Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

Mika Suvanto

A Monitoring System for Authentication and Authorisation Infrastructure

Master's Thesis

Espoo, April 26, 2010

Supervisor: Professor Tuomas Aura

Instructor: Dr. Mikael Linden

Aalto University
School of Science and Technology
Faculty of Information and Natural Sciences
Degree Programme of Computer Science and Engineering

ABSTRACT OF THE MASTER'S THESIS

Author:	Mika Suvanto		
Name of the thesis:	A Monitoring System for Authentication and Authorisation Infrastructure		
Date:	April 26, 2010	**Number of pages:**	54 + 1
Language:	English		
Professorship:	Data Communications Software	**Code:**	T-110
Supervisor:	Prof. Tuomas Aura		
Instructor:	Dr. Mikael Linden		

Federated authentication and authorisation infrastructure enables Single Sign On with the user's home organisation account to services provided by other organisations. Finnish universities, polytechnics and CSC – IT Center for Science have established the Haka federation, which is used for user authentication in many WWW services offered for students and faculty. The availability and reliability of authentication and authorisation infrastructure is critical to the services that rely on it.

This thesis introduces the problem of monitoring an authentication and authorisation infrastructure. A monitoring application called AAIEye is introduced. The application is used for monitoring availability and usage statistics of the authentication and authorisation infrastructure. It is available as open source and can be used for monitoring other federations that use the same kind of technology as Haka, i.e. SAML 2.0 or Shibboleth.

The distributed nature of a federation makes it demanding to operate it and diagnosing problems can be difficult between different organisations. Monitoring the service availability is hard, because monitoring a single component does not say much about the availability of the federation. In this thesis, an end user approach is used to address this problem, and availability monitoring is implemented by simulating a WWW browser.

Another challenge for a federation is collecting the usage statistics. The services of the federation create logs, but getting the whole picture of how much the services are used would help when planning new services or federation operator's work. For this purpose, a service for collecting and presenting the usage statistics of a federation is introduced.

Keywords: monitoring, Haka, federation, authentication, authorisation, SAML

Aalto-yliopisto
Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietotekniikan tutkinto-/koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Mika Suvanto		
Työn nimi:	Luottamusverkoston valvontajärjestelmä		
Päivämäärä:	26.4.2010	**Sivuja:**	54 + 1
Kieli:	Englanti		
Professuuri:	Tietoliikenneohjelmistot	**Professuurikoodi:**	T-110
Työn valvoja:	Prof. Tuomas Aura		
Työn ohjaaja:	Tekn. tri Mikael Linden		

Luottamusverkosto mahdollistaa käyttäjille kertakirjautumisen toisen organisaation palveluihin oman organisaationsa käyttäjätunnuksella. Suomen korkeakoulut yhdessä tieteen tietotekniikan keskus CSC:n kanssa muodostavat Haka-luottamusverkoston, jota käytetään laajalti opiskelijoille ja henkilökunnalle tarjottavissa WWW-pohjaisissa palveluissa käyttäjien autentikointiin. Luottamusverkoston luotettava toiminta ja saatavuus ovat olennaisia siitä riippuville palveluille.

Tässä työssä tutustutaan luottamusverkoston valvonnan erityispiirteisiin ja haasteisiin. Työssä esitellään valvontaa varten kehitetty AAIEye-ohjelmisto, jota käytetään Haka-luottamusverkoston valvontaan ja käyttötilastointiin. Ohjelmisto on julkaistu avoimen lähdekoodin periaatteella ja sovellettavissa myös muiden SAML 2.0 tai Shibboleth -tekniikkaa käytettävien luottamusverkostojen valvontaan.

Luottamusverkoston hajautettu rakenne tuo sen ylläpitoon haasteensa, ja vikaselvittely monen organisaation välisissä palveluissa voi olla työlästä. Myös palveluiden saatavuuden valvonta on vaikeaa, koska yhden komponentin valvonta ei vielä kerro koko luottamusverkoston toiminnasta. Tämän vuoksi työssä kehitetyssä ratkaisussa palveluiden saatavuuden valvonta on toteutettu loppukäyttäjän näkökulmasta WWW-selainta simuloimalla.

Hajautetussa palvelussa haasteena on myös käyttötilastojen kerääminen. Luottamusverkoston palvelut muodostavat kukin oman lokinsa, mutta käytön kokonaiskuvan saaminen auttaa luottamusverkoston operoinnin ja uusien palveluiden suunnittelussa. Työssä esitellään ratkaisu käyttötilastojen keskitetystä tallennus- ja esityspalvelusta.

Avainsanat: valvonta, Haka, luottamusverkosto, autentikointi, auktorisointi, SAML

Acknowledgements

This thesis has been written along my work at CSC – IT Center for Science, and I want to thank my employer for providing me such interesting topic and possibility to do this work. My colleagues at CSC have given their impact on this work as the ideas were discussed during the project, and I want to thank them for it. I have learned a lot during this project.

I want to thank Professor Tuomas Aura for supervising this thesis, Professor Sasu Tarkoma and my instructor, Dr. Mikael Linden for their comments and feedback. Also I would like to thank Dr. Jürgen Rauschenbach and other members of Geant2 JRA5 project. Their interest and feedback during the project has given me a lot of ideas to think about.

Finally, a big thanks goes to my friends and family for their support. It has been a long way.

Tapiola, April 01, 2010

Mika Suvanto

Contents

Abbreviations **viii**

1 Introduction **1**

1.1 Background 1
1.2 Problem Statement 3
1.3 Evaluation Criteria 3
1.4 Scope 4
1.5 Structure 4

2 Authentication and Authorisation **5**

2.1 Authentication 5
2.2 Authorisation 6
2.3 Authentication and Authorisation Infrastructure 7
2.4 Federated Identity 7
2.4.1 Attributes 9
2.4.2 Managing Trust 10
2.4.3 Single Sign On and Single Logout 10
2.5 Security Assertions Markup Language (SAML) 12
2.5.1 Liberty Alliance and WS-* Protocols 13
2.5.2 SAML protocol versions 1.1 and 2.0 14
2.5.3 Profiles 14
2.5.4 XML Digital Signatures 15
2.5.5 SAML Metadata 15
2.5.6 Identity Provider Discovery 16
2.6 Shibboleth 17

2.6.1 Identity Provider 19
2.6.2 Service Provider 20
2.7 Haka federation 20
2.8 The eduroam 21

3 Beyond federations **24**
3.1 Building interoperability between federations 24
3.1.1 Technical and practical challenges 25
3.2 eduGAIN 26
3.3 Kalmar Union 28
3.4 Monitoring a Confederation 28

4 Requirements for AAI Monitoring System **30**
4.1 Motivation 30
4.2 Requirements Specification 30
4.2.1 Interfaces 31
4.2.2 Performance 32
4.3 Privacy and Security 32
4.4 Related Work 33
4.4.1 EDDY 33
4.4.2 eduroam monitoring 34
4.4.3 Nagios 34
4.4.4 Summary 34

5 Design and Implementation **36**
5.1 Design Goals 36
5.2 Architecture 36
5.2.1 Monitoring Server 37
5.2.2 Probes 40
5.2.3 Communication between Probes and the Server 43
5.2.4 User Interfaces 44
5.3 Deployment 47

6 Results **48**
6.1 Status of the Work 48

6.1.1 Functionality 50
6.1.2 Security and Privacy 50
6.1.3 Performance 51
6.2 Future Work 51

7 Conclusions **53**

A List of Illustrations **55**

Bibliography **56**

Abbreviations

AAI	Authentication and Authorisation Infrastructure
ADFS	Active Directory Federation Services
BE	Bridging Element
CA	Certificate Authority
DS	Discovery Service
DSA	Digital Signature Algorithm
FPP	Federation Peering Point
GPL	GNU Public License
GUI	Graphical User Interface
HTML	Hypertext Markup Language
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IdP	Identity Provider
IEEE	Institute of Electrical and Electronics Engineers
IIS	Internet Information Services
IP	Internet Protocol
MDS	Metadata Service

OASIS	Organization for the Advancement of Structured Information Standards
OID	Object Identifier
PKI	Public Key Infrastructure
RADIUS	Remote Authentication Dial In User Service
RSA	Rivest, Shamir and Adleman algorithm
SAML	Security Assertions Markup Language
SCHAC	Schema for Academica
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
SSO	Single Sign On
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTF	Unicode Transformation Format
WAYF	Where Are You From
WFAYF	Which Federation Are You From
WLAN	Wireless Local Area Network
WS	Web Services
WWW	World Wide Web
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Background

Wide range of applications today share a common need for *authentication* – that is, determining whether someone or something is, in fact, who or what it is declared to be. Closely related to the problem of authentication is *authorisation* – determining whether someone or something has a permission to do something. For example, an ATM requires that the user authenticates herself using her credit card and a PIN code – a successful authentication authorises her to withdraw money from her bank account. The problem is common for many different applications within organisation and also inter-organisationally – the organisation barriers have to be crossed today in many ways.

Inter-organisational authentication and authorisation has been a problem of many applications which have to identify their users reliably. There is a need for secure user authentication and an up-to-date database of users personal data, which can be used in authorisation decisions. These tasks are common for a wide range of applications. Currently, the authentication is commonly done with a separate username and password to each service accessed. The users have several usernames and passwords to remember, and this causes more administrative work on user management and possibly weakened security as the password quality tends to decrease when there are many separate passwords needed. Authorisation is also problematic as the up-to-date user data has to be stored in every service, but many applications have little means to make sure that this data is updated.

An *Authentication and Authorisation Infrastructure*(AAI) offers a solution to many

of these requirements. The idea is to provide a single digital identity and to store the identity data about the user in one location – which is then handled by an *Identity Provider* (IdP). Identity Provider is responsible for authenticating the users and providing the identity data, which can be queried by the services. The services must then rely on the information provided by the Identity Provider, meaning that there must be trust between the services offering authentication and the services using that information. This trust network is called a federation.

This thesis concentrates on AAI solutions for web applications. We will introduce the Security Assertions Markup Language (SAML) which has become a popular solution in building AAI between organisations. We will also have a look on Shibboleth software developed by Internet2 as an implementation of SAML protocol and see how it is used.

In higher education there are many applications that can benefit from AAI, library and e-learning systems like Moodle as an example. In 2002, CSC – IT Center for Science and several Finnish universities started a project to support the development of user management processes. This project, Haka, resulted in establishing a Finnish higher education federation which also got the name Haka. The technical solution for Haka federation was Shibboleth. Haka has been in production use in Finnish higher education since August 2005. It is currently used in most of the Finnish universities and in many polytechnics, and in various service providers such as library portals, including service providers outside Finnish borders. Shibboleth is being used in several other countries as well, like France, Switzerland and the United States. Some work is also being done to enable co-operation of the national federations. The Géant2/Joint Research Activity 5 (GN2/JRA5) project tries to establish an European confederation using eduGAIN AAI, a software currently being developed across the Europe aiming to enable interoperability between different AAIs. Other work in this are is Kalmar Union which started in September 2009. Kalmar Union connects educational federations of the Nordic countries, including Finland, Sweden, Norway, Denmark and Iceland.

As the federations are gaining more members and more critical services, their operationality becomes more important, and to measure this some statistics are needed. This means that the need for service monitoring and usage statistics is growing. There are wide range of monitoring systems suitable for network and service monitoring, but it turns out that the existing solutions do not fit well for the needs of AAI monitoring. The distributed nature of AAI adds complexity, since the AAI crosses organisational borders and there is no centralised entity in charge. The distributed

management, the handling of sensitive personal information and the complex and varied deployment environment makes it challenging to maintain the federation, and it also makes it challenging to build a monitoring and a reporting system for them. To overcome these, this thesis describes the design and implementation of a monitoring and reporting system suitable for web-based authentication and authorisation infrastructures.

1.2 Problem Statement

The objectives of this thesis are to design and implement a monitoring and reporting system for federations. The main functional requirements are that the solution:

- Can monitor the availability of the federation providers. This means, that the system must be able to determinate whether the components are working and configured correctly - so the users can actually access the resources.
- Can produce usage statistics, that can be used in monitoring and developing the federation. This includes the number of login events in the federations services.
- Does not compromise the security of the federation.
- Does not compromise the privacy of the end users.
- Does not notably affect the performance of the federation.

These requirements are discussed in more detail in **Chapter** 4.

1.3 Evaluation Criteria

The main purpose of the work is to produce a practical and useful solution for monitoring and reporting in AAI. One way to measure successfulness is how the requirements are met.

The other way of measurement is to look how quickly and comprehensively the solution is deployed. Due to nature of distributed environment and organisations there are however lots of other dependencies slowing down the deployment. These are not purely technical or can be affected by this work – the workload of the system

administrators, their technical competence, data security policies in universities are examples of factors that will have effect on the speed of deployment.

The monitoring and reporting system will collect status data on how the federation works. This data can be used in defining an evaluation criteria - the number of false alarms versus real cases, the required maintenance work, the detection and problem solving time in failure cases can be used when estimating the success of this work. However, as there is no historical data, the real value of the work can not be calculated just by numbers.

1.4 Scope

The architecture is designed to fit for the monitoring and reporting needs of Haka federation, and tries to make it possible to use it even in upcoming eduGAIN confederation.

The implementation is focused on the Shibboleth software and protocol, and in particular its usage in Finnish higher education federation Haka.

1.5 Structure

This thesis is structured as follows:

Chapter 2 describes the main concepts of AAI and architecture of Shibboleth and eduGAIN authentication and authorisation infrastructures.

Chapter 4 explains the requirements of monitoring and reporting that were defined in the beginning of this work, and introduces some existing applications that are used for network systems monitoring and reporting and how they meet the requirements.

In **Chapter** 5, the architecture, design and implementation choices of the work is described.

The results of the work are then summarised in **Chapter** 6.

Chapter 2

Authentication and Authorisation

This chapter introduces Authentication and Authorisation Infrastructure, its main ideas, protocols and components and how they work together. The protocols and their interoperability is discussed and two examples of identity federations – Haka federation and eduroam – are introduced.

2.1 Authentication

There is a need to distinguish users in many applications to provide personalised service for the users, for example when using e-mail. A computer system needs to find out who the user is and link the user to her personal data. E-mail address and social security number are examples of identity attributes, which form the users digital identity.

Authentication is a procedure where user's identity is verified. There are numerous different authentication mechanisms, which can be categorised in several ways. The authentication mechanism can depend on

- Something that the user *knows* (username and password, for example).
- Something that the user *has* (an ID card, private key).
- Something that the user *is* (biometrics [10]).

or some combination of these. For example, an ATM requires a valid card (which the user has) and a correct password (which the user should know).

A problem when authenticating users is confidence in authentication strength. By relying on username and password there is always a possibility that they are used by someone else than the person they belong to. Relying on something that the user has, like car keys, has similar weakness. On the other hand, using more secure authentication methods may be too expensive considering the risks. With authentication strength we can sort authentication mechanism based on the confidence we have on them.

The strength of authentication and level of assurance depends on the authentication mechanism, but a strong authentication mechanism is not sufficient alone. The identity of the user must be ensured properly when opening a user account, and every time when the identity data is altered. Changing a password is one example – the request for password change must also be authenticated properly.

2.2 Authorisation

An authorisation decision is done after authentication. Positive access control decision means that the authenticated user has access to the service. There are several ways to do authorisation decisions, depending on their authorisation policy. One simple way is that all authenticated users are also authorised, but things can go far beyond that. An authorisation decision can be based, for example, on:

- The roles that the identity has (A professor has an access to student's grading system, while a student has not).
- Time; the authorisation may be dependent on some time interval (the doors of the campus are open only daytime).
- The strength of the authentication.
- Delegation; rights may be delegated to other user.
- The requested data or service.
- A (physical) location; an IP address range (Intranet is only accessible inside the company's own network).
- Access control list or attributes of the requested data or service.

2.3 Authentication and Authorisation Infrastructure

The idea behind Authentication and Authorisation Infrastructure (AAI) is to offer authentication and authorisation as a service which applications may rely on. Middleware software (see **F**igure 2.1) is defined "as a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers" [14]. AAI software can be categorised as middleware as it removes the need for every application to define its own mechanism for authentication and authorisation. An *Authentication Service* is a service which authenticates the user and produces authentication statements.

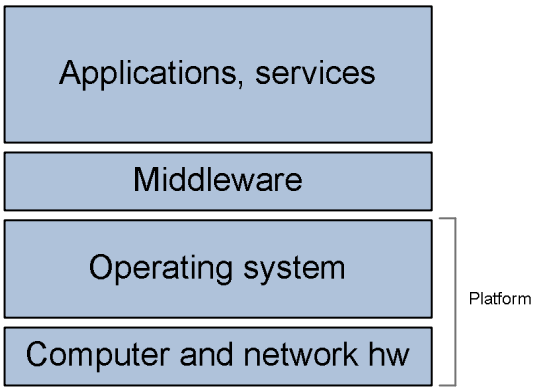

Figure 2.1: Software and hardware layers in distributed systems [14]

An *Identity Provider* guards the identity information of its users. Identity Provider offers authentication service, and it may offer attribute service, releasing the identity attributes to the Service Providers that are requesting the information.

A *Service Provider* protects the service that requires authentication. It's role is to request authentication from the Identity Provider, if the user has no valid session. The Service Provider consumes Authentication and Attribute assertions, and it may make authorisation decision based on them or let the application handle it.

2.4 Federated Identity

Traditionally, a user has multiple identities – an information about the user is stored in every service she is using. For example, in academic world, user's e-mail, name and student id number are common examples of personal information which can be

stored in dozens of services she is using. For every service, she has to register and supply the information needed, if the services are not connected to some centralised directory. Since the services can be offered by various organisations and companies, this is not often possible. The services need usually some kind of authentication, so the user has to login every time she wants to access the service, and for security reasons, several passwords are required.

Another problem comes from the service providers point of view. Every service has to handle and store information about the user and that data must be up-to-date, which is practically impossible to guarantee, when the only source of information relies on users activity.

Federated identity simplifies both end users and service providers task. A single identity is created and hosted in the users home organisation, and that identity is used throughout the federation.

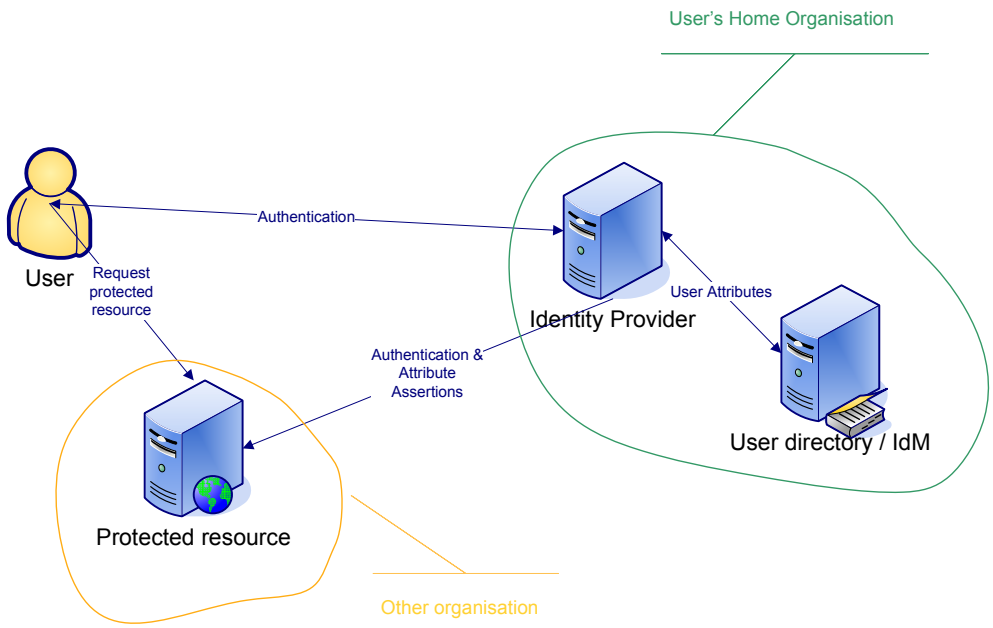

Figure 2.2: Federated Identity

A federation is a trust network – federation's member trust each other on identity management. This poses high requirements for home organisation's identity management, since all the Service Providers will trust the information that they share. If there is only one place where user's attributes are stored, it is enough to keep that source updated. For example, when a user's address changes, it is enough to inform the home organisation about the new address, not all of the services where

the user has an account. In practice, most services will still need some data that is related to the specific service and the specific user. Using federated identity means that the question of where the data should be stored needs to be answered when designing a new service. A service may get all of its user identity data from the identity provider, it may store some data locally or it may only use the identity provider for authentication.

Federated identity has some benefits for the end user also, apart from reducing the work for updating her identity information in different places. It enables a single authentication mechanism to be used for every service in federation – a single password, biometrics, etc. – whatever authentication mechanism that is implemented in her home organisation's identity provider. Using this identity provider, a single sing-on solution may be offered even between different organisations and their services. The service providers can benefit by saving human work, as the need for credential provisioning and helpdesk work for recovering lost passwords is reduced, when the identity is stored in a single place.

The number and quality of passwords is one of the biggest risks in traditional services. The users may use passwords that are easy to guess, and the same password can be used in many services. Federated identity management changes the situation – the risks are even bigger, since one password is enough to access whole federation's services. But since the number of passwords is reduced, stronger passwords can be forced to use. Stronger authentication mechanisms can replace the passwords, and only the home organisations have to implement them – not every service. It is not needed to store sensitive passwords and personal information on every service, which will reduce the risk of identity theft in case of services get compromised.

2.4.1 Attributes

One main benefit of federated identity management is possibility to store user information only in her own home organisation, and this information can be queried from the home organisation's Identity Provider by Service Providers. The information transferred is called *attributes*, which syntax and semantics must be agreed between the service requesting the information and the Identity Provider. Typical examples of attributes are e.g. an e-mail address or a person's name, and some identifier which uniquely identifies the person within a federation.

If the Service Provider and the Identity Provider are part of larger federation, the attributes may be defined for the whole federation. This removes the need to agree

on attributes between every Identity and Service Provider. In Haka federation for example, funetEduPersonSchema [17] is a schema defining the syntax and semantics of attributes exchanged within Haka federation. There are other schemas, like Schac or eduPerson, which are used in other federations around the world.

2.4.2 Managing Trust

The idea of federated identity requires trust between participating parties. The Service Provider must rely on Identity Provider to implement proper authentication and to provide valid and up-to-date attributes about the end user. On the other hand the Identity Provider must trust that the Service Provider handles the possibly sensitive identity information in a proper way. Establishing trust is not a technical question that AAI software can solve – it is a question that must be solved offline. The level of trust requirement varies, some federations may require very high level of trust while some others can work with lower level trust. This depends on the federations services and the trust requirements must be thought when adding a new Service Provider to a new federation.

A federation is basically a trust network. The trust relationship is born by joining a federation. Depending on the federation model, this relationship may be bilateral between every federation member, or it may be transferred to some degree.

The end user needs a trust relationship too. She should be concerned on how her digital identity is stored, transferred and used within a federation. In European Union there are strong laws and directives helping to protect the privacy. For a Service Provider this means that only those attributes that are necessary for providing the service may be used. The user must also be informed what personal information is transferred and how it is used and released to third parties.

2.4.3 Single Sign On and Single Logout

Single Sign On solutions are a way to simplify the login procedure. The user needs to authenticate herself only once – this authentication information is then available for other SSO aware applications too.

While it is possible to build a web-based SSO systems, the question of single logout is perhaps harder. A logout button can be presented in the Service Provider after the user has logged in, and it can log out the users session on that Service Provider. But the session still exists in the user's Identity Provider and in the other Service

Providers she has used in this session. The difficulty comes from knowing all the providers involved - and even if they can be known, the stateless nature of HTTP makes it hard to build a reliable logout functionality. Using HTTP redirects, the user's browser could be sent to all the providers necessary, but what would happen if some site does not respond? The redirect chain would be incomplete, leaving the session valid in part of the providers.

Figure 2.3 represents a situation where user is accessing several applications. There are multiple sessions – the Identity Provider has a session, each Service Provider have a session, and the applications often have their own session handling too. A traditional logout would end only the session with each application, but leave session still valid with Service Provider and Identity Provider. A Single Logout, as specified in SAML specification, would have to clear the session in each Service Provider the user has accessed, and in the Identity Provider. A cooperation is then needed to clear the sessions from accessed applications too.

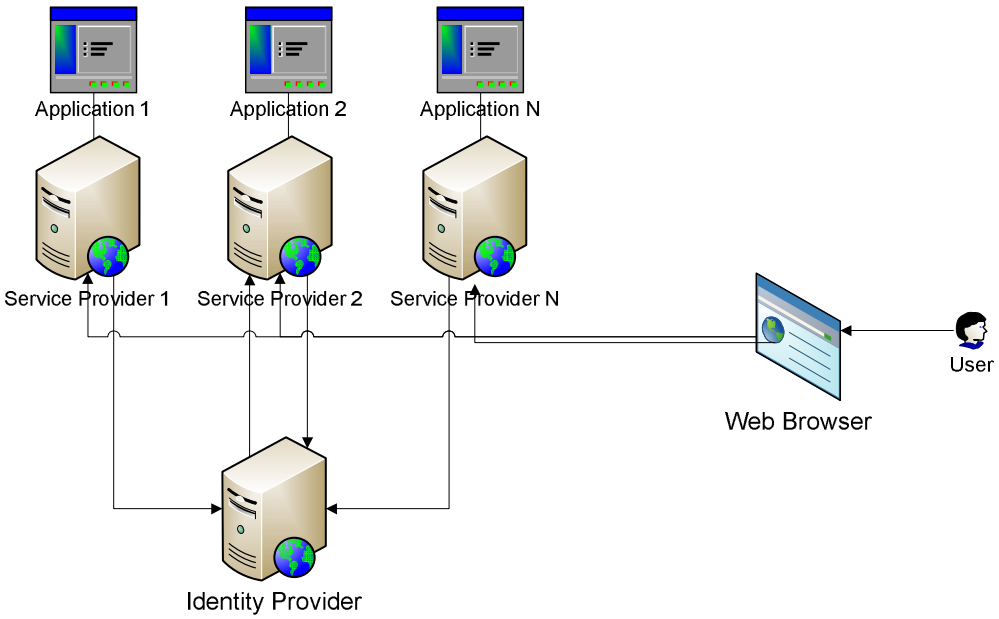

Figure 2.3: Single Logout Problem

However, it is not clear what kind of functionality the user is expecting when she clicks on the logout button. Traditionally, this would log her out of the application where she requested logout without affecting other applications. The Single Logout behaviour might be unpleasant for her – the applications may be completely unrelated to each other and single logout can be unexpected and unwanted result.

2.5 Security Assertions Markup Language (SAML)

The Security Assertion Markup Language (SAML)[26][12] is a framework for exchanging security information between partners. It is based on XML, SOAP and HTTP - authentication and authorisation information transferred in a common XML format between applications. This authorisation information – attributes – makes it possible to apply complex authorisation policies in service provider, possibly combining attributes and deciding authorisation on the combined values.

```
<samlp:AuthnRequest
  xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
  AssertionConsumerServiceURL=
"https://aitta2.funet.fi/Shibboleth.sso/SAML2/POST"
  Destination=
"https://aitta2.funet.fi:8443/idp/profile/SAML2/Redirect/SSO"
  ID="_3f0bbf72ad491c03452c139b58c811a4"
  IssueInstant="2010-01-18T05:35:02Z"
  ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
  Version="2.0">
  <saml:Issuer
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
      https://aitta2.funet.fi/shibboleth
    </saml:Issuer>
    <samlp:NameIDPolicy AllowCreate="1"/>
    <samlp:RequestedAuthnContext>
      <saml:AuthnContextClassRef
        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml:AuthnContextClassRef>
    </samlp:RequestedAuthnContext>
</samlp:AuthnRequest>
```

Listing 2.4: SAML Authentication Request

SAML assertions can be transferred using *front-channel* or *back-channel*. The back-channel is a connection between Service Provider and Identity Provider using HTTP/SOAP protocol for transferring the assertions. The front-channel uses no direct communication between Service Provider and Identity Provider, instead, the user's web browser using normal HTTP protocol with GET or POST method is used.

To ensure the message integrity and confidentiality, SAML recommends using HTTP

over SSL 3.0 or TLS 1.0, and authentication should be done for both server and the client. XML signatures must be used when transferring assertions using Browser/POST profile, which is described later in this section. XML Encryption may be used to assure message-level security when SSL/TLS is not used or additional protection is wanted.

Today there exists numerous SAML implementations, both as commercial products as well as open-source. Some examples are listed in **Table** 2.1, but the list is not comprehensive.

	Supported Protocol
Lasso	ID-FF 1.2, ID-WSF, SAML 2.0
Microsoft ADFS	WS-Federation
OAuth	OAuth protocol
OpenID	OpenID protocol
OpenSSO	SAML 1.1, 2.0
Ping Identity	SAML 2.0
simpleSAMLphp	SAML 2.0
Shibboleth	SAML 1.1, 2.0
ZXID	SAML 2.0

Table 2.1: Examples of AAI software

2.5.1 Liberty Alliance and WS-* Protocols

The Liberty Alliance is a consortium formed in 2001 for developing common specifications for federated identity management. They have developed the Liberty Federation, including ID-FF 1.1, ID-FF 1.2 and SAML 2.0 specifications. To help the interoperability between different products and vendors, Liberty Alliance organises SAML 2.0 interoperability testing.

Microsoft and IBM have developed their own specifications for federated identity, including WS-Security, WS-Federation, WS-Trust and WS-Policy and several other specifications. These specifications describe different security tokens, infrastructures and trust topologies. Though WS-* can be configured to accept SAML tokens, the interoperability is limited. However, Microsoft's ADFS 2.0 (Geneva) services, which

are currently at beta stage, do support SAML 2.0.

2.5.2 SAML protocol versions 1.1 and 2.0

The SAML protocol specification converged from Liberty Alliance's work and OASIS SAML 1.1 version to SAML version 2.0. This version brings in major changes, and it is incompatible with previous versions. Main new features are support for W3C XML Encryption for encrypting SAML assertions, merging the Browser/Artifact and Browser/POST into a single Web Browser Single Sign-On (SSO) Profile and a new Identity Provider Discovery Profile.

Though SAML 2.0 is not compatible with previous versions, the main benefit is better compatibility between SAML 2.0 implementations. The full protocol specification is however large and complex, and most implementations implement only a small subset of it. This in turn may cause incompatibility, which is handled by defining SAML deployment profiles and running interoperability tests. The Liberty Alliance project does this kind of interoperability testing, and one example of defining SAML deployment profile is saml2int[32] – a simple, but useful SAML profile for Web SSO that defines the mandatory features of SAML that must be implemented to be saml2int compatible. **Figure** 2.5 shows relationships between Oasis standards, Microsoft WS-* and Liberty Alliance.

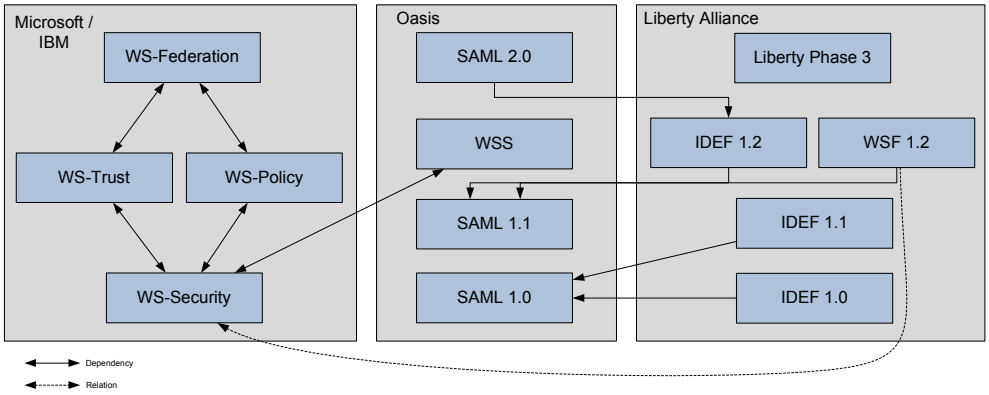

Figure 2.5: The interrelationship among federation standards [30]

2.5.3 Profiles

A SAML profile is an exact description how a supported use case of SAML is implemented, the most important use case for this thesis being Web Browser Single

Sign-On (SSO).

SAML 1.1 defines two profiles. Browser/POST profile, like its name suggests, relies on HTTP POST method as a way to transfer assertions to the Service Provider. Once the user has authenticated herself, a HTML POST form is presented. This form contains the assertion, which is then (perhaps automatically using Javascript) sent to the Service Provider. The SAML response containing the assertion must be digitally signed using XML digital signature which is described later in this chapter.

Browser/Artifact is another profile; it uses pull model for assertions instead of push. The user has an artifact, which is an identifier containing information about the source site and a reference to the assertion. Using this reference, the Service Provider can request the assertion from Identity Provider using back-channel. The back-channel is implemented by HTTP/SOAP connection between the Service Provider and Identity Provider.

In SAML 2.0, these two profiles have been merged into single Web SSO profile.

2.5.4 XML Digital Signatures

The SAML assertions can be transferred in unencrypted format which would make them vulnerable for alternation attacks. For example, using Browser/POST profile means that SAML assertions are transferred via users web browser, making it possible to alter the content for the user. It is also possible to use HTTP as a transfer protocol instead of HTTPS, where assertions are transferred unencrypted all the way. To deal with these risks XML digital signatures[19] are mandatory in SAML assertions when using unencrypted transfer methods. XML signature is W3C Recommendation, and it can be used to guarantee the integrity of any XML document.

2.5.5 SAML Metadata

In SAML the trust is largely based on SAML metadata [11]. SAML metadata defines the providers that are trusted and their technical contact details, like addresses and protocols and profiles that are supported. The Identity Provider needs to have valid SAML metadata about the Service Provider in order to provide authentication and attribute statements to Service Provider. Vice versa, the Service Provider needs the metadata about the Identity Provider, otherwise it can not trust received assertions.

The metadata is provided in XML file, which can be distributed bilaterally, or if

provider is part of a federation, a federation may distribute a common metadata. A federation metadata includes the metadata of all participants of the federation - the Identity Providers and Service Providers. Its integrity and availability are important, since all providers will trust its contents.

Federation metadata needs to be updated when federations participants change their deployment, new Providers are added to the federation or existing providers are removed. When all trust is build on the metadata, removing a compromised service from federation requires a metadata update. Thus the Providers need to update that file regularly.

2.5.6 Identity Provider Discovery

When a Service Provider decides that the user who is accessing its resources must authenticate herself, it should redirect the user to her Identity Provider's authentication service. The problem is that if there are multiple Identity Providers, in general the Service Provider has no knowledge of the user and what her Identity Provider might be. In some cases it is possible to send all users to the very same IdP, if that is the only one whose users are authorisated to access, but if there are several Identity Providers providing access, this is not possible.

Instead of redirecting the user to the IdP, it is possible to use a Discovery Service (DS) [29] or Where Are You From (WAYF) service and redirect the user to it. A DS/WAYF service is an independent component that decides the user's Identity Provider with or without user interaction. A typical DS/WAYF service displays a list of Identity Providers and asks for the user to select which one she wants to use. This selection can be remembered or pre-selected by guessing based on the end users IP address for example.

It is also possible to integrate the Identity Provider selection functionality directly into the Service Provider. The Service Provider then has a user interface for selecting the Identity Provider where it will redirect the user. This removes a single point of failure, which centralised WAYF service introduces, and simplifies the user experience, since the service can only provide links to those IdP's whose users can actually access the resource. On the other hand, it requires some programming logic on the Service Provider instead of simply redirecting users to an external service.

Identity Provider discovery is a simple question technically. Instead, it has significant usability problems. The federation may have hundreds or thousands Identity Providers, and it is hard task to find the correct one. When existing federations are

combined as a confederation this will make Identity Provider selection even harder. The end user may not even know what Identity Provider to use – some users have account on several Identity Providers, and authorisation may be possible only when using some of them. The user must also use the Identity Provider consistently – otherwise, using different Identity Provider may result in using different identity, since those two accounts may not be linked within a service.

2.6 Shibboleth

Name *Shibboleth* is used here in two meanings - it is the name of the software [21] developed by Internet2, and it is the name of the protocol that the software uses. In this section, an overview of both Shibboleth software and protocol is given.

Shibboleth 1.x as a protocol is an extension to OASIS SAML 1.1 profile. Most importantly, it adds the Service Provider first -approach as the SAML 1.1 specifies only Identity Provider first -approach which is simpler. In the IdP first -scenario, the login sequence starts when user authenticates herself on the IdP, which then redirects the user to the service. In a federation with only one IdP this approach can work pretty well. The SP first -approach makes it possible to begin the login sequence by accessing a protected resource, and then be redirected to the Identity Provider for authentication as the Service Provider sends an authentication request to the IdP. This scenario requires that the IdP discovery problem is somehow addressed; for example by using a WAYF service described earlier. The downside of using Shibboleth 1.x as a protocol is incompatibility with other implementations, since the SP first -approach and some other differences are not part of SAML 1.1.

Shibboleth versions 2.0 and above support SAML 2.0 protocol as well as previous Shibboleth protocol. The default settings support quite well saml2int profile mentioned earlier. A major change when moving to Shibboleth 2.0 is that attribute queries are left out by default, encouraging that the attributes are transferred via user's web browser. This change makes deployment of Shibboleth easier, as there is no need to configure back-channel support.

Both versions of Shibboleth use a subset of SAML metadata. The trust management can be done in two ways. The currently recommended way is to define Service and Identity Provider credentials directly in the SAML metadata, either by using X.509 certificates or plain RSA/DSA public keys in the metadata. The other option is to use PKI approach, and define a Certificate Authority (CA) certificate. The Identity

and Service Provider certificates are then validated against this common CA.

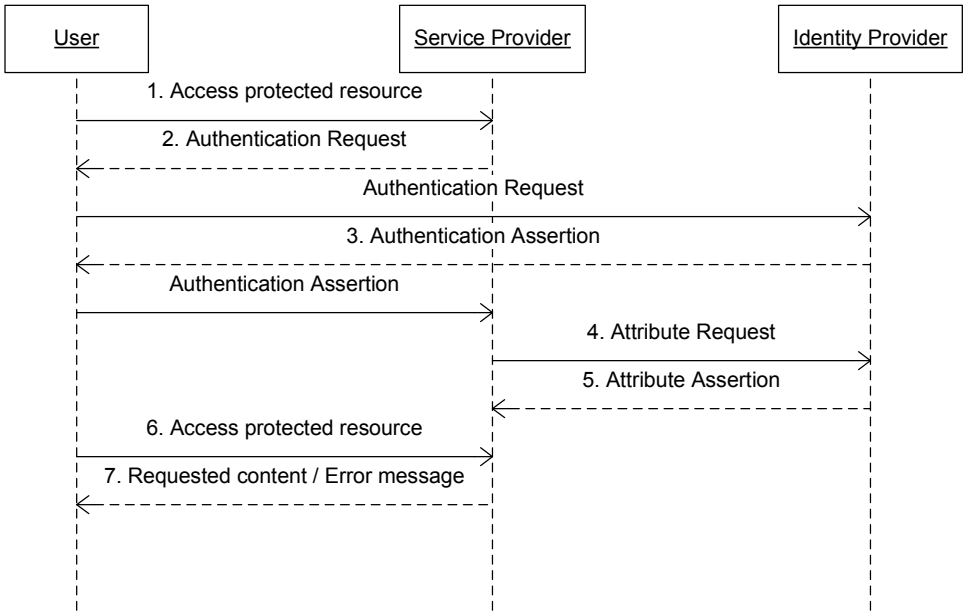

Figure 2.6: Shibboleth Login

Figure 2.6 shows the login procedure using Shibboleth with Browser/POST profile.

1. User requests some resource that is protected by Shibboleth.
2. Service Provider requests an authentication. It checks if the user has a handle describing her session - if not, a Authentication Request is issued. The user may need to use a separate WAYF service to proceed to her Identity Provider.
3. After authenticating herself in her Identity Provider, the Identity Provider assigns a handle describing this session. The handle is implemented as a cookie or it may be attached to the URLs used.
4. The Service Provider will then ask for attributes from the Identity Provider the user has used to authenticate herself. This is done by Attribute Request, which contains the handle of the user.
5. The Identity Provider releases requested attributes about the user to the Service Provider.
6. If using Browser/POST profile, the attributes are transmitted to the Service Provider using a HTML form.

7. Based on the attributes, the Service Provider does an authorisation decision and returns the requested resource, or an error message.

2.6.1 Identity Provider

An Shibboleth Identity Provider is responsible for user's attributes. A Authentication Authority component issues authentication statements to other Shibboleth components.

When a user is accessing a resource and gets redirected to the Identity Provider, first component that is connected is Single Sign-On service (SSO). A Single Sign-On component is a Shibboleth specific - SAML 1.1 does not define it. SSO component enables Service Provider first -approach, while SAML 1.1 supports only Identity Provider first -solution.

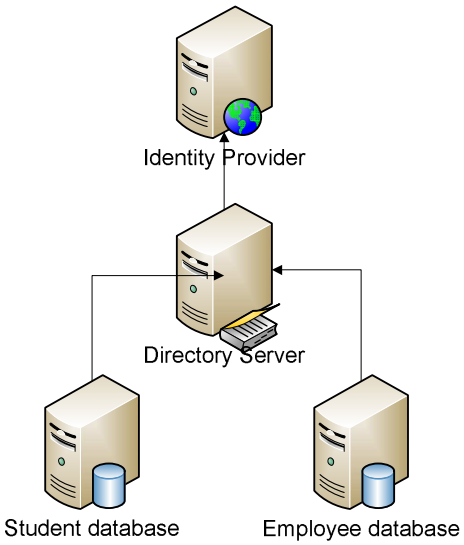

Figure 2.7: Generating Attributes

Artifact Resolution Service is needed if Browser/Artifact -profile is used. It will listen to Service Providers, which will send artifacts to Artifact Resolution Service in order to get authentication assertion.

Attribute Authority answers to attribute requests send by Service Provider's Attribute Requester component. The attribute requests and assertions are transmitted using back channel without web browser.

2.6.2 Service Provider

Shibboleth Service Provider is responsible for protecting the resources hosted by a HTTP server. It is implemented as a module for HTTP server, like Apache or Microsoft IIS, so it is limited to WWW applications. The authorisation decision can be done by requiring specific attributes and specific attribute values. Shibboleth Service Provider's main components are Assertion Consumer Service and Attribute Requester.

Assertion Consumer Service handles the authentication assertion. If attributes are needed, Assertion Consumer Service will initiate an attribute request using Attribute Requester component. If all goes well, the Assertion Consumer Service will finally redirect the user to the requested resource.

Attribute Requester is responsible for sending attribute requests to the selected Identity Provider.

2.7 Haka federation

The Haka federation [22] is a identity federation for higher education in Finland. It was formed in May 2005 and it became operational in August 2005. Haka is open for Finnish universities, polytechnics and research institutions to join as a federation member. Federation member may bring its Identity Provider and Service Provider(s) to Haka. Commercial companies may join Haka as partners if they are providing services serving education or research. Haka federation partners may not register Identity Provider to Haka, but only Service Providers. Joining Haka as an Identity Provider requires that the organisations identity management passes a self-audit, which requires some good practices are used when user accounts are created, how they are updated and how identity management process works in the organisation.

CSC – IT Center for Science acts as a Haka operator. Joining Haka means signing a service agreement with CSC. In this hub-and-spoke federation model [30] a single contract is enough, there is no need to sign contract with every member of Haka – the trust and agreement is transferred via the federation operator. Federation operators other main tasks include maintaining federation metadata, the Identity Provider Discovery Service, test services and technical support for members and partners.

The end users of Haka are higher education students, teachers and faculty. At the moment there are 39 Identity Providers and 78 Service Providers in Haka. Twelve commercial companies have joined Haka as a partner, providing their services for Haka community. Most heavily used services in Haka are library and learning management systems, e.g. Moodle systems.

Technically Haka started with SAML 1.1 protocol implemented by Shibboleth software. When Shibboleth 2.0 release added support for SAML 2.0, Haka started the migration process to SAML 2.0. This process is still ongoing and should be completed by the end of 2010, when all services must be SAML 2.0 compliant. The main motivation for migrating the federation's protocol is to enable interoperability between various products, and first non-Shibboleth implementations using only SAML 2.0 have been in use since the end of 2009.

For attributes Haka federation uses funetEduPerson [17] attribute schema which is based on SCHema for ACademia (SCHAC) [9] and eduPerson schemas. The funetEduPerson schema adds common attributes for Finnish higher education and defines syntax and semantics for them. An example of such attribute is funetEduPersonStudyStart attribute, which describes the date when a student started his/her studies. Technically, the schema defines the name of the attributes in three format – a human readable format, an URN, which is used when using Shibboleth 1.x software, and OID format which is used with SAML 2.0. An example of this attribute definition is in **Table** 2.2.

OID	1.3.6.1.4.1.16161.1.1.14
URN	urn:mace:funet.fi:attribute-def:funetEduPersonStudyStart
Format:	YYYYMMDD
Example:	20050826

Table 2.2: funetEduPersonStudyStart attribute definition

2.8 The eduroam

The eduroam [1] is a service for secure roaming network access for education networks around the world. It is based on RADIUS servers and IEEE 802.1X standard. Authentication in eduroam is done by forwarding users login credentials to her home organisation. For organisation to join eduroam, setting up a RADIUS server is re-

quired. This server authenticates organisation's eduroam users. Also the WLAN hotspots must be configured to support eduroam requirements, for example they must be IEEE 802.1X capable, and capable of forwarding credentials to the RADIUS server. Technically, the eduroam RADIUS servers are used hierarchically (see **Figure** 2.9). The top-level RADIUS server forwards the authentication requests to national level server, which in turn knows the organisational level servers and sends the authentication request there. The eduroam does only authentication, but in [13] it is presented how authorisation can also be added to eduroam.

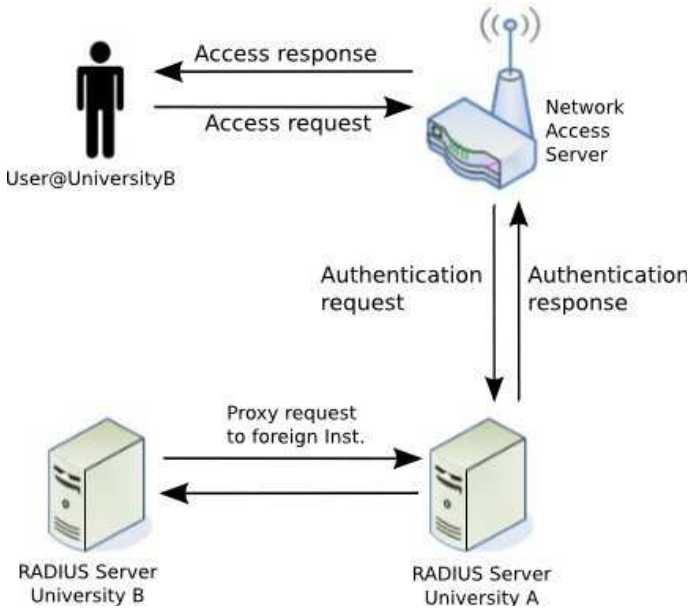

Figure 2.8: eduroam infrastructure [13]

Though eduroam uses different technology than SAML and is a bit out of scope of this thesis, it is useful to have a closer look on it since the problems are similar. Both eduroam and web-based AAI are distributed infrastructures and suffer from same kind of difficulties in maintenance and diagnostics. The monitoring solution for both infrastructures can well have similarities.

The eduroam is an example of confederation; national federations have joined together to form the eduroam confederation. This enables the end users of one federation to get network access even outside her own federation and country. There is no accounting, so no money is directly involved. The reasoning is that "if you let my users to use the network, I will let your users to do the same", and in eduroam, this works between organisations around the world. The idea of confederations is discussed in more detail in the next chapter.

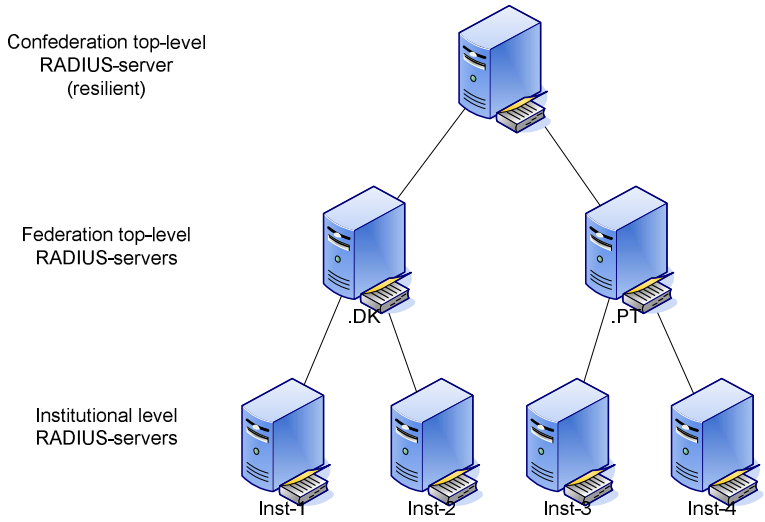

Figure 2.9: eduroam RADIUS hierarchy [31]

Chapter 3

Beyond federations

This chapter takes a closer look on how existing federations can be joined to a single confederation. The term "confederation" refers to "a federation of federations" and that's how federations can work together to offer their services for other federation's users too. The problems of confederations are discussed, and the Kalmar Union and the eduGAIN confederations are introduced in this chapter.

3.1 Building interoperability between federations

As discussed in the previous chapter, federated identity has become a reality and sees production use today. Many federations have been established around the world. In the higher education field, federations are mostly within national borders, available only to members from their home country. But there are potential use cases where it is feasible to use federated identity even across these borders, like the example of eduroam shows.

In the federated world there exists many different federations which are not compatible, either in technical means or by policy differences, or, most likely, both of them. For example, in Finland there is a federation for higher education (Haka), and in Norway there is operating Feide federation [4] for their universities, but if Finnish students want to access Norwegian resources it is not possible, since the user's identity is in different federation than the service. Such scenarios may arise also inside national borders – there might be several commercial federations, an educational or state federations, so there is a need for interoperability between federations. Confederations, or interfederation, aims in making this kind of use cases

possible.

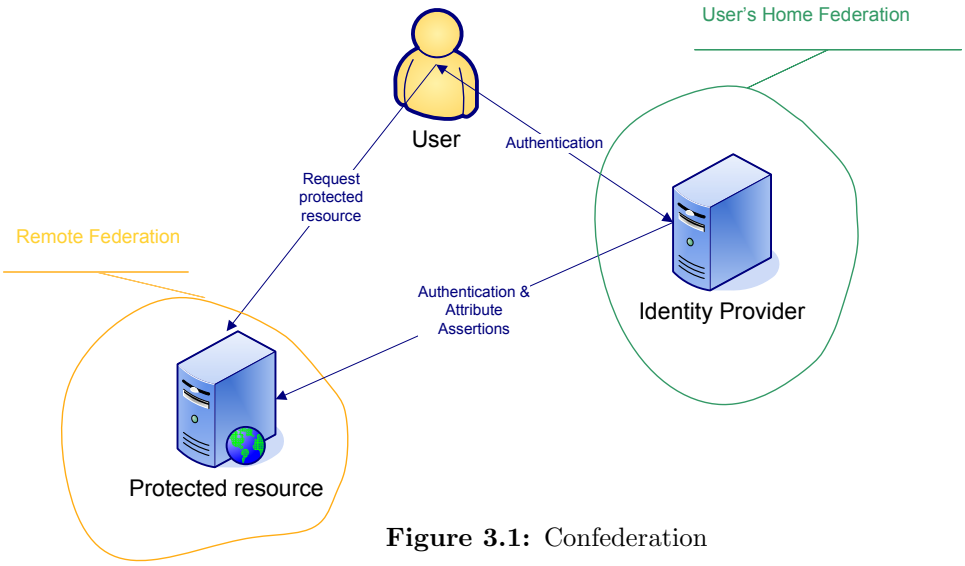

Figure 3.1: Confederation

3.1.1 Technical and practical challenges

There are however problems when building a confederation. From the technical point of view, the variety of AAI systems is an issue - generally they are not compatible with each other, each using different protocols. In the future the compatibility is likely to increase, as many AAI's are moving towards or already using SAML 2.0. Unfortunately SAML 2.0 can be implemented and used in many ways that interoperability is not clear even when having a common protocol, as was discussed earlier.

Federation bridges[20] may be implemented when the AAI software is not compatible. The idea is to do protocol conversion at some bridge, which acts as gateway between federations. A federation sees the bridge as an Identity Provider while the other federation sees it as a Service Provider. Attributes can also be transformed in the bridge. The bridging solution has some drawbacks however. It is a single point of failure. It will break the confidentiality of the traffic, since it needs to be able to decrypt the messages in order to do any transforming, so end-to-end confidentiality is lost. And quite likely some functionality is lost when using bridging, since not all features may be supported by both federations and by the bridge.

A common protocol and technology is not enough. A federation is basically a trust network, and establishing trust is mostly non-technical issue. Different federations typically have different requirements for joining the federation and different levels

and requirements for security and privacy. For example, in Haka federation it is required, that the end users identity is verified using an official document such as passport. Not all federations have such requirement, and when this kind of federations are interconnected this may become a problem for the services.

Not only the technical part has challenges, but the policies become important issue when connecting different federations. For example, a term "student" can have quite different interpretations between universities – even inside on country, but especially between countries. To effectively use AAI's as an authorisation mechanism, there must be some sort of agreement on the syntax and semantics of transferred identity data. A schema harmonisation work tries to find commonly agreeable interpretations on the attributes. Also the privacy and data protection laws have to be considered when building confederations.

Knowing Your Federation

The question of knowing user's Identity Provider is harder with multiple federations. Like WAYF, a WFAYF (Which Federation Are You From) service could be implemented to offer a way to determinate to which federation's WAYF the user is to be redirected. The idea is however as cumbersome as the acronym - perhaps more intelligent solutions will appear as a Service Provider component or even as a web browser add-ons.

3.2 eduGAIN

The eduGAIN [24] is an AAI that has been developed by Géant 2's JRA5 project with interoperability of the federations as a main focus. It will bridge different AAI implementations and thus allows connecting federations that are using different AAI that are not directly compatible. eduGAIN is SAML 1.1 compatible, however, it has been planned to move to SAML 2.0 as open source implementations stabilise – this work is being done in Géant 3 project.

The eduGAIN defines new components to enable interoperability. *Bridging Element* (BE) is responsible for transferring eduGAIN assertions and queries to local federations format and vice versa, for example, there is an implementation for bridging Shibboleth for federations using it. For a federation to join eduGAIN it is required that it will install a bridging element which supports the software used within a federation, if the federation's software is not directly eduGAIN compatible.

Metadata Service (MDS) answers metadata queries. It will contain the necessary SAML metadata of the eduGAIN Service and Identity Providers. MDS is queried by BE's, and it acts as a common trusted source for eduGAIN metadata. The *Federation Peering Point* (FPP) is responsible for publishing metadata to Metadata Service. There is exactly one FPP per federation. The metadata is never signed by MDS, its function is only to forward the metadata. Thus there are no high requirements for trusting MDS, since the federations will sign their own metadata by themselves.

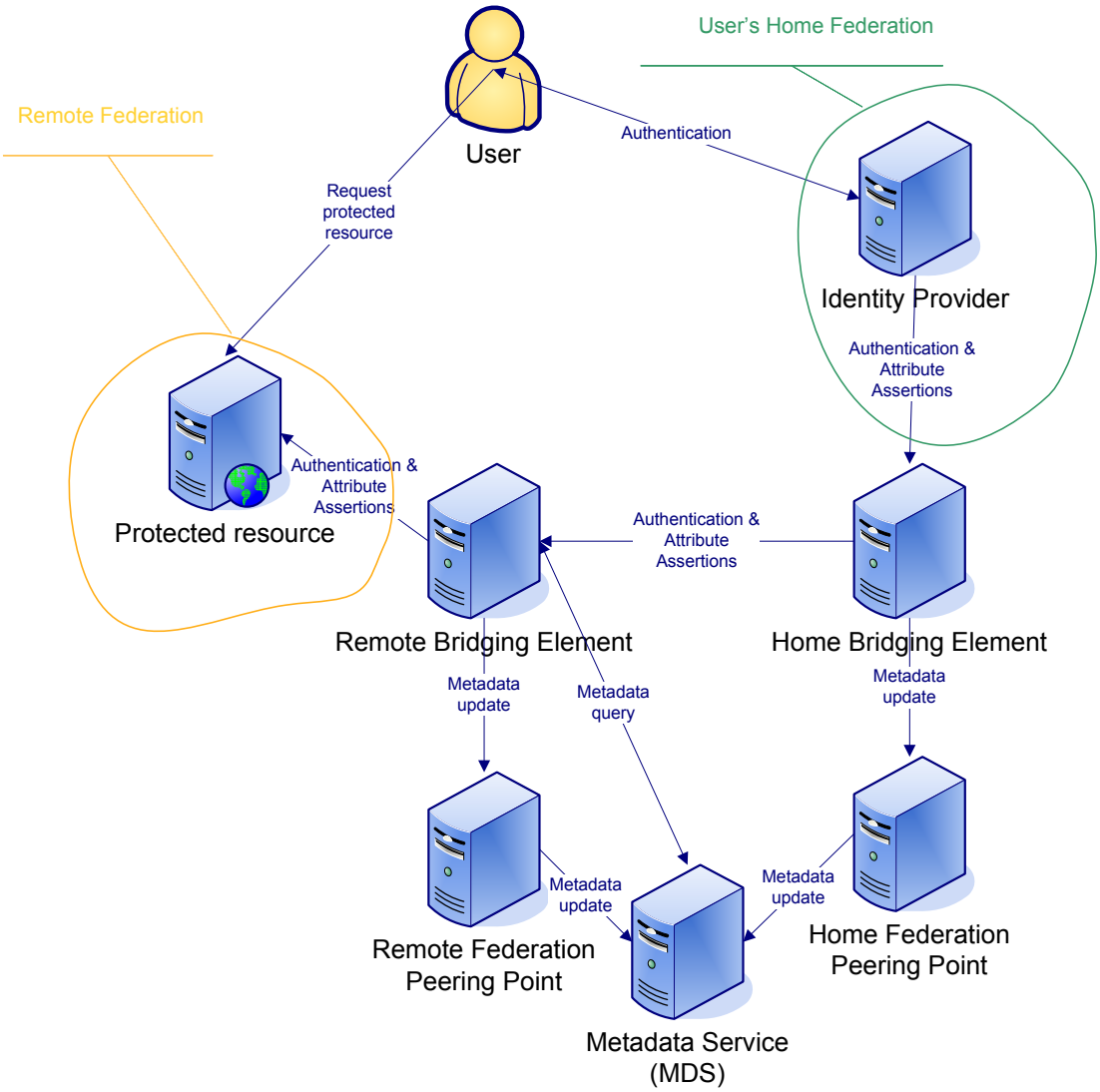

Figure 3.2: eduGAIN login procedure

Trust in eduGAIN is managed with public-key infrastructure (PKI). The eduGAIN

has a common Certificate Authority (eduGAINCA), which is used to create eduGAIN X.509 certificates. The XML signatures used in SAML assertions must be validated against this certificate authority.

3.3 Kalmar Union

Kalmar Union [7], [28], [23] is a Nordic confederation which joins national higher education federations of Finland, Sweden, Norway, Denmark and Iceland. A contract has been signed between national federation operators and Kalmar Union went into production use in September 2009. For Service/Identity Provider to join Kalmar Union it is required to join some national federation first.

The architecture of Kalmar Union is significantly simpler than eduGAIN. All participating federations are using SAML 2.0 compatible protocol, so no protocol bridging is required. There is still need agree on how exactly SAML 2.0 protocol is to be used since the protocol offers multiple choices for implementation, implementing and supporting all of them would be impractical for all confederation members. The interoperability in Kalmar Union is based on saml2int profile[32] which defines mandatory profiles. In addition, the attribute semantics differ slightly within federations, so an agreement on common attributes was needed.

The Kalmar Union offers WFAYF server for selecting the home federation and then a home Identity Provider. It also offers a metadata aggregate containing metadata of all participating services. These are the only centralised services that are needed. For a service already in some national federation, joining Kalmar Union is usually straightforward, the Kalmar Union metadata must be used and service's own metadata must be aggregated to confederation metadata. Since the Kalmar Union is a federation of federations, it is needed that services are part of a national federation first – joining straight to Kalmar Union is not possible for a single service.

3.4 Monitoring a Confederation

When there are challenges to set up and operate a single federation, working with multiple federations is even more complex. Even when each federation is functioning properly, there are components and settings that are needed for interfederation support, like WFAYF service or metadata aggregate service that are used in Kalmar Union, or the various bridging elements used in eduGAIN. A problem with one of

them is enough to cause the system to fail. Many of those service are configured dynamically, as new members are joining national federations or their configuration is changed, and there is always a risk that something goes wrong when doing such updates.

Testing login from different federations is one problem unique in confederation operations. The login may work just fine from one federation, but not from the others. The service administrator may have a user account on one Identity Provider, but this does not help much; it would require multiple user accounts from different federations to be reasonably sure that the service is accessible within whole confederation, which is quite impractical. The number of required test cases also starts to climb, and involves manual work from the administrators. So as the confederations grow and are being used, the need for their availability and availability monitoring is likely to rise.

Chapter 4

Requirements for AAI Monitoring System

4.1 Motivation

The Haka federation steering committee expressed that getting usage statistics and developing a monitoring system is important. CSC as Haka federation operator started to study the problem. Since no off-the-self solution was found during the preliminary study a software for monitoring was to be developed. CSC also participated in Geant2 project where JRA5 activity was focusing on AAI. As the problem of monitoring and interest is common for all federations, part of the funding was provided within Geant2 project. This had an impact for the requirements specification as the needs of eduGAIN had to be taken into account.

4.2 Requirements Specification

The work on monitoring and reporting system started with requirement specification, where the requirements and constrains were identified and defined. These include several functional requirements which define the functionality of the monitoring and reporting system, as well as non-functional requirements, which include usability, stability and scalability for example.

The main functional requirement for the system is that it should be able to reliably monitor the state of the AAI. Since the AAI is complex and distributed system, monitoring just single components of it would not tell the whole truth – the compo-

nents might be working just right, but the end user is still not experiencing what she should. Several settings of the AAI are constantly changing and must be regularly updated to keep the service running – the federation metadata and the attribute release policies as an example. Monitoring all these settings individually would be difficult or even impossible, so at the requirement phase it was clear that monitoring must be done from the end user's point of view. This approach would give the most realistic results too.

Another important functional requirement was that the monitoring system must be able to collect usage reports in Shibboleth-based AAI. There are no tools for this provided within the Shibboleth software, so the administrators of Shibboleth servers have a little idea on how much their services are used and who are the end users. It is possible to collect this information from the log files that the Shibboleth application generates, but there were no tools for this, so some automatised way had to be developed. The usage report collecting system must also be designed in a way that support for eduGAIN AAI can later be added.

It turned out that there would probably be different kinds of deployments of the monitoring system. Not all organisations will create a test account, which is required for the active monitoring, and not all of the organisations are willing to install and maintain any kind of additional software - the probes - in their Identity or Service provider host. These limitations mean, that the monitoring and reporting tool should support various deployment strategies, and allow also partial use. At least the monitoring and the reporting functions should be separate.

4.2.1 Interfaces

Many institutions using AAI are also doing some kind of network/service monitoring already. For example, Nagios [8] is one popular tool for monitoring purposes. To take advantage of this, the system developed should be integrateable to the existing monitoring and reporting applications. As a demonstration of this, a plug-in for Nagios is to be developed during the project. The existence of such applications like Nagios means also that they can be used to perform many common tasks for monitoring and reporting systems. An example would be e-mail notifications of the problems - Nagios has a feature for this, so to take advantage of it, the monitoring tool does not need to implement it. Other commonly needed feature is scheduled service breaks – there has to be a way to temporarily disable the monitoring when doing service maintenance work.

Support for different AAI implementations is also useful. At least Shibboleth versions 1.3 and 2.0 and eduGAIN AAI's should be supported for active monitoring. Since there are many other AAI software in use, it has to be simple to add the monitoring and reporting functions for other applications as well.

4.2.2 Performance

The performance requirements define that the system may not significantly affect the performance of AAI. It is also required, that the parameters that can have performance effects are customisable, to allow modifications depending on the needs of local administrators. Allowing distributed configuration is not a requirement, but an idea to consider in the future. The system administrators of the Identity and Service Providers are the right persons to know how their services should be monitored.

The active monitoring of AAI has some scalability issues if the tests are comprehensive as they should be. To cover all connections, m x n tests are required, where m means number of IdP's in federation and n number of SP's in federation. On the other hand, it may not be necessary to test all connections very frequently, and probably not all IdP's and SP's will ever want to take part in monitoring - at least some commercial service providers might not want to publish their availability statistics. Still, scalability can be an issue when federations grow very large and comprehensive monitoring is wanted. Deciding how often a connection is to be monitored affects heavily on the load and performance of the monitoring.

4.3 Privacy and Security

Both Identity Providers and Service Providers may handle sensitive information, like personal information about the end users - names, social security numbers etc. Identity Providers store this information for all of their users, which usually means hundreds or thousands users. Some Service Providers protect valuable resources - super computers, personal databases, payment systems etc. This all imposes serious requirements for the privacy and security of AAI system and components directly involved with it, like the monitoring and reporting system developed here.

The active monitoring component is a potential security threat, since monitoring requires a test user account and a password, which must be stored on the monitoring host. These user accounts must be protected, since they will give an access to the

protected resources that are monitored. Not all service providers are comfortable with the idea of having such accounts that can be used to access their services. A possible approach to this problem could be to define an attribute, which identifies the user as a test user. The service provider could then decide how it wants to handle these users - it could release enough information for determining that the login procedure was successful, but denying access to any real resource. The downside of this is that customisation is needed to the service's software to handle these restricted users.

Collecting the usage statistics is another problematic area. The log files of Identity and Service Provider may contain lots of personal information about the users. But since this personal data is not required for statistical reasons, a requirement can be that the tool may not handle and store personal information.

4.4 Related Work

At the start of the work, information was gathered about similar monitoring and reporting solutions and about possibly useful technologies for implementation. A few AAI federations are using some kind of monitoring system, and at least within Swiss higher education federation SWITCHaai there has been interest in usage statistics as they have studied adding accounting to AAI [27].

Though it seemed that no easily adoptable solution can be found, the solutions and tools found helped to collect the requirements and gave hint when selecting suitable technologies for the implementation. There is some promising activity in this area, like End-to-end Diagnostics System (EDDY) [3],[18] developed by Internet2, which may offer similar functionalities in the near future. As more and more federations evolve to production use, the need for monitoring and reporting solutions is likely to rise and result in new applications in this area.

4.4.1 EDDY

EDDY project [3] aims on helping diagnosing problems that are hard and complex due to their distributed nature. Many services are dependent from may lower-level services and one of them misbehaving can cause errors that are difficult to diagnose.

EDDY tries to unify this diagnostic data by defining a common event format (CER) and sharing diagnostic data between domains. A log from single application may

not be enough to find out the real culprit of the problem. Combining logs of different services to get the full picture may be helpful.

It was considered if the monitoring tool should support EDDY format. At the time of the requirement specification and design, EDDY was in quite early stage but seemed reasonably complex system. It was decided not to go this way, but if EDDY becomes popular adding support for it might be considered.

4.4.2 eduroam monitoring

The eduroam infrastructure suffers from similar difficulties what comes to monitoring. Eduroam architecture is equally distributed across organisations and countries, making diagnostics hard. The monitoring solution discussed in [25] is pretty much similar than the one described in this paper. The work for both projects has been done under GEANT2/JRA5 project so the ideas could be shared and tested for both monitoring purpose. The eduroam monitoring has been in production use since August 2008 and is found from [2].

4.4.3 Nagios

Nagios is an open-source monitoring framework that can be used to monitor systems in different levels. Often it is used to run simple tests, like probing the connection to some host, or asking how much disk space is left on the server. This works by using plug-ins, which are run by the Nagios process to report their status.

Nagios also offers a simple interface for application developers to write their own plug-ins to satisfy their monitoring needs. This interface makes it possible to add support for new monitoring targets, AAI monitoring as one example.

4.4.4 Summary

Although many approaches were reviewed for this monitoring purpose, none of them seemed exactly match the requirements. This means that some new code had to be written. A general monitoring application like Nagios or OpenNMS or their commercial alternatives is probably used in most of the bigger organisations, so it is important that the monitoring solution to developed would work together with them rather than being a separated application. Since Nagios is being widely used in organisations using Shibboleth at least within Haka, it was a requirement that

the monitoring part has to be implemented as a Nagios plug-in. Support other monitoring applications should be easy to implement later if needed.

Chapter 5

Design and Implementation

5.1 Design Goals

A design principle was to keep the functionality separated in different components, to make it possible to change one component without much effort. This will make it easier to add support for different AAI implementations and develop new functionality for the application. The application to be developed was named as AAIEye.

5.2 Architecture

Figure 5.1 shows the main components of the monitoring and reporting system, the arrows showing the dependencies between components. The heart of the system is monitoring server. Its responsibilities are:

- Perform the active monitoring test cases.
- Receive and store the usage statistics data.
- Respond to the queries about monitoring status.

The usage statistics and monitoring test results can be browsed with AAIEye Statistics Viewer application. It is a PHP script that reads its input data from SQL database, where it gets filled by AAIEye monitoring server.

To offer support for Nagios, a plug-in is implemented. The plug-in is a small Perl application that Nagios calls periodically to provide status information about active monitoring test cases. The plug-in uses a generic interface for Nagios plug-ins.

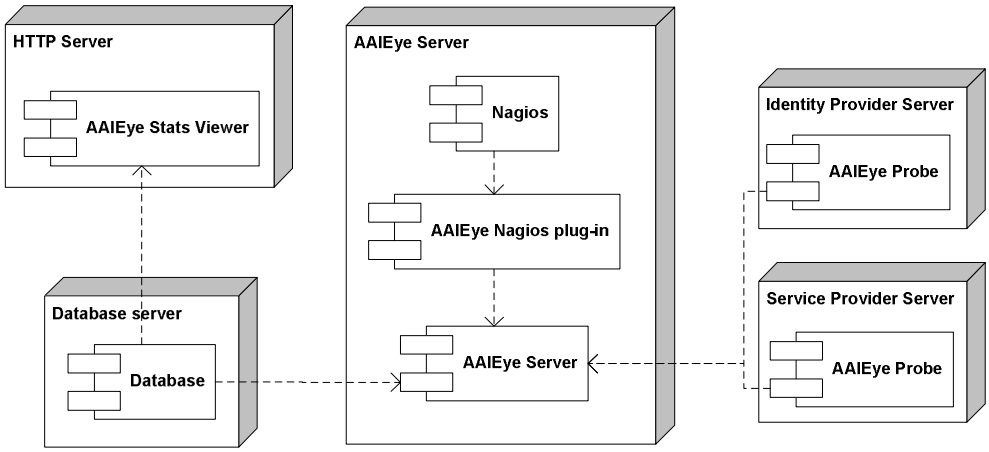

Figure 5.1: AAIEye Component Diagram

Other important components are probes, which are small applications designed to run on the Service/Identity Provider host. They communicate with monitoring server, sending status data about the Service/Identity Provider, which is described later in this chapter.

5.2.1 Monitoring Server

The AAIEye monitoring server is centralised data collection point in the monitoring system. It is responsible for running the active monitoring login tests, collecting the test results and receiving and storing the data that is obtained from the probes. A monitoring server may monitor the whole federation, a part of it or even multiple federations depending on configuration. It is possible to run multiple monitoring servers - for example, an organisation might run a monitoring server for its internal federation, and the other federations it is part of might run their own monitoring servers.

The monitoring server is implemented in Java, to allow support for multiple platforms and to have benefit from Java's extensive libraries for XML and unit testing.

Using Unit Testing For Monitoring

The distributed nature of the AAI makes it challenging to get reliable information about the availability of the services. Typically, an Identity Provider installation will depend on many other services - the Identity Provider application, a HTTP server, a directory or a database server are the most common ones. A Service Provider has

likewise dependencies.

One approach would be to monitor the availability of individual services, like the response codes from HTTP server or from the database. Many organisations actually do already this. From the experiences as the Haka federation operator it can be seen, that this is not enough – the problems are often related to configuration mistakes or changes, or the federation metadata update process – problems that currently can not be easily detected. The providers of the federation are hosted by many different organisations so monitoring them is more complex due to firewalls and security policies. Knowing the exact point of failure is sometimes hard because of these organisation borders – the diagnostics takes time and requires an experienced administrator.

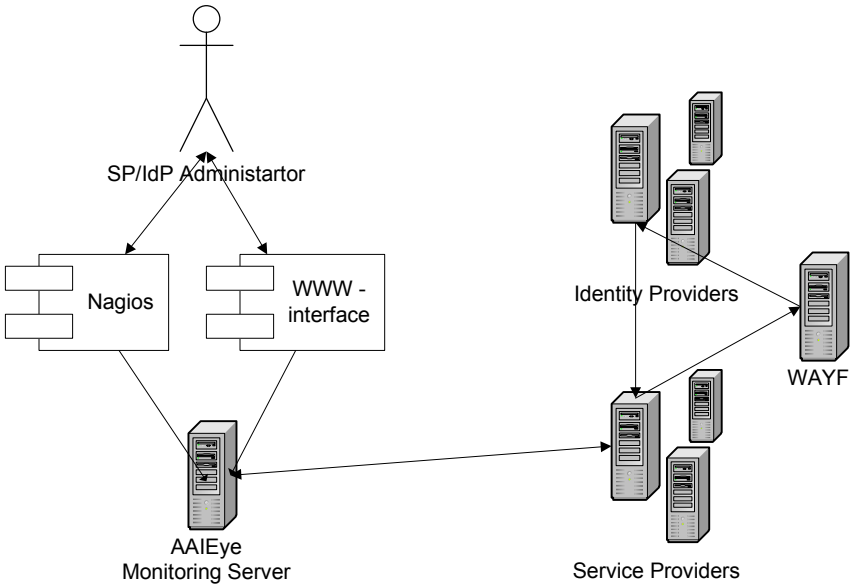

Figure 5.2: Monitoring Architecture

Because of these limitations, the AAI monitoring is designed to be done from the end user point of view. A unit testing approach was chosen, using JUnit [5]- and HtmlUnit -based JWebUnit [6] -framework. For monitoring the availability of a Service Provider, a test case is to be created. The test case should describe the required steps for an end user to log in to the Service. The success or failure is detected by searching for a text string that should appear on the HTML page that the end user sees after successful login.

The JWebUnit is a testing framework for web application development. It has a high level application programming interface (API) to support easy testing for web

site correctness. It is used by defining a test case, which describes the initial setup, the test procedure and the expected result. This approach fits well for monitoring the federated login – the user credentials and service URL are set on the initial setup, the login phase is described as the test procedure and the expected result is successful login on the target web site. The JWebUnit framework handles many details of this invisibly for the monitoring application, for example, HTML parsing and HTTP redirects. This means that using JWebUnit for monitoring is possible without writing much new code.

Configuring the test cases is done with XML based configuration files, as seen on **Listing** 5.3. This example shows a simple case of federated login. What needs to be configured is the details for Identity Provider – the user name, password and the Identity Provider to be used, the actual login flow and the expected result. In this example, a Shibboleth Identity Provider "aitta2.funet.fi" is used with test account "username". The Service Provider is using simpleSAMLphp software, and we expect to find string "Welcome username" in case of successful login.

```
<IdentityProvider id="aitta2IdP" name="aitta2.funet.fi">
  <Attribute name="j_username" type="text" value="username"/>
  <Attribute name="j_password" type="text" value="password"/>
</IdentityProvider>

<ServiceProvider id="maneesiSP"
  name="SimpleSAMLPHP test server"
  startURL="http://maneesi.funet.fi/saml2-example.php">
  <Page submit="" />
  <Page type="IdP" />
  <Page submit="" />
</ServiceProvider>

<Test federation="Haka" name="maneesiSP/Aitta2IdP">
  <ServiceProvider ref="maneesiSP" />
  <IdentityProvider ref="aitta2IdP" />
  <AssertResult type="text" value="Welcome username"/>
  <Period>
    <Minutes>15</Minutes>
  </Period>
</Test>
```

Listing 5.3: Example of configuring active monitoring

5.2.2 Probes

The probes are small applications that are constantly running on the AAI components, the Identity Provider host or the Service Provider host. Three probes are implemented; a usage statistics probe for both Shibboleth Service Provider and Shibboleth Identity Provider, and the metadata checking probe, which can be run on either Service Provider or Identity Provider.

The interface for the probes has to be designed with extensibility in mind. Adding support for different AAI software will require writing a custom probe for them that complies with the probe interface. For this reason, the interfaces is kept small and simple as possible. The configuration of the probes will include support for dynamic class loading, to support implementation of the probes more easily and independently from other components.

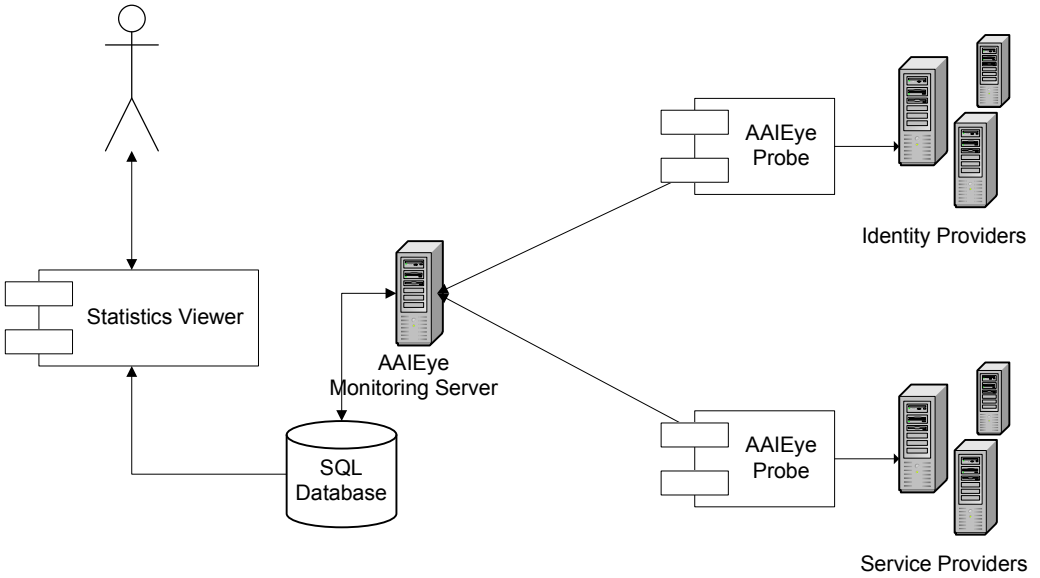

Figure 5.4: Probe Architecture

The Java code example in **Listing** 5.5 describes the Probe interface.

Usage statistics

During their normal operation the Service Providers keep a log about the most important events. In case of Shibboleth, this includes at least new sessions, when users that have authenticated themselves connect to the resources. The Usage statistics probe works by periodically scanning the log files of Service Provider, and it looks

```
public interface Probe {
    /** Sends the results of the Probe to the Monitor Server */
    public void sendResults();
    /** Runs the Probe. */
    public String run();
    /** Initialises the probe.
     * @param p
     * @param runner */
    public void init(ProbeRunner runner, ProbeType p);
}
```

Listing 5.5: AAIEye Interface Probe

for the details of user logins. Shibboleth keeps track of the IP address, session id, the Identity Provider that was used and the application that was accessed. The probe then sends this usage data to the monitoring server.

For persistent storing of probe data, a data storage has to be implemented. A relational SQL database is a common solution, and it is used for storing the log events. A support for MySQL database is implemented, but it is relatively easy to add support for other database engines as well.

Though it is the monitoring server's responsibility to check the timestamps of the log events, the usage statistics probe will also keep track of the latest successful event transmission, to reduce unnecessary network traffic. The XML format is rather verbose, and the much data will be transferred if there are lots of login attempts on a provider.

The log files on Service Provider may contain personal information, as Shibboleth logs the values of the attributes when logging level is high. To comply with data protection principle, no personal information is transferred – all the processing will be done on the Service Provider and the resulting data will not contain anything sensitive. The following data is transferred:

- Entity ID of the Service Provider Probe.
- Identity Provider that performed the authentication.
- Time of the event.
- Event type; successful login or error.

An example of typical data sent by Service Provider Probe is shown in **Listing** 5.6.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<AAIEyeProbeData>
  <Sender entityId="https://aitta.funet.fi/" />
  <IdP entityId="urn:mace:funet.fi:haka:csc.fi:idp">
    <Application applicationId="default">
      <Event type="login">
        <Time>2007-01-17T14:42:31.000</Time>
      </Event>
    </Application>
  </IdP>
  <IdP entityId="aitta.funet.fi:8443">
    <Application applicationId="default">
      <Event type="login">
        <Time>2007-01-16T05:15:34.000</Time>
      </Event>
      <Event type="login">
        <Time>2007-01-16T05:21:34.000</Time>
      </Event>
  </IdP>
</AAIEyeProbeData>
```

Listing 5.6: AAIEye Probe Data Example

There is also a probe for parsing the Identity Provider logs. This probe does the similar task than the Service Provider probe described using the Identity Provider's audit logs. Naturally, it would be unnecessary to deploy probes both on Identity and Service Providers, choosing either is enough for complete statistics. But it was expected that not every organisation is willing to participate by installing a probe, so both versions are implemented.

Metadata checking

As specified in the requirements specification, a probe for analysing the freshness of metadata files was to be developed. The probe is to be installed on the Provider to be monitored, where it polls the metadata file the provider is using. Using Adler32 algorithm, the probe calculates a checksum of the file's contents. The checksum is then sent to the monitoring host.

The monitoring server receives the checksums from various probes. It has a reference metadata file URL configured, which it polls regularly and compares the checksum to those sent by the probes. From this information, the monitoring server determinates

if the providers are using a current version of the metadata or not.

Metadata checking probe is so general functionality, that it is no way limited to any particular AAI implementation. In fact, it could be used for completely different purposes than AAI monitoring.

5.2.3 Communication between Probes and the Server

The probes will communicate with the server using XML over HTTPS. For this purpose, a limited HTTP server functionality has to be implemented for the monitoring server. It will listen to probe connections, do authentication and authorisation decisions, data validation and pass the data to the storage component.

Authentication is based on server certificates. Since both Shibboleth Identity and Service Providers will use server certificates, they are easy to use for other purposes also. The monitoring server is configured with access control list, defining the hostname and the entity IDs for the probes that are allowed to connect. A Java keystore is used for storing certificates both on server and on probes. The keystore must contain the certificates of the probes/server. For each incoming connection, the certificate is checked. If it is valid, the server will check that the probe is using a entity ID that is allowed for its hostname. This prevents accidental and misuse cases, where usage data would be stored for wrong service.

Figure 5.7 describes the communication protocol between Probes and the monitoring server. A Probe is configured to send its results periodically. It will open a HTTPS connection to the monitoring server. In case of communication failure, it will remember its state and will try to send the data again when next connection is scheduled:

- A TLS connection is opened by the Probe. In this phase, the authorisation is done using SSL certificates with client authentication.
- The server validates the probe data. If everything goes well and the data is processed normally, it will return a HTTP 200 OK -response code. In other cases, it will return a HTTP error code. If the server does not respond at all, the connection will terminate on timeout.
- In case of error response, the probe will try to resend its results when the next transmission is scheduled.

- If the server can now successfully handle the data, it will now return a HTTP 200 OK response. In other cases, steps 3 and 4 are repeated.

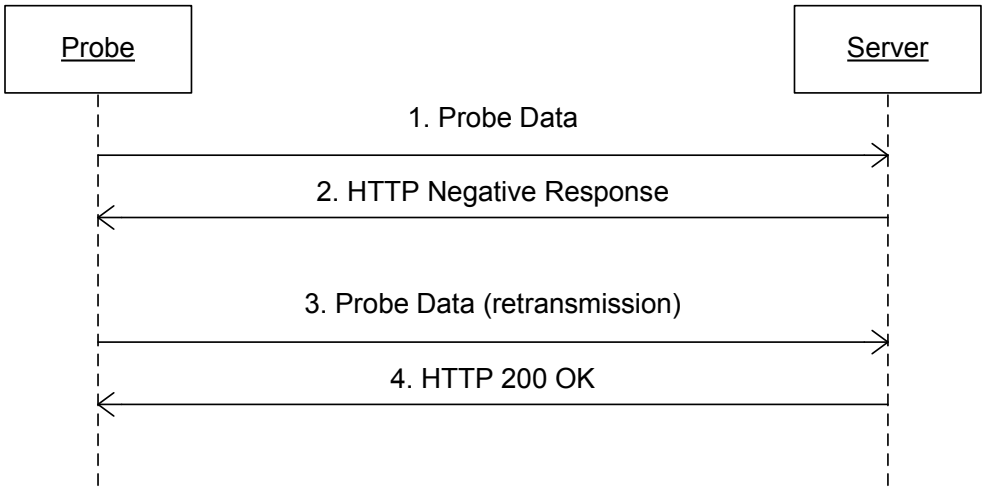

Figure 5.7: Probe Data protocol

5.2.4 User Interfaces

A user interface for viewing monitoring results and use statistics is needed – there are various use cases for both of them. Potential users for such information could be:

- Service Provider administrators.
- Identity Provider administrators
- Federation operator
- End users of the federation's services
- Management of the Providers

The requirements for the user interface is different between these user groups. The user interfaces are designed to be modular, so that the different user groups can get relevant information in an easy and secure way.

Monitoring UI

For viewing the monitoring data two use cases can be found. First, the administrators need to find out if their services are healthy and manage the way their systems are monitored. Secondly, a simple UI is provided for the end users to quickly see if there is some problems with the services they are trying to use.

To serve the system administrators, a Nagios plug-in for monitoring data is implemented. The plug-in is designed to be run on the same host running the monitoring server, and communicates with it using TCP socket. The Nagios then connects to the plug-in using NRPE. The exchanged data between the plug-in and Nagios includes the status of the service, the execution time of the latest test case and the time when the test case was last run. Using Nagios with AAIEye makes it possible to use advanced alert functionality, service break notifications and management functions that Nagios offers.

Figure 5.8 gives an example of Nagios's web interface when using AAIEye and Nagios plug-in for AAIEye.

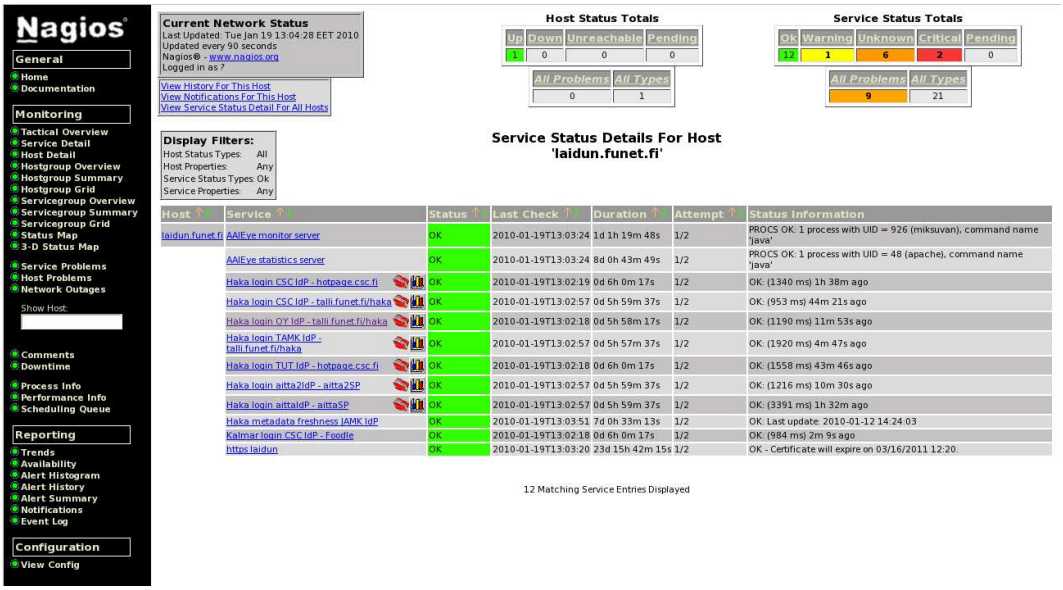

Figure 5.8: Nagios with AAIEye plug-in

For the end users and the administrators not using Nagios integration there is a simple PHP-based web user interface. This UI offers only the status information of the single monitoring test cases. The information is retrieved from AAIEye Monitoring server's SQL database.

Usage Statistics UI

The usage statistics collected by the monitoring server are visualised with a WWW application showing the statistics graphically. The application will get its data from the database of the monitoring server. The user interface has options for showing the usage of Service Providers and Identity Providers which are sending their statistics. This data can be shown in several ways – for example, monthly statistics of which Identity Providers were used with a specific Service Provider, or login counts per day, or error counts per day can be seen. This information can be used to predict how much the services are used, and even to detect some problems. If the usage of a busy service suddenly drops to zero this may indicate some problem with the service.

Figure 5.9 shows how statistics look like. The services are identified by their SAML entity ID, which in turn is mapped to more human-friendly name. The number of failed logins is shown in this graph as red.

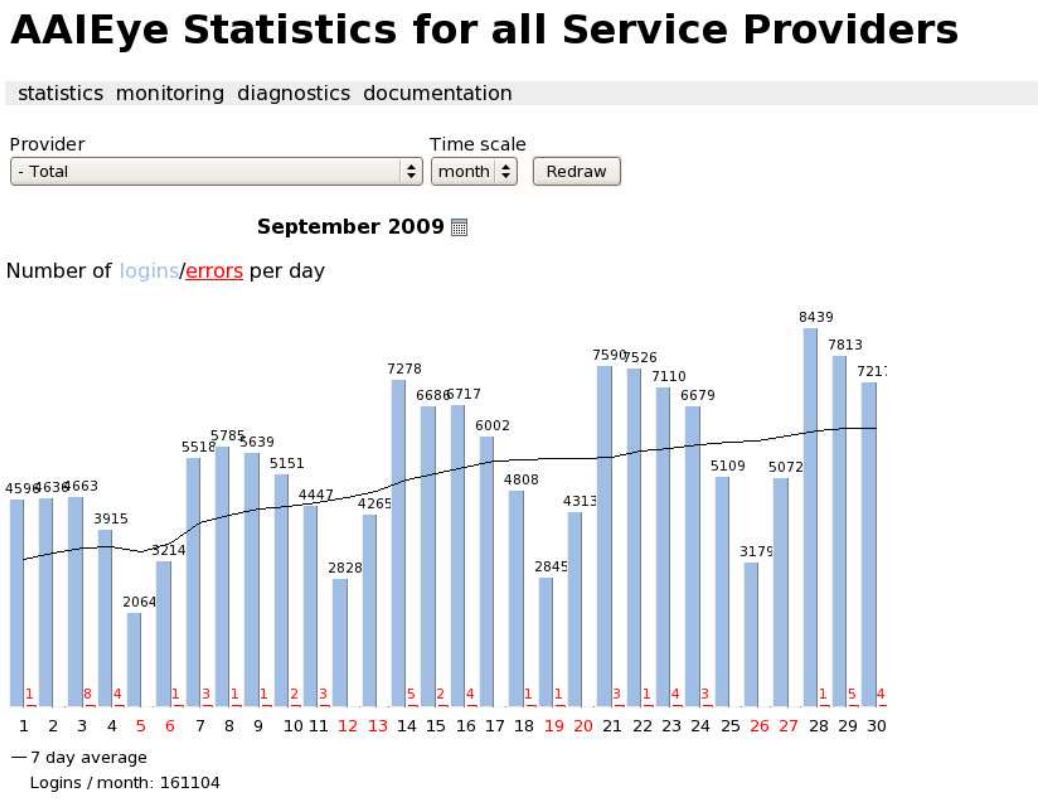

Figure 5.9: AAIEye Statistics

5.3 Deployment

During the development process, the testing was done using two hosts which were both running Shibboleth Service Provider and a Shibboleth Identity Provider software. This test and development setup was quite limited for simulating the behavior when used in whole federation.

Soon after some initial tests the probe was installed on one production level Service Provider. This was needed to get more experience on the performance and reliability of the code, since the usage of the test systems was very low. And to get even more real-world experience, a testing phase was extended to include two production level Identity Providers soon after this phase. Separate test accounts were created on Identity Providers which were used in active monitoring. These Identity Providers did not install Probes, but participated only in active monitoring.

The monitoring software and service were introduced to Haka federation system administrators in annual meetings. At this point, the source code was also released to general public for anyone interested to give it a try. During the Shibboleth training sessions which were arranged by CSC, a brief introduction and encouragement for deploying AAIEye was also given.

For installing the probe software a little Java and XML experience is beneficial. The probe software is distributed as a source code format including an Ant build script which does the actual compiling and also some preconfiguration. The probe must then be configured to match the Service/Identity Provider where it is to be used. This includes setting up the paths to the log files, the entity ID / name of the Provider and the expected attributes.

Chapter 6

Results

6.1 Status of the Work

The requirements were turned into implementation and this work produced a working software which got the name AAIEye. The software is licensed under GPL and available as source code at [15] for free. AAIEye monitoring has been offered as a free service for Haka federation members and partners since 2007. The work was also presented in Terena Networking Conference (TNC) in May 2008 and has gained some interest from other federations too, most importantly within Kalmar Union, as some of its Providers are monitored with CSC's AAIEye service too.

CSC has promoted the monitoring solution for Haka system administrators. They are encouraged to join the monitoring and statistics collection, but it is completely voluntary. **Table** 6.1 gives a summary of how much AAIEye is currently used in Haka. The numbers of Identity/Service Providers is from 22 January 2010. [16]

	Active Monitoring	Statistics Collection
Identity Providers	5 / 39 (13 %)	4 / 39 (10 %)
Service Providers	5 / 78 (6 %)	18 / 78 (23 %)

Table 6.1: AAIEye usage in Haka

As the table shows, most of the services have not adapted into using monitoring or statistics collection yet. Some Identity/Service Providers provide their usage statistics manually to CSC, by parsing the log files manually, but for some reason have not installed AAIEye Probe for this.

There are various reasons that delay the deployment process. Not all services are considered that important that monitoring is needed – an end user complaining on service failures might be enough for them. Many service administrators are also too busy to install and maintain anything else than absolutely necessary. Other reasons may include:

- It is not possible to create a test user account on IdP just for monitoring purpose.
- The service is so rarely used that monitoring and statistics are not interesting.
- The administrator does not have enough knowledge or interest in installing AAIEye Probe.
- The Probe implementation in Java is not suitable for the server environment.
- The availability data and/or usage statistics are considered as classified information.

Getting complete usage statistics would be useful for marketing and developing the federation, but for single IdP or SP administrator the benefits are not very clear. Some kind of reward or requirement is probably needed to increase the number of participants. For example, the federation contract could include a statement requiring that members submit usage statistics in a way or another.

IdP and SP administrators and federation operator are the main users of the monitoring service. In this service, it is not so important to get all connections monitored – the system administrators probably know which services should be monitored. The federation operator can benefit from the monitoring service as it generates alerts if the operator has made a mistake, the metadata is incorrect or the WAYF service is unavailable. For this purpose having just a few test cases is enough.

Other options to increase the monitoring and statistics service usage have been considered. The AAIEye software is introduced in the training sessions that CSC organises for the Haka federation administrators. In those sessions it is possible to get personal help on installing and configuring the software. When new services are registered to Haka federation, the registration process asks also if the administrator wants to take part of the monitoring.

6.1.1 Functionality

The two primary functional requirement were ability to monitor the state of AAI and gather usage statistics from Shibboleth Identity/Service Providers. In this work, the software can fulfill these requirements. The monitoring does not produce much false alarms either. An active monitoring setup is also highly independent on AAI components used in a federation – a demonstration of this is monitoring Kalmar Union connectivity with AAIEye which does not require code changes. In this particular monitoring test, a Norwegian Service Provider using simpleSAMLphp software is monitored from Finnish Identity Provider using Shibboleth software.

The architecture of Probes is extensible enough so it is straightforward to add support for gathering statistics from different AAI software too. The Probes include instructions for doing this and since they are available as source code, administrators can even do this by themselves. The support for Shibboleth Identity Provider version 2.0 was added during the implementation process as a demonstration of this.

The requirement of partial use and easy separation of monitoring and statistics functionality is met. At CSC there are currently two instances of monitoring server running, one for active monitoring and the other for collecting statistics. Nagios support is also in use and there is also an option to integrate AAIEye to customer's own Nagios installation. This option is currently used in one organisation.

Monitoring the metadata freshness is working, but has not seen much use. The active monitoring test cases already check more reliably that the services are up and running. The Shibboleth software starting from version 2.0 supports loading the federation metadata dynamically using HTTP/HTTPS. Using this feature is recommended and it has probably decreased the problems with using outdated metadata.

6.1.2 Security and Privacy

An authentication and authorisation infrastructure is all about security, and by monitoring it we wanted to improve it, not decrease. A monitoring solution must offer high enough security, otherwise it is of little use. The security requirements that were previously defined are however met in this work. By using an end-user point of view, the monitoring application uses the infrastructure in a way it was designed – no additional back doors were opened and no new features to the core AAI software was implemented.

The main security weakness in this solution is probably same as in federated identity management in general – the username/password combination. For active monitoring, the usernames and passwords must be stored in the monitoring service where they are protected, but obviously storing those is always a risk. This risk is further reduced by creating such monitoring accounts that have no access to other services than those needed for monitoring purpose. Deploying other authentication mechanisms like X.509 certificate authentication on the Identity Providers can also be used to reduce this risk.

The privacy requirements were high for this kind of tool, so it limited the detail of information that this software processes. The Probes are designed in a way that no personal information is ever sent outside. The statistics details stored in the central server are not sensitive. Access to the availability information and to monitoring statistics can be easily restricted to the level find necessary.

6.1.3 Performance

The active monitoring has generally proved itself a light-weight operation and has not caused any notable performance problem. This depends on how often the monitoring is performed – the Haka federation monitoring is done in 15-120 minute interval at the moment. What comes into monitoring server performance this interval could easily shortened significantly, but the load to the Identity Provider and Service Provider must also be considered if doing this. Still, a busy Identity Provider in Haka currently handles hundreds of authentication events per hour so even intensive monitoring should not cause much harm for them.

Generating the XML data that the Probe sends to the server is memory intensive operation. If the log files are big, this operation may use more memory than is available for Java virtual machine by default. This problem can be handled by increasing the memory settings. Other option is to alter the logging configuration so that generated log files are smaller.

6.2 Future Work

The software developed is functional and usable in a way it was designed, however, there has naturally appeared new use cases and improvement ideas as the software has been used. Some of them have already been implemented, but there are still many which are just ideas. This section discuss some of those areas of future work.

Many www-sites are using functionality that depends on the user's browser, Javascript being one of the most common technology used. These kind of sites turned out to be challenging to monitor by automated client – the Javascript support of the underlying library was not always complete enough. This made it impossible to monitor all of the services. Another major problem with the Javascript support was that the library was leaking memory, which rapidly caused problems as the tests were being run time after time.

The HTML code used in web is not always perfect, in fact, most of the sites have pages that are broken or use non-standard HTML. This makes the life of the automated tests harder. Writing a monitoring configuration for this kind of web applications is also difficult. Generally improving and easing the configuration for active monitoring would be useful, perhaps even some kind of tool or GUI for doing this could be considered.

The statistics are currently transferred with perhaps too much detail. To reduce network traffic and monitoring server load, more processing could be done in the Probes to calculate some aggregated sums of login events and sending those instead of details of every single event. This would naturally lose some information, but for busy services this option would be useful.

It is worthwhile to follow closely what happens on the field of monitoring. The Nagios integration has been very useful feature for AAIEye and there could be other similar systems that would benefit from integration. For presenting the usage statistics a common library and user interface could be developed or integrated, since there are probably many similar statistics producing tools, the eduroam monitor as one example.

There are now many federations and they are growing. The existing federations are being connected together, and this creates more difficulties and failure points. These confederations could benefit from monitoring solution, and a system described here would adapt quite easily to different AAI software since it is dependent only on external WWW user interface. More options for confederation monitoring would useful to be able to fully separate confederation problems and helping to diagnose them, and presenting the results in a proper way.

Chapter 7

Conclusions

The project of developing monitoring and reporting system has now reached its goals as described in previous chapter. Main requirements were fulfilled and the software has now been in real-life use for some time. Deploying the new service into wider use has been a slow process, like it was expected at the start of the project. The nature of the tool is challenging as many busy system administrators may see this kind of tool purely optional, and more urgent projects are priorised over deploying it.

The monitoring part of the system is not yet widely used in Haka. It monitors some connections in Haka, but majority of Identity and Service Providers are not monitored at the moment. However, on systems where there are constantly changes, the monitoring application has proved its value. For example, CSC internal test services have been monitored for a long time now and automated alerts when a system administrator makes a mistake and services are down has been a real benefit.

The statistics are probably considered more important by the system administrators; the statistics probe has seen more real-life action than the monitoring part of the application. The operator of the Haka federation has been asking for usage statistics to be parsed manually from the log files which involves manual work. Avoiding this work has probably been motivation for the administrators to install our automated solution. There has not been such driving force on the monitoring tool which may explain the difference in their deployment numbers.

The support for different AAI systems and even confederations was considered important. The eduGAIN infrastructure is not in use in Finland at least yet. Thus, the monitoring of eduGAIN has not been tested. Instead, as Kalmar Union went

into production, the monitoring has been tested with some of its providers. This example demonstrates how the monitoring tool can adapt easily enough to changes in the underlying software.

Overall, the project has produced a working solution that quite accurately meets its requirements. The limitations and problems of the solution were found out early while gathering the requirements and designing the software. As the result, no real surprises were seen. The software has been producing information to justify the needs of AAI systems and this way it has helped operating and marketing the Haka federation. Many new improvement ideas have been found, especially when considering the confederation monitoring which is something new in this area. The monitoring and statistics are perhaps even more valuable when system complexity increases which means that further study and development in this area is likely to be seen.

Appendix A

List of Illustrations

2.1 Software and hardware layers in distributed systems.

2.2 Federated Identity.

2.3 Single Logout Problem.

2.5 The interrelationship among federation standards.

2.6 Shibboleth login sequence diagram.

2.7 Generating attributes.

2.9 The eduroam infrastructure.

2.8 The eduroam RADIUS hierarchy.

3.1 A confederation.

3.2 The eduGAIN login procedure.

5.1 AAIEye Component Diagram.

5.2 Active monitoring architecture.

5.4 Probe architecture.

5.7 Probe data protocol.

5.8 Nagios with AAIEye plug-in.

5.9 AAIEye statistics.

Bibliography

[1] *eduroam*. http://eduroam.org, referenced 22 January 2010.

[2] *eduroam monitoring*. http://monitor.eduroam.org/, referenced 22 January 2010.

[3] *End-to-end Diagnostic Discovery*. http://www.cmu.edu/eddy, referenced 22 January 2010.

[4] *FEIDE*. http://www.feide.no, referenced 22 January 2010.

[5] *JUnit*. http://www.junit.org, referenced 22 January 2010.

[6] Jwebunit. http://jwebunit.sourceforge.net, referenced at 22 January 2010.

[7] *Kalmar Union*. http://www.kalmar2.org, referenced 22 January 2010.

[8] *Nagios*. http://nagios.org, referenced at 22 January 2010.

[9] *SCHAC schema*. Available at http://www.terena.org/activities/tf-emc2/schac.html, referenced 22 January 2010.

[10] Ross Anderson. *Security Engineering*. John Wiley & Sons, Inc., 2001.

[11] Scott Cantor. *Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS, 2005.

[12] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler (editors). *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, 2008. At http://www.oasis-open.org/committees/download.php/11898/saml-core-2.0-os.pdf.

[13] David Chadwick, George Beitis, and Gareth Owen. Adding authorisation to eduroam. In *TERENA Networking Conference*, Brugge, Belgium, May 2008.

[14] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems – concepts and design*. Addison Wesley, 2004.

[15] CSC – IT Center for Science. *AAIEye monitoring system*. `http://www.csc.fi/english/institutions/haka/technology/aaieye`, referenced 22 January 2010.

[16] CSC – IT Center for Science. *Haka federation metadata*. `http://haka.funet.fi/fed/haka-metadata.xml`, referenced 22 January 2010.

[17] Haka federation CSC IT Center for Science. *funetEduPersonSchema version 2.1*, August 2008.

[18] Chas DiFatta and Mark Poepping. The case for comprehensive diagnostics. Technical report, Carnegie Mellon University, 2005. `http://www.cmu.edu/eddy/docs/Diagnostics%20Motivation%20Whitepaper0.93.pdf`.

[19] Donald Eastlake, Joseph Reagle, David Solo, Frederick Hirsch, Thomas Roessler (editors), Mark Bartel, John Boyer, Barb Fox, Brian LaMacchia, and Ed Simon. *XML Signature Syntax and Processing (Second Edition)*, June 2008. At `http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/`.

[20] Makoto Hatakeyama. Federation proxy for cross domain identity federation. In *Conference on Computer and Communications Security Proceedings of the 5th ACM workshop on Digital identity management*, pages 53–62, 2009.

[21] Internet2. *Shibboleth*. `http://shibboleth.internet2.edu`, referenced 22 April 2010.

[22] Mikael Linden. Organising federated identity in Finnish higher education. *Computational Methods in Science and Technology*, 11(2):109–117, 2005.

[23] Mikael Linden, David Simonsen, Andreas Åkre Solberg, Ingrid Melve, and Walter M. Tveter. Kalmar union, a confederation of Nordic identity federations. In *TERENA Networking Conference*, Malaga, Spain, June 2009.

[24] Diego Lopez, R. Castro, B. Kerver, T. Lenggenhager, M. Linden, I. Melve, M. Milinovic, J. Rauschenbach, M. Stanica, K. Wierenga, S. Winter, and H. Ziemek. *GEANT2 Authorisation and Authentication Infrastructure (AAI) Architecture – second edition*, 2007. Available at `http://www.geant2.net/upload/pdf/GN2-07-024-DJ5-2-2-2-GEANT2_AAI_Architecture_And_Design.pdf`.

[25] Miroslav Milinovic, Dubravko Penezic, and Ian Thomson. *Report on Introduction of Monitoring System and Diagnostics Tools*, 2009. Available at `http://www.eduroam.org/downloads/docs/GN2-09-008v2-DS5-3-1-Report-on-Introduction-of-Monitoring_System_and_Diagnostics_Tools_Final.pdf`.

[26] Nick Ragouzis, John Hughes, Rob Philpott, Eve Maler, Paul Madsen, and Tom Scavo (editors). *Security Assertion Markup Language (SAML) V2.0 Technical Overview*. OASIS, 2008.

[27] Patrik Schnellmann, Patrick Chénais, and André Redard. Enhancing SWITCHaai with micropayment functionality for Swiss universities, 2006.

[28] Walter M. Tveter, Ingrid Melve, and Mikael Linden. Towards interconnecting the Nordic identity federations. *Campus-Wide Information Systems*, 24:252–259, 2007.

[29] Rod Widdowson and Scott Cantor (editors). *Identity Provider Discovery Service Protocol and Profile*, 2008. Committee Specification 01.

[30] Philip J. Windley. *Digital Identity*. O'Reilly, 2005.

[31] S. Winter, T. Kersting, P. Dekkers, L. Guido, S. Papageorgiou, J. Mohacsi, R. Papez, M. Milinovic, D. Penevic, J. Rauschenbach, J. Tomasek, K. Wierenga, T. Wolniewicz, José-Manuel Macias-Luna, and I. Thomson. *Inter-NREN Roaming Infrastructure and Service Support Cookbook - Third Edition*, 2008. Available at `http://www.eduroam.org/downloads/docs/GN2-08-230-DJ5.1.5.3-eduroamCookbook.pdf`.

[32] Andreas Åkre Solberg, Scott Cantor, Eve Maler, and Leif Johansson. *Interoperable SAML 2.0 Web Browser SSO Deployment Profile version 0.1*, November 2009. At `http://saml2int.org/profile/0.1`.