were elicited. The requirements lists from Chapter 7 and the following prioritized quality attributes: 1. Security, 2. Availability, 3. Performance, 4. Redundancy, 5. Scalability, were used as a foundation for the elicitation. The resulting scenarios are presented in Table 9.1.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accept SAML HTTP requests.</td>
</tr>
<tr>
<td>2</td>
<td>Accept SAML SOAP requests.</td>
</tr>
<tr>
<td>3</td>
<td>Choose authentication method using web interface.</td>
</tr>
<tr>
<td>4</td>
<td>Present customized user interface.</td>
</tr>
<tr>
<td>5</td>
<td>Multiple-factor authentication.</td>
</tr>
<tr>
<td>6</td>
<td>Request and receive user attributes.</td>
</tr>
<tr>
<td>7</td>
<td>Generate SAML protocol messages.</td>
</tr>
<tr>
<td>8</td>
<td>Sign in using Single Sign-On.</td>
</tr>
<tr>
<td>9</td>
<td>Logout using Single Logout.</td>
</tr>
<tr>
<td>10</td>
<td>SSO over organizational boundaries.</td>
</tr>
<tr>
<td>11</td>
<td>Participate in an identity federation.</td>
</tr>
<tr>
<td>12</td>
<td>Receive federation data from central metadata register.</td>
</tr>
<tr>
<td>13</td>
<td>Support the EID 2.0 infrastructure.</td>
</tr>
<tr>
<td>14</td>
<td>Validate SAML protocol messages.</td>
</tr>
<tr>
<td>15</td>
<td>Validate SAML protocol messages according to different SAML profiles.</td>
</tr>
<tr>
<td>16</td>
<td>Check certificates using CRL and OCSP.</td>
</tr>
<tr>
<td>17</td>
<td>Validate allowed authentication methods by relying party.</td>
</tr>
<tr>
<td>18</td>
<td>Perform attribute processing.</td>
</tr>
<tr>
<td>19</td>
<td>Retrieval of attributes from certificate.</td>
</tr>
<tr>
<td>20</td>
<td>Authentication using the current standard for electronic IDs in Sweden.</td>
</tr>
<tr>
<td>21</td>
<td>Perform an additional step after the user credentials are received and before the actual authentication takes place.</td>
</tr>
<tr>
<td>22</td>
<td>Perform login timeouts to discourage brute force attacks.</td>
</tr>
<tr>
<td>23</td>
<td>Perform actions when number of failed login attempts occur.</td>
</tr>
<tr>
<td>24</td>
<td>Provide information about which LoA that is used in SAML responses.</td>
</tr>
<tr>
<td>25</td>
<td>Authenticate using legacy system.</td>
</tr>
<tr>
<td>26</td>
<td>SSO for legacy systems.</td>
</tr>
</tbody>
</table>

Continued on next page
9.3. RESULTS

Table 9.1 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Single logout for legacy systems.</td>
</tr>
<tr>
<td>28</td>
<td>Error logging.</td>
</tr>
<tr>
<td>29</td>
<td>Audit logging.</td>
</tr>
<tr>
<td>30</td>
<td>Performance logging</td>
</tr>
<tr>
<td>31</td>
<td>Save logs when central logging service is available.</td>
</tr>
<tr>
<td>32</td>
<td>Send logs to central logging service.</td>
</tr>
<tr>
<td>33</td>
<td>Change configuration using administration interface.</td>
</tr>
<tr>
<td>34</td>
<td>Change attribute sources using administration interface.</td>
</tr>
<tr>
<td>35</td>
<td>Change SAML profile settings through administration interface.</td>
</tr>
<tr>
<td>36</td>
<td>Change configuration without restarting the authentication service.</td>
</tr>
<tr>
<td>37</td>
<td>Configure the authentication service using SAML metadata.</td>
</tr>
<tr>
<td>38</td>
<td>Change attribute configuration using attribute-profiles.</td>
</tr>
<tr>
<td>39</td>
<td>Change configuration of allowed authentication methods for each relying party.</td>
</tr>
<tr>
<td>40</td>
<td>Use multiple authentication service servers for load balancing.</td>
</tr>
<tr>
<td>41</td>
<td>Provide a failover server.</td>
</tr>
<tr>
<td>42</td>
<td>Serve users even if one server stops working.</td>
</tr>
</tbody>
</table>

Table 9.1: List of Scenarios

9.3.3 Perform Scenario Evaluations

The scenario evaluations were performed in three steps, identification of scenario type (direct/indirect), identification of which changes that are needed in the architecture to support the scenarios and an estimation of the effort needed to include the changes. The scenario evaluation was performed for both the current active architecture and the candidate architecture presented in this thesis. The results of the scenario evaluation of the current active architecture is presented in Table 9.2.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accept SAML HTTP requests.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Accept SAML SOAP requests.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### Table 9.2 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Choose authentication method using web interface.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Present customized user interface.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Multiple-factor authentication.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Request and receive user attributes.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Generate SAML protocol messages.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Sign in using Single Sign-On.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Logout using Single Logout.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SSO over organizational boundaries.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Participate in an identity federation.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Receive federation data from central metadata register.</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>
### 9.3. RESULTS

#### Table 9.2 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Support the EID 2.0 infrastructure.</td>
<td>Indirect</td>
<td>It is as of today hard to determine the changes needed to support this since the EID 2.0 infrastructure is not yet in place. The issues known today are metadata compliance and LoA support.</td>
<td>Medium</td>
</tr>
<tr>
<td>14</td>
<td>Validate SAML protocol messages.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Validate SAML protocol messages according to different SAML profiles.</td>
<td>Indirect</td>
<td>Today, only Web SSO Profile is supported. Add support for profile validation. Add support for configuring the authentication service according to different SAML profiles.</td>
<td>Large</td>
</tr>
<tr>
<td>16</td>
<td>Check certificates using CRL and OCSP.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Validate allowed authentication methods by relying party.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### Table 9.2 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Perform attribute processing.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Retrieval of attributes from certificate.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Authentication using the current standard for electronic IDs in Sweden.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Perform an additional step after the user credentials are received and before the actual authentication takes place.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Perform login timeouts to discourage brute force attacks.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Perform actions when number of failed login attempts occur.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Provide information about which LoA that is used in SAML responses.</td>
<td>Indirect, Add mapping of authentication method to LoA. Add inclusion of LoA in SAML assertion.</td>
<td>Small</td>
</tr>
<tr>
<td>25</td>
<td>Authenticate using legacy system.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>SSO for legacy systems.</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
## 9.3. RESULTS

Table 9.2 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>Single logout for legacy systems.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Error logging.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Audit logging.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Performance logging.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Save logs when central logging service is unavailable.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Send logs to central logging service.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Change configuration using administration interface.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Change attribute sources using administration interface.</td>
<td>Indirect</td>
<td>Move administration from configuration file to administration interface.</td>
<td>Large</td>
</tr>
<tr>
<td>35</td>
<td>Change SAML profile settings through administration interface.</td>
<td>Indirect</td>
<td>Add profile settings to administration interface.</td>
<td>Small</td>
</tr>
<tr>
<td>36</td>
<td>Change configuration without restarting the authentica-</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Configure the authentication service using SAML metadata.</td>
<td></td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
CHAPTER 9. EVALUATION

Table 9.2 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>Change attribute configuration using attribute-profiles.</td>
<td>Indirect</td>
<td>Move administration from configuration file to administration interface.</td>
<td>Medium</td>
</tr>
<tr>
<td>39</td>
<td>Change configuration of allowed authentication methods for each relying party.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Use multiple authentication service servers for load balancing.</td>
<td>Indirect</td>
<td>Add support for session sharing.</td>
<td>Large</td>
</tr>
<tr>
<td>41</td>
<td>Provide a failover server.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Serve users even if one server stops working.</td>
<td>Indirect</td>
<td>Add support for session sharing.</td>
<td>Large</td>
</tr>
</tbody>
</table>

Table 9.2: Scenario Evaluation for Current Active Architecture

The results of the scenario evaluation for the candidate architecture presented in this thesis is shown in Table 9.3.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Accept SAML HTTP requests.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Accept SAML SOAP requests.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Choose authentication method using web interface.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
9.3. RESULTS

Table 9.3 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Present customized user interface.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Multiple-factor authentication.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Request and receive user attributes.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Generate SAML protocol messages.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Sign in using Single Sign-On.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Logout using Single Logout.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SSO over organizational boundaries.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Participate in an identity federation.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Receive federation data from central metadata register.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Support the EID 2.0 infrastructure.</td>
<td>Direct</td>
<td>It is as of today hard to determine the changes needed to support this since the EID 2.0 infrastructure is not yet in place. No issues known.</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Validate SAML protocol messages.</td>
<td>Direct</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Table 9.3 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Validate SAML protocol messages according to different SAML profiles.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Check certificates using CRL and OCSP.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Validate allowed authentication methods by relying party.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Perform attribute processing.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Retrieval of attributes from certificate.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Authentication using the current standard for electronic IDs in Sweden.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Perform an additional step after the user credentials are received and before the actual authentication takes place.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Perform login timeouts to discourage brute force attacks.</td>
<td>Indirect</td>
<td>Add login time check for sessions</td>
<td>Small</td>
</tr>
</tbody>
</table>

Continued on next page.
### Table 9.3 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>Perform actions when number of failed login attempts occur.</td>
<td>Indirect</td>
<td>Add failed login count for sessions. Small</td>
</tr>
<tr>
<td>24</td>
<td>Provide information about which LoA that is used in SAML responses.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Authenticate using legacy system.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>SSO for legacy systems.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Single logout for legacy systems.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Error logging.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Audit logging.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Performance logging.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Save logs when central logging service is unavailable.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Send logs to central logging service.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Change configuration using administration interface.</td>
<td>Direct</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Change attribute sources using administration interface.</td>
<td>Direct</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### CHAPTER 9. EVALUATION

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
<th>Scenario Type</th>
<th>Changes</th>
<th>Effort (estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>Change SAML profile settings through administration interface.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Change configuration without restarting the authentication service.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>Configure the authentication service using SAML metadata.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>Change attribute configuration using attribute-profiles.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>Change configuration of allowed authentication methods for each relying party.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Use multiple authentication service servers for load balancing.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>Provide a failover server.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Serve users even if one server stops working.</td>
<td>Direct</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 9.3: Scenario Evaluation for Candidate Architecture
9.3. RESULTS

9.3.4 Overall Evaluation

As a final step of the evaluation all scenarios were weighted by the stakeholders involved in the evaluation to produce an overall ranking. The scenarios were weighted with votes which reflects their relative importance. The more votes a scenario have, the more important it is to the stakeholders. The overall evaluation is summarized in Table 9.4 where the table is sorted by the number of votes that the scenarios received. The table includes a final comparison between the current active architecture and the candidate architecture, where they were weighted according to the following notation: + = better, - = worse and 0 = no difference.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Votes</th>
<th>Current Active Architecture</th>
<th>Candidate Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>49</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>48</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>47</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>47</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>41</td>
<td>45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>44</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>43</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>43</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>42</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>42</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>41</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>41</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>41</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>19</td>
<td>41</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>41</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>39</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>39</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>38</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>31</td>
<td>38</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>38</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>37</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
CHAPTER 9. EVALUATION

Table 9.4 – continued from previous page

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Votes</th>
<th>Current Active Architecture</th>
<th>Candidate Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>31</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>30</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>24</td>
<td>30</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>26</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>30</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>29</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>29</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>38</td>
<td>28</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>21</td>
<td>27</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>25</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

Table 9.4: Evaluation Summary Table

To summarize the final comparison, eight scenarios were considered better for the candidate architecture and two scenarios were considered better for the current active architecture (shown in Table 9.5).

<table>
<thead>
<tr>
<th>Scenarios Better For</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current active architecture</td>
<td>2</td>
</tr>
<tr>
<td>Candidate architecture</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 9.5: Evaluation Results Table

The two indirect scenarios for the candidate architecture only need minor changes and the estimations of effort is small. The eight indirect scenarios for the current active architecture needs changes of varying amounts. The effort estimations ranges from small to large.

Worth noting is also that the two indirect scenarios for the candidate architecture (scenario 22 and scenario 23) are covered by the requirements list in Section 7.3 (requirement R1.10 and requirement R1.11). However these requirements were missed during the architectural design. The architectural design could with small efforts have been changed to cover these requirements but it was left unchanged to demonstrate that the evaluation gives results and helps to discover deficiencies in the architecture.
Chapter 10

Conclusions and Future Work

This chapter summarizes the conclusions from this thesis and discusses possible future work.

10.1 Conclusions

The contribution of this thesis is an architectural design of a conformative authentication service. This includes requirements analysis, architectural design and evaluation. The problem formulation in Chapter 1 stated the following problems:

- How can an architectural design of a conformative authentication service be developed based on a given set of requirements?
- How can versatile authentication be enabled in the same architectural design?
- Are there any problems with combining different authentication methods in the same authentication service?
- How can software architectures be evaluated?

Architectural designs are often described using views; having multiple views helps stakeholders to handle complexity by separating various concerns. For the architectural design of the authentication service, Kruchten’s 4+1 View Model [32] was chosen. Kruchten’s 4+1 View Model consists of a logical view, a process view, a development view, a physical view and scenarios. Various architectural styles and secure design patterns were used to satisfy quality attributes and requirements. Some of the architectural styles and secure design patterns used in the architectural design presented
in this thesis are: Communicating Processes, Decomposition Style, Information Hiding and Encapsulation, Layered Style, Shared-Data Style, Distributor, Interceptor and Trusted Third Party. The architectural design of the authentication service is conformative in various aspects, e.g. usage of various authentication methods, usage of various SAML profiles, usage of legacy systems not supporting SAML, possibilities to participate in various identity federations and possibilities to use an administration interface which makes the authentication service highly configurable.

Versatile authentication is enabled in the architectural design by providing possibilities to use various authentication methods. The architectural design provides various authentication methods by placing them as separate submodules in an authentication method module. In addition, the architectural design also enables additional authentication methods to be added and allows authentication methods to be configured.

During this thesis work no indications of any problems with combining various authentication methods were found. However, possibilities to configure which authentication methods that are allowed are considered an important security aspect.

Evaluation of software architectures can be categorized into two types: qualitative and quantitative evaluations. A common type of qualitative evaluations are scenario-based evaluations which are simple, flexible and enable systematic investigation of software architectures. In this thesis Software Architecture Analysis Method (SAAM) was used to compare the architectural design presented in this thesis with the current active architectural design. The evaluation concludes that the architectural design presented in this thesis was considered better for more scenarios compared to the current active architectural design. In addition, the evaluation also discovered some minor deficiencies in the architectural design presented in this thesis, which demonstrates that architectural evaluations are valuable for software architects.

10.2 Future Work

During the work with this thesis some ideas of possible further work have emerged:

- During the requirements analysis, one of the elicited requirements (R1.26) is about supporting the new standard for electronic IDs (Eid2.0) in Sweden. When this thesis was written the Eid2.0 infrastructure was not yet in place, therefore investigation of compliance with this standard is proposed as future work.

- Possibilities to add additional authentication methods are included in the architectural design presented in this thesis. However, details about how the administration interface shall be used to add new
10.2. FUTURE WORK

authentication methods are not described. Therefore, investigations about exactly how this functionality shall be performed is proposed as future work.

- The architectural design presented in this thesis contains a validation module which includes validation of SAML 2.0 profiles. A proposed further work related to this validation is to investigate exactly how the validation shall be performed and more specifically what needs to be evaluated for each SAML 2.0 profile.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


127
BIBLIOGRAPHY


Authentication services in security platforms often need to handle different types of systems which have various requirements regarding the authentication. These requirements can often interfere with each other and the issue here is that the authentication service often needs to be manually adjusted to comply with these requirements. Therefore there is a need for a flexible architectural design which enables changes and could open up for new emerging technologies and possibilities. This thesis presents an architectural design of a conformative authentication service based on SAML 2.0 to be used in security platforms. In this thesis a requirements analysis was performed and an architectural design was developed. The architectural design presented in this thesis is conformative in various aspects, e.g. usage of various authentication methods, versatile handling of attributes, handling of various SAML 2.0 profiles, possibilities to participate in various identity federations and handling of legacy systems not supporting SAML. In addition, an evaluation comparing the candidate architectural design presented in this thesis with a currently active architectural design was performed. This evaluation showed that the candidate architectural design was considered better for more usage scenarios.
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnt som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Mikael Hermansson