Mid Sweden University
The Department of Information Technology and Media (ITM)
Author: Carl Wikblom
E-mail address: cawi0701@student.miun.se
Study programme: M. Sc. in engineering – computer engineering, 300 ECTS
Examiner: Tingting Zhang, tingting.zhang@miun.se
Tutors: Tingting Zhang, Mid Sweden University, tingting.zhang@miun.se
Mattias Sjölander, Logica, mattias.sjolander@logica.com
Scope: 14165 words inclusive of appendices
Date: 2012-08-28

M.Sc. Thesis within
Computer Engineering AV
30 ECTS

Federated identity management
AD FS for single sign-on and federated identity management

Carl Wikblom
Abstract

Organizations are continuously expanding their use of computer services. As the number of applications in an organization grows, so does the load on the user management. Registering and unregistering users both from within the organization and also from partner organizations, as well as managing their privileges and providing support all accumulates significant costs for the user management. FIdM is a solution that can centralize user management, allow partner organizations to federate, ease users’ password management, provide SSO functionality and externalize the authentication logic from application development. An FIdM system with two organizations, AD FS and two applications have been deployed. The applications are constructed in .NET, with WIF, and in Java using a custom implementation of WS-Federation. In order to evaluate the system, a functional test and a security analysis have been performed. The result of the functional test shows that the system has been implemented successfully. With the use of AD FS, users from both organizations are able to authenticate within their own organization and are then able to access the applications in the organizations without any repeated authentication. The result of the security analysis shows that the overall security in the system is good. The use of AD FS does not allow anyone to bypass authentication. However, the standard integration of WIF in the .NET application makes it more susceptible to a DoS attack. It has been indicated that FIdM can have positive effects on an organization’s user management, a user’s password management and login procedures, authentication logic in application development, while still maintaining a good level of security.

Keywords: Federated identity management, active directory federation services, windows identity foundation, WS-Federation.
Acknowledgements

I would like to thank Carin Lindström and Mattias Sjölander at Logica for giving me the opportunity to conduct my thesis in cooperation with Logica. Special thanks to my supervisor, Mattias Sjölander, for the quick help and input when needed. I would also like to thank my family and friends, who have supported me during this thesis work and throughout my education.
# Table of Contents

Abstract .................................................................................................................. ii  
Acknowledgements ............................................................................................... iii  
Terminology ............................................................................................................ vii  

1 Introduction ......................................................................................................... 1  
1.1 Background and problem motivation ......................................................... 1  
1.2 Overall aim ................................................................................................. 2  
1.3 Scope ........................................................................................................... 2  
1.4 Concrete and verifiable goals .................................................................... 3  
1.5 Outline ......................................................................................................... 4  

2 Federated identity management ........................................................................... 5  
2.1 Identity information exchange models ...................................................... 5  
2.2 Single sign on ............................................................................................. 8  
2.3 Problems and opportunities ...................................................................... 8  

3 Research in federated identity management ..................................................... 10  
3.1 Keys to success .......................................................................................... 10  
3.2 Security ..................................................................................................... 10  
3.3 Dynamic federations ................................................................................ 11  
3.4 Architecture interoperability .................................................................... 11  
3.5 Federated global identity management ..................................................... 12  
3.6 Identity aggregation .................................................................................. 12  

4 Active Directory Federation Services ............................................................... 14  
4.1 Attribute stores .......................................................................................... 14  
4.2 Security tokens and claims ......................................................................... 14  
4.3 Roles ......................................................................................................... 15  
4.4 Configuring trusted partners ..................................................................... 15  
4.5 WS-Federation, WS-Trust and SAML ....................................................... 17  
4.5.1 XML Signature verification .................................................................. 18  
4.6 Alternative architectures ........................................................................... 19  
4.6.1 PingFederate ....................................................................................... 19  
4.6.2 Shibboleth ........................................................................................... 20  
4.6.3 OpenAM ............................................................................................. 20  

5 Methodology ..................................................................................................... 21
Federated identity management - AD FS for single sign-on and federated identity management

Carl Wikblom

Table of Contents

2012-08-28

5.1 Experimental environment ........................................... 21
5.1.1 .NET and Java applications ...................................... 21
5.1.2 Servers and software resources ................................... 22
5.2 Functional test and security analysis .......................... 22
5.2.1 Threat model and security tests ............................... 23

6 Design ........................................................................... 25
6.1 Requirements analysis and design considerations .......... 25
6.2 Experimental environment ........................................... 26
6.2.1 Configuring AD FS .................................................. 26
6.2.2 Environment issues ................................................ 27
6.3 .NET application ....................................................... 27
6.3.1 Configuring authentication and registering WIF modules ........................................................................ 27
6.3.2 Configuration WIF modules .................................... 28
6.3.3 Request validation .................................................. 29
6.3.4 Deployment ......................................................... 29
6.3.5 Application issues .................................................. 30
6.4 Java application .......................................................... 30
6.4.1 Overview of the WS-Federation design ..................... 30
6.4.2 WS-Federation implementation ............................ 31
6.4.3 Authentication handler implementation .................. 32
6.4.4 Application issues ................................................ 33

7 Security analysis ............................................................. 34
7.1 Security mechanisms .................................................... 34
7.2 Security tests .......................................................... 35
7.2.1 Forging a security token ....................................... 35
7.2.2 Erroneously acquiring a security token ................. 36
7.2.3 Denial of service attacks ....................................... 37
7.3 Security considerations ............................................... 37

8 Results ........................................................................... 39
8.1 Functional test .......................................................... 39
8.2 Security analysis ........................................................ 40

9 Conclusions ................................................................. 42
9.1 Effects of federated identity management .................... 43
9.2 Recommendations for future work .......................... 44
9.3 Recommendations regarding integrating federated identity management into an organization .................. 45

References ........................................................................ 47
## Terminology

### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>AD DS</td>
<td>Active Directory Domain Services</td>
</tr>
<tr>
<td>AD FS</td>
<td>Active Directory Federation Services</td>
</tr>
<tr>
<td>AD LDS</td>
<td>Active Directory Lightweight Directory Services</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASP</td>
<td>Active Server Pages</td>
</tr>
<tr>
<td>B2B</td>
<td>Business-to-business</td>
</tr>
<tr>
<td>CA</td>
<td>Certificate Authority</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSRF</td>
<td>Cross-site Request Forgery</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DoS</td>
<td>Denial-of-service</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition</td>
</tr>
<tr>
<td>FIdM</td>
<td>Federated Identity Management</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hypertext Transfer Protocol Secure</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IdM</td>
<td>Identity Management</td>
</tr>
<tr>
<td>IdSP</td>
<td>Identity Service Provider</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Service</td>
</tr>
<tr>
<td>JDK</td>
<td>Java Development Kit</td>
</tr>
<tr>
<td>MITM</td>
<td>Man-in-the-middle</td>
</tr>
<tr>
<td>PHP</td>
<td>Hypertext Preprocessor</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RP</td>
<td>Relying Party</td>
</tr>
<tr>
<td>RST</td>
<td>Requested Security Token</td>
</tr>
<tr>
<td>SAML</td>
<td>Security Assertion Markup Language</td>
</tr>
<tr>
<td>SID</td>
<td>Security Identifier</td>
</tr>
<tr>
<td>SLO</td>
<td>Single Logout</td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Socket Layer</td>
</tr>
<tr>
<td>SSO</td>
<td>Single Sign-On</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WIF</td>
<td>Windows Identity Foundation</td>
</tr>
<tr>
<td>W-RST</td>
<td>WS-Trust Request Security Token</td>
</tr>
<tr>
<td>W-RSTR</td>
<td>WS-Trust Request Security Token Response</td>
</tr>
<tr>
<td>WS</td>
<td>Web Services</td>
</tr>
<tr>
<td>WSIT</td>
<td>Web Services Interoperability Technologies</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSS</td>
<td>Cross-site Scripting</td>
</tr>
</tbody>
</table>
1 Introduction

The de facto standard for authentication within computer systems at the present time is by using an, often user specific, password. Together with the extensive use of computers, both personally and professionally, this leads to users managing an abundance of passwords which all must be remembered [1]. Regardless of the authentication used, the users in a computer system must be managed. In publically available systems, this might be achieved by the users themselves, but, in corporate systems, this must often be performed by a system administrator. As the number of applications in an organization increases, so does the need for mutual authentication, both for users and system administrators.

1.1 Background and problem motivation

Organizations are continuously expanding their use of computer services, and since they often require authentication there is a requirement to establish a unified architecture that allows a user to authenticate once and subsequently be able to access the appropriate applications and systems.

In most organizations it is necessary manage computer users, controlling their access and privileges for different applications and systems. This also includes registering and unregistering users, changing privileges, and providing support and all this adds up to significant costs [2]. When the number of applications grows, so does the load on the user management. Users must be registered with the appropriate privileges and also unregistered at several places. The likelihood for mistakes to occur increases with the increase in applications, which, in turn, will lead to more support being required. Some mistakes, such as supplying incorrect privileges or failing to completely unregister a user, might even jeopardize the security of the system.

Additionally, users from partner organizations might require access to applications within the organization. This leads to organizations, apart from their own users, also requiring to manage users from partners. The users might change roles, or even quit, without the partners informing the organization of the changes, leaving unauthorized users with the
ability to gain access. Since organizations might manage users on behalf of partners, they also have to provide support for the partners’ users, which is actually the partners’ responsibility.

Users must remember usernames and passwords for many applications and systems. The passwords might also be required to possess a particular complexity and be changed regularly. Many users cannot remember all the passwords, which leads to risky behaviors including writing down passwords and using the same password in many applications. The problems associated with authentication lead to frustration and can reduce productivity for users. [1] [3]

When applications requiring authentication are developed, the developers often build their own authentication functionality. Consequently, functionality for managing the users must also be created. Making all of this secure is difficult, especially for developers who are not usually concerned with security. It is also time-consuming to create this functionally correctly for all applications.

1.2 Overall aim

The overall aim of this thesis is to research different architectures for single sign-on (SSO), to deploy such an environment and evaluate its security, thus obtaining knowledge regarding the effects it might have on an organization’s user management, a user’s password management, application development, and security. It is also likely to reveal limitations, difficulties, benefits, and trade-offs regarding the deployment of an SSO environment.

Hopefully, the results of this thesis will show that federated identity management can reduce organizations’ costs for support and administration by easing their user management. Another desirable outcome is to improve the productivity due to enhanced interoperability, simplified login procedures, and reduced authentication complexity with regards to application development. Finally, the expectation is that it will improve security.

1.3 Scope

Authentication is determining who the user is while authorization is deciding what the user is allowed to do. Although both topics are pre-
sent in federated identity management, the focus of this thesis is on authentication.

Federated identity management can be used in different scenarios, for which different requirements and problems exist. This thesis is focused on a business-to-business scenario.

There are different types of SSO solutions. Some consist of programs that automatically fill authentication forms whilst others share credentials, but, still require multiple logins. This work focuses on solutions for which the user has to authenticate only once and where the credentials are managed by one authority.

Computer security is a vast subject and all systems are different. Hence it is not possible to cover all possible aspects. What will be discussed in this case, is the security of the involved protocols and security risks that come explicitly from utilizing them.

1.4 Concrete and verifiable goals

An approach to dealing with the problem areas involves the use of Active Directory Federation Services (AD FS) to create a corporate SSO solution. The goal is to deploy an environment with two organizations in which they are able to authenticate across domain borders using federated identity management.

The intention of this thesis is that the following results are to be achieved:

- A brief introduction to federated identity management including current research.
- A study of AD FS including a review of alternative architectures and an analysis of setting up trusted partners.
- A virtual environment including two organizations which are able to authenticate between each other using AD FS.
- Applications written in Java and .NET that accept authentication from AD FS.
• A security analysis of the created environment, identifying the risks and vulnerabilities.

1.5 Outline

Chapter 2 provides a brief introduction to FIdM and Chapter 3 presents the current research within the subject. AD FS, together with alternative architectures, are described in Chapter 4. Chapter 5 describes the methodology used to create and evaluate the environment. The design and implementation of the environment and the two applications are presented in Chapter 6. In Chapter 7, the security analysis is summarized. Chapter 8 presents the result obtained when evaluating the environment and Chapter 9 presents the conclusions of the thesis.
2 Federated identity management

An entity is anything in existence that can be uniquely identified [4], but, in relation to this topic, it is most often a user. Characteristic information elements belonging to an entity are called attributes and if the attributes are able to sufficiently identify an entity within a certain context they provide an identity [4]. Briefly, identity management (IdM) is the process of managing identities and their attributes [4]. In this context, a federation is an association of any number of users, service providers and identity service providers and a federated identity is an identity that can be used to access the services of a federation [4] [5]. Putting this together, federated identity management (FIdM), is the process of managing federated identities. With FIdM, users require only one identity in order to access all the services within the federation.

Although FIdM is not a novel idea, during last decade it has matured into a feasible option for cross-domain authentication. History shows that gaining wide-spread acceptance for a single FIdM system is problematic and success has only been achieved within particular contexts [6] [7]. Recognizing the flaws of the old systems, new ones have emerged and only time will tell whether or not they are successful. One reason that the various systems have failed is because they were intended to be used in very broad scope, meaning that they were providing identity for arbitrary services at the Internet.

Many of the problems associated with providing identity in the Internet are not present when partner organizations form a federation. Trust between the organizations is implicit, because they are in a business relationship, and a user trusts his/her organization to handle his/her identity. In this case, it becomes clear as to where a user belongs, where he/she should authenticate and what the identity will be used for.

2.1 Identity information exchange models

The standard method for FIdM involves identity service providers (IdSPs) and relying parties (RPs). An IdSP manages and verifies the identities of other entities and an RP relies on identity representations issued by IdSPs [4]. The identity information is often exchanged as claims (or
assertions). Claim and assertion are used interchangeably and they both imply a statement, about anything, made without any evidence of its legitimacy; the RP trusts that the claims from the IdSP are correct.

![Diagram of identity information exchange](image1)

Figure 1: Basic model for identity information exchange.

In Figure 1, a basic model for exchanging identity information is illustrated. The RP receives a claim from a user concerning the user’s identity, asks the IdSP to validate the claim and, if the claim validates, grants access. This model is currently used by some open IdM systems [8]. One problem associated with this model involves privacy, since the user has no control over the information exchanged between the RP and the IdSP.

Figure 2: User-centric identity information exchange model.

A solution to the privacy concern involves the use of user-centric models that offer the users full control over the use of their identity by directing all identity information exchange through the users [8]. In a user-centric
model, as illustrated in Figure 2, the user retrieves a claim about his/her identity from an IdSP that the RP trusts. That claim will then allow the user to access the RP.

One complication associated with both these models involves scalability when adding IdSPs and RPs; users must either be added to several IdSPs or the RPs must be extended to trust more IdSPs. Figure 3 illustrates a model in which a user is able to use a single IdSP to access different RPs, regardless of whether or not they trust the user’s IdSP. In this model, RP only trusts IdSP2, and the user is only able to authenticate with IdSP1, however, since IdSP2 trusts IdSP1 to authenticate users, the user can be granted access. Neither the user nor the RP are required to be aware of anything other than the IdSP that they trust. Users and RPs can be added to the IdSP that they belong to and new IdSPs only require to be configured at the existing IdSPs. This model is useful for federation in a business relationship since each organization can have an IdSP, which is managing its users and RPs and the IdSPs can build trust relationships with each other, allowing the users to access all the appropriate RPs in the federation.

Figure 3: Model with one IdSP trusting another IdSP to authenticate users.

The last model can also be changed into a user-centric model and all the mentioned models can be extended to include, for example, identity attribute providers to supply attributes in place of the IdSP.
2.2 Single sign on

One evident benefit of FIdM is the possibility of providing single sign on (SSO) capability. When a user is authenticated at an IdSP, he/she is able to access all RPs for which privilege has been provided, without any repeat authentication. The IdSP is able to remember the user’s successful authentication and hence, the user needs not to authenticate again when requesting access to new RPs. This is advantageous to the user who is not required to supply login credentials repeatedly and an organization also has the ability to opt for stronger policies regarding authentication.

A desirable feature in SSO environments is single logout (SLO) [9], which enables users to logout from all active applications simultaneously. However, SLO possesses some problems and a number of the main ones are addressed in this chapter. If a user issues a logout from one application, it must be clear as to whether the user has logged out from the application itself, or all active applications and all applications should behave consistently. In addition, if the user performs an SLO, it should also be possible to verify that all applications have terminated successfully, thus leaving no active applications dangling.

2.3 Problems and opportunities

Properly implemented, FIdM provides several advantages. The users are only required to remember one password and together with SSO it can increase their productivity. Users are also less likely to forget passwords thus lowering the costs involved in resetting passwords. The users are managed in one place, so adding, deleting and resetting users will take less time. This also makes it easier to back up the user database. Organizations in a federation need only manage their own users and can rely on the other organizations to manage theirs. Adding organizations to the federation will be simpler and less time consuming. Additionally, application developers can leave user management to the IdSPs and, instead, focus on developing the application functionality. Overall, FIdM can enhance the security of a system by reducing the probability of mistakes in user management, application development and users’ password management.

A Norwegian study [10] found several benefits associated with FIdM for both users and businesses. Benefits from a user perspective included improved usability, increased privacy protection, and security. From a
business perspective, the benefits included reduced costs, increased security, improved data quality and user management, reduced complexity for RPs, and enhanced business cooperation.

One problem associated with FIdM is that the IdSP can become a single point of failure. If an IdSP is unavailable, users might not be able to access any services and, additionally, no RPs relying on the IdSP can be accessed. Providing backup IdSPs can reduce the problem. Another problem is that a user’s credentials are used to access all services and if they are lost, so is all access. Additionally, an attacker with the credentials will gain access to all services. Therefore, protecting the credentials is even more important. Mutual authentication can assist in protecting users’ credentials as it allows them to correctly identify the IdSPs, and to not become tricked into providing their credentials to a false IdSP [11].

Interoperability can become a problem if organizations use different syntax for attributes [12] or utilize different federation architectures [13].

When deploying FIdM, usability is of great concern. Users are focused on performing their primary tasks and they are not likely to accept the fact that obstacles have been introduced due to IdM, security or privacy. It should be that an FIdM system facilitates their primary task seamlessly and securely. Care must be taken so as not to increase a user’s overall workload when reducing the burden associated with one task. Users follow the path of least resistance and introducing new burdens will only force them to take shortcuts, simultaneously jeopardizing security. [11]
3 Research in federated identity management

FIdM currently receives a significant amount of research attention. This chapter offers a brief review of the research, both to highlight some important characteristics for success and also to provide some hints regarding the future of FIdM.

3.1 Keys to success

S. Landau and T. Moore [6] have analyzed both successful and unsuccessful FIdM systems and have drawn conclusions regarding the underlying reasons. They present four obstacles that must be avoided for a system to achieve success. Namely, it must be clear who collects the user data, sets the authentication rules, and is responsible when authentication fails. Furthermore, all parties must gain something by implementing the system. [6]

After researching success and failure for several identity systems, K. Cameron [7] formed seven laws of identity that an identity system must conform to. Poor privacy is a major reason for failure and the laws he created reflect this. The laws stipulate that a user must consent to all information that is revealed about him/her and that a system must only reveal the information which is strictly necessary. Furthermore, only justified parties must be present in the identity information exchange. A user must also be able to choose from different IdSPs and use different identities according to his/her needs. Lastly, the user must have a simple, consistent experience and be integrated in the system. K. Cameron [7] thinks that it is unlikely that a single FIdM system will become a universal standard and instead proposes an identity metasystem that allows several smaller systems to coexist and cooperate. [7]

3.2 Security

R. Wang, S. Chen, and X. Wang [14], have researched the security of several popular FIdM systems on the Internet, including Google ID and Facebook Connect. Flaws that circumvented the login procedure were found in eight systems and possible vulnerabilities were identified in
others. All systems used web browsers to relay the messages. All vulnerabilities were the result of implementation flaws and not vulnerabilities in the used protocols. One example of a flaw is that unsigned elements were trusted, which allowed a user to simply change the value of the elements in order to gain access as another user. A flaw in another system was that the RP always interpreted the claim it received as an e-mail address. Then, a user could register an e-mail address as a street address and thus cause the IdSP to send the street address to the RP. The RP would still interpret it as an e-mail address and allow access based on it. Several other exploits were possibly because an attacker could register a RP and then trick the IdSP to send it security tokens for other RPs; a user authentication for the fake RP would then give the attacker its security token for another RP. [14]

3.3 Dynamic federations

In traditional federations, trust between the parties is established and configured manually. This might be favoured in some scenarios, but can be limiting in others. With dynamic federations, trust can, instead, be established dynamically. R. Sanchez et al. [15] proposed a protocol for which a federation can be expanded dynamically with the use of reputation requests. Utilizing a dynamic trust list, IdSPs and RPs store information of other parties together with their reputation. When a user wants to authenticate with an (for the RP) unknown IdSP, the RP sends a reputation request to the parties it trusts and, depending on the answers, decides to add the new IdSP or not. A similar approach, also establishing trust by reputation, is suggested by L. Boursas and W. Hommel [16].

An alternative to expanding existing federations is to create a virtual federation, covering two or more federations [17]. This can be performed in a static manner, but also dynamically using reputation.

3.4 Architecture interoperability

Different federations might operate with heterogeneous architectures, thus raising the requirement to achieve interoperability between architectures. M. Ates et al. [13] think that active requestors, that is, requestors that actively take part in the information exchange, will be the future of federation architecture interoperability. An active requestor can trans-
late messages between two protocols and hence, make them operate together. But, recognizing that the translation technique is the same, whether or not the requestor is active, they have implemented a proof of concept third party translation service that can be used to achieve interoperability. The translation service is inserted between the different federation architectures and transforms all messages passing through.

Oxford Computer Group [18] verified that architectures can be setup to understand several protocols. By using several protocols, architectures are able to interoperate with others without the need for message translation by third parties. If the architectures can use the same protocol, achieving interoperability is, although challenging [18], merely a matter of configuration.

Technical interoperability is not the only problem present in heterogeneous environments. Systems using different vocabularies might be unable to interact when attributes are named differently. Recognizing the naming heterogeneity problem, F. Paci et al. [20] propose a protocol that is able to resolve this using lookup tables, dictionaries and ontology mapping techniques.

3.5 Federated global identity management

Trust-ME is a framework for global FIdM proposed by J. Siddiqi et al. [20]. The framework aims to provide secure techniques for globally identifiable users and services. Implementation of the framework would allow a user to traverse different networks while continuously being able to use services requiring authentication. For example, a user authenticated to Trust-ME is able to move from his/her home network, via a mobile network, to a café in which automatic authentication is provided to use the café’s network and is thus seamlessly able to access protected services without having to re-authenticate.

3.6 Identity aggregation

A user’s full identity might be spread throughout several federations with various IdSP’s holding only parts of the user’s identity. When a RP requires information concerning a user from more than one IdSP, the need to combine identity information arises. For example, a credit card company might authenticate the user’s credit card while the government authenticates his/her age.
D. W. Chadwick and G. Inman [21] proposed a linking service to handle identity aggregation. The linking service will group all the IdSPs of a user and allow the user to choose which IdSPs he/she wants to use when authenticating, possibly using attributes from several IdSPs simultaneously. To allow the user to use the linking service without authenticating separately to all IdSPs, one IdSP can be used to authenticate the user to the other IdSPs.

Another solution, organizing all IdSPs and RPs in an overlay network, is proposed by K. Lampropoulos and S. Denazis [22]. With all the IdSPs and RPs in the same overlay network, the users have the ability to choose which IdSPs they would want to hold their information and then to combine the relevant information when authenticating.
4 Active Directory Federation Services

Active Directory Federation Services (AD FS) is a server role in the Microsoft Windows Server (2003 and 2008) operating systems. The current version, and that explained in this chapter, is 2.0. Using FdM, AD FS can be set up to provide cross-organizational access and SSO for an organization’s web applications. AD FS 2.0 supports the Web Service (WS) protocols WS-Trust and WS-Federation, as well as the Security Assertion Markup Language (SAML) standard. Authentication decisions are made with claims and are called claims-based. The main focus of AD FS is to be a federation solution for business-to-business (B2B) relationships. [23]

4.1 Attribute stores

Attribute stores [24] are used to authenticate and retrieve claims about users. There are multiple attribute stores supported by AD FS: Active Directory Domain Services (AD DS), Active Directory Lightweight Directory Services (AD LDS), Structured Query Language (SQL) databases and custom built stores [24]. AD DS [25] is a distributed database, in the form of a directory, which can be used to organize and manage network resources such as users, computers, and other devices in a hierarchical structure. A server running AD DS is called a domain controller. AD LDS [25] provides a great deal of the same functionality as AD DS but does not require the deployment of a domain controller; any computer can host an instance of AD LDS.

4.2 Security tokens and claims

A security token is a set of claims, expressed as a unit and cryptographically signed. AD FS uses X.509 certificates to sign and verify security tokens. The private key, used to sign, is referred to as a token-signing certificate and the public key, used to verify, is called a verification certificate. [26]

AD FS supports any type of claim and is configured using several standard claim types. Apart from several standard claims, such as name, e-mail and user principal name, Microsoft has included Windows specif-
ic claims in AD FS. Examples of those are windows account name, group security identifier (SID) and primary SID. [27]

4.3 Roles

Servers running AD FS can have different roles. The roles are basically IdSP and proxy.

The Federation Service role makes the AD FS act as an IdSP. There are two types of IdSPs in AD FS, account partner and resource partner. An account partner authenticates users against an account store and sends claims about users’ identities to resource partners. The resource partner receives claims from account partners and then redirects the claims to RPs. The AD FS instance can be both an account partner and a resource partner. If AD FS has several account partners, it is also responsible for redirecting users to the appropriate account partner.

As a Federation Service Proxy, the AD FS forwards communication to either an account partner or a resource partner that is, for example, protected by a firewall. It can proxy several types of requests, including WS-Trust and WS-Federation. [28]

In earlier versions of AD FS, RP applications had to be run on an AD FS instance configured as an AD FS Web Agent [29]. However, as of version 2.0, there is no special requirement for any RP except that they can communicate with AD FS in any of AD FS’s supported protocols.

4.4 Configuring trusted partners

In a federation of organizations, some kind of mutual trust exists. Even though the mutual trust ranges from business level agreements to the technical level, the trust of authentication users in an AD FS federation is thought of as one-way [30]. This means that the resource partner trusts the account partner to authenticate users, but not the other way around. Figure 4 illustrates this one-way, trust relationship.

Establishing technical trust and making them interoperate requires the configuration of both the account partner and the resource partner. This includes configuring trust and mapping claims. In order to secure the communication between the two partners, they have to exchange certificates. As the security of the system completely relies on the certificates,
this is a delicate task that must be handled carefully. Exchanging them over the Internet might compromise their security.

Figure 4: Direction of the authentication trust in a federation. [30]

Setting up trust between two AD FS servers is performed by configuring the trust policies of both servers. Each server is required to know the name, the uniform resource identifier (URI), and the uniform resource locator (URL) of the other server. They must also agree on which identity claims to use, and the resource partner must know the verification certificate of the account partner. Furthermore, the account partner can choose to enhance the users' privacy by removing sensitive data in the claims [31]. Lastly, if the account partner can authenticate users from different domains, then the resource partner must decide which domains to trusts. [32] [33]

Figure 5: Claims mapping process. [34]

Organizational claims are claims expressed in an organizations local namespace. Before exchanging claims, the partners must map their
organizational claims into a common set of claims. This process is called *claims mapping* and includes mapping, removing and the filtering of claims. The account partner maps its organizational claims into outgoing claims and the resource partner its incoming claims into organizational claims. See Figure 5 for an illustration of claims mapping. [34]

### 4.5 WS-Federation, WS-Trust and SAML

WS-Federation is a protocol which allows different security realms to federate and exchange identity information. The mechanisms used in AD FS are basically requesting and returning security tokens with the WS-Trust message *Request Security Token* (W-RST) and *Request Security Token Response* (W-RSTR). The W-RSTR usually contains an embedded SAML assertion. Since most web browsers are not able to make *simple object access protocol* requests, the WS-Federation specifies the *Web (Passive) Requestor* profile, which enables them to request and return security tokens. [35]

**Figure 6:** A sample WS-Federation request to AD FS for a security token.

An example request to AD FS, using the WS-Federation Passive profile, can be seen in Figure 6. The message is a *Hypertext Transfer Protocol* (HTTP) GET request to the AD FS endpoint with two parameters. The *wa* parameter must be set to “wsignin1.0” for login requests and the parameter *wtrealm* is used to specify the RP that the AD FS should issue a security token for. Upon successful authentication, AD FS will return the security token and redirect the browser to the RP. Figure 7 shows how the browser can return the security token to the RP. It is an HTTP POST request with a WS-Trust RSTR in the *wresult* parameter.

**Figure 7:** A sample WS-Federation request to return the security token to the RP.

The WS-Trust W-RSTR is shown, in a simplified form, in Figure 8. Expressed in *Extensible Markup Language* (XML), the W-RSTR specifies the lifetime of the token, the RP it is issued for, and the *requested security token* (RST). The RST in the figure is a SAML assertion, although, it can be any type of security token [36].
Federated identity management - AD FS for single sign-on and federated identity management
Carl Wikblom

4 Active Directory Federation Services 2012-08-28

Figure 8: A sample W-RSTR, excluding namespace declarations.

The SAML assertion contains the actual claims, as well as information about the issuer, the assertion lifetime, the authentication method, and the RP. A simplified SAML assertion, with the claims name and role, is shown in Figure 9. It can be noted that this part of the W-RSTR message is protected by a signature, and hence, a receiver is able to detect whether it has been tampered with.

```
<wst:RequestSecurityTokenResponse>
  <wst:Lifetime>
    <wsu:Created>2012-06-19T11:36:20.400Z</wsu:Created>
  </wst:Lifetime>
  <wst:AppliesTo>
    <wsu:Address>https://rp_address</wsu:Address>
  </wst:AppliesTo>
  <wst:RequestedSecurityToken>
    <saml:Assertion>...
      <saml:Issuer>https://rp_address</saml:Issuer>
      <saml:AttributeStatement>
        <saml:Attribute>
          <saml:NameFormat>urn:oasis:names:tc:SAML:1.0:attrnameFormat:uri</saml:NameFormat>
          <saml:AttributeValue>Alice</saml:AttributeValue>
        </saml:Attribute>
        <saml:Attribute>
          <saml:NameFormat>urn:oasis:names:tc:SAML:1.0:attrnameFormat:uri</saml:NameFormat>
          <saml:AttributeValue>Administrator</saml:AttributeValue>
        </saml:Attribute>
      </saml:AttributeStatement>
    </saml:Assertion>
  </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
```

Figure 9: Sample SAML assertion.

### 4.5.1 XML Signature verification

The XML Signature provides message integrity and signer authentication. The SAML assertion is the only part that is protected by a signature in the message. An XML Signature element is illustrated in Figure 10.
To verify the signature, a receiver must first normalize the signature element using the specified canonicalization method. Only one canonicalization element is allowed. After normalization, the hash digest value must be computed, using the specified signature method, over the entire signature element and compared with the supplied signature value. The signature might contain several references that can be validated by using the specified transforms, hash algorithms and digest values.

### 4.6 Alternative architectures

A study of FIdM architectures has been made in order to discover feasible alternatives to AD FS. The architectures had to support AD DS, claims based authentication, X.509 certificates, and have a respectable user base to be considered as alternatives. Of the several architectures that fitted the criteria, the three most interesting are presented in summary form in this chapter.

#### 4.6.1 PingFederate

Developed by Ping Identity, PingFederate is a flexible FIdM architecture that can be integrated into many different environments. For example, it can natively interoperate with AD FS. The IdSP and RP can be installed on several different operating systems and web servers. As account stores, PingFederate can make use of AD DS (also AD LDS), as well as relational databases and provides tools for custom integration of additional account stores. The IdSPs accepts identities from account stores, other IdSPs, and several IdM systems. Applications can integrate with
RPs from Java, .NET, Hypertext Preprocessor (PHP), as well as directly with IdSPs through Hypertext Transfer Protocol Secure (HTTPS) calls. [38]

4.6.2 Shibboleth
Shibboleth is an open source FIdM system with widespread usage in the academic world. It is possible to achieve interoperability with AD FS [18]. The IdSP requires a Java servlet container to run, while the RP supports different web servers and operating systems [39]. Users can be authenticated to different account stores, including AD DS, relational databases and other IdSPs [40]. The RPs communicates the claims to applications through environment variables [41]. This enables any program language, supported by the web server, to integrate with the RP.

4.6.3 OpenAM
ForgeRock provides an open source federation product named OpenAM. It is a continuation of Sun Microsystem’s OpenSSO. OpenAM IdSPs run as Java servlets, and the RPs can run on various web servers. AD DS and relational databases, among others, are supported as account stores [42]. Applications can integrate with RPs from Java, .NET and C, or choose to use HTTPS calls [43].
5 Methodology

To gain knowledge regarding the effects Federated Identity Management (FIdM) might have for organizations, a suitable experimental environment has to be created and evaluated. The environment will consist of two organizations in which FIdM allows for cross-domain authentication and Single Sign On (SSO). A security analysis and a functional test will be performed to evaluate the environment. This chapter describes the methodology used to create and evaluate the environment.

5.1 Experimental environment

The environment will have two organizations, in two separate domains, in which one of the organizations will host applications, which users from both domains must be able to access. Each organization will use Active Directory Federation Services 2.0 (AD FS) as an Identity Service Provider (IdSP) and thus allow for FIdM and SSO with the use of the WS-Federation protocol. While there are alternatives to AD FS, this thesis is written in cooperation with Logica which has requested its use. While this choice affects the installation and configuration of the IdSP, AD FS support for the WS-Federation protocol will still allow general results regarding the environment.

5.1.1 .NET and Java applications

Two applications that accept security tokens from AD FS will be created: one in .NET and one in Java. The only functionality of the applications will be that they will accept valid security tokens and print the claims contained in them.

Windows Identity Foundation (WIF) will be used to create the .NET application since it contains the necessary tools and Application Programming Interfaces (APIs) for managing security tokens in .NET and it is likely to be a major choice for developers [44].

For Java, no satisfactory implementation of WS-Federation has been found as they have been either incomplete or application server specific. Therefore, a custom implementation of WS-Federation Passive profile will be made to research the procedure of implementing it and to show
that no special technology is required to use it. This will also provide a deeper understanding of the protocol, which might prove useful in the security analysis. This will be an iterative development, as the security analysis might discover vulnerabilities that will require to be fixed. Two alternatives solutions are [45], which only supports Security Assertion Markup Language (SAML) 2.0, and [46], which is a tomcat plugin.

5.1.2 Servers and software resources
Five virtual machines have been provided by Logica and will be used to create the environment. Three machines, with 2.4 gigahertz central processing units (CPU) and 4.00 gigabytes of random access memory (RAM), run Microsoft Server 2008 R2 Enterprise, one runs Microsoft Windows 7 Ultimate Service Pack 1 and one runs Microsoft Windows XP Professional Service Pack 3. The three machines running server versions will host AD FS and the applications, while the other two machines will be used as clients. No major alternative exists to the Microsoft Server for hosting the .NET application, but, the Java application can be hosted on different platforms. Java’s platform independence will provide general results although some installation steps might differ between platforms. The only requirement for the clients is that they are able to use a web browser with basic functionality and therefore, the platform becomes irrelevant.

The .NET application will be developed in version 4.0 of the .NET Framework, with Active Server Pages (ASP) as the programming language, and deployed on an Internet Information Service (IIS) 7.0 web server. Microsoft Visual Studio 2010 will be used as the integrated development environment (IDE). Version 1.7 of the Java Development Kit (JDK) will be used for Java. NetBeans 7.1.1 will be used as the IDE and the application will be deployed on an Apache Tomcat 7.0.27 application server.

5.2 Functional test and security analysis
A simple functional test will be performed to verify the functionality of the environment. Using a web browser, the client in each organization will authenticate in its own domain and then be able to access both applications without any repeat authentication. The test will be performed using Microsoft Internet Explorer 8.0 and Mozilla Firefox 14.0.1.
Performing a security analysis of a complete environment is complex. Given the time constraints of this thesis, it is not possible to research every option available to an attacker. Since this is an experimental environment, hosting demonstration applications with very limited functionality, it is not possible to cover every threat in a live production environment and, indeed, it is highly likely that real world applications themselves will have vulnerabilities. However, the analysis will research the threats that using AD FS, FIdM, and SSO introduces. The main difference these techniques introduce is in the use of a security token instead of a username and password combination when a user authenticates against an application. The analysis will therefore include a brief research of the security mechanisms protecting the security token, test these security mechanisms in practice, and also, provide a brief discussion of threats that applications might introduce in the environment.

5.2.1 Threat model and security tests
The threat model explains the attack surface in the environment. The transport channel of the security tokens is assumed to be secure so that no attacker can exploit it to gain access. Furthermore, components in the environment that form no part of the authentication logic, such as the operating systems and web servers, are assumed to be secure. The applications and the certificates in the environment are also assumed to be secure. This means that the attacker must actively take part in the identity information exchange and exploit the security tokens in order to gain access or harm the services. Three roles take part in the exchange of security tokens, namely IdSP, RP, and user. Since IdSPs authenticate the users and the RPs trust them, it is assumed that an attacker cannot control them. Then, the attacker is able to take part in the identity information exchange as either a user or an RP. As a user, the attacker will have access to his/her own tokens and as a RP the attacker will have access to the tokens for all other users that access the RP. So the attack surface in the environment is the use of security tokens.

The purpose of the tests is to discover whether an attacker can exploit the security tokens in order to gain illegitimate access or otherwise harm the service. Three scenarios will be tested: A, B and C. Scenario A will test whether the attacker can forge, or alter an existing security token to gain access as another user. In scenario B, the purpose is to test whether the attacker can trick the IdSP into providing him/her with a valid
security token for another RP than that for which it is authenticated thus enabling the attacker to gain access to the other RP. The purpose of the last scenario, C, is to test whether an attacker can alter a security token in order to create a long processing time, which will ease a Denial-of-Service (DoS) attack. Firebug, an extension to Firefox, will be used to measure the time it takes for a server to process a request. The measurement will not provide values of the actual CPU and RAM resources consumed by the server, but longer processing time means that more resources are consumed.
6 Design

In order to investigate the effects of FIdM, a test environment has to be designed and deployed. This chapter describes the design and implementation of the environment as well as the two applications.

6.1 Requirements analysis and design considerations

According to section 1.4 in the introduction chapter, the following requirements have to be designed and implemented:

- A virtual environment including two organizations in which they can authenticate between each other using AD FS.

- Applications written in Java and .NET that accept authentication from AD FS.

Another requirement, in order for the security analysis to be meaningful, is that the environment and applications are implemented to be as secure as possible. This means that it will use all the security mechanisms available.

As this is an experimental environment with limited resources, some simplifications are made. No external Certificate Authority (CA) is used, so the AD FS servers will also function as CAs. Another role that the AD FS servers will have is as domain controllers, since two domains are necessary for the environment.

The .NET application will be a simple ASP.NET Web Application where WIF is used to add the functionality for claims-based authentication.

In Java, custom classes will be written to add the capability of claims-based authentication. This basically means that a class for handling the WS-Federation protocol has to be implemented. As has been determined from section 3.2, the protocols are often securely designed but their implementations and integration leads to security vulnerabilities, which makes this a delicate task. Great care must be taken so as not to introduce vulnerabilities. WS-Federation is a simple protocol in itself, but it uses more complex protocols such as WS-Trust and SAML. Luckily,
implementations of WS-Trust and SAML do exist, namely the Web Services Interoperability Technologies (WSIT), which will be used to ease the implementation of the WS-Federation protocol.

One purpose of claims-based authentication is that it can reduce the authentication complexity in application development. To make it easier for other developers to use the WS-Federation class in conjunction with AD FS, an authentication handler class will be implemented which eases the integration of the Java solution in future projects.

6.2 Experimental environment

An overview of the experimental environment can be seen in Figure 11. Both companies contain a client and an AD FS server (called A1 and A2). The AD FS servers are also configured as domain controllers, CAs, and Domain Name System (DNS) servers. Lastly, Company 2 has a server hosting IIS and Tomcat. All communication will be with HTTPS.

6.2.1 Configuring AD FS

A1 must be configured to issue claims for A2 and A2 must be configured to accept claims from A1 and issue claims for the two applications. The configurations can be performed automatically by providing metadata files but it is also easy to enter the data manually. A1 must be configured with A2’s public key (for encrypting security tokens), endpoint address (for sending the tokens), and URI (to be able to uniquely identify A2). If A1 has more than one token-signing certificate, it must also choose one to be used. The same configuration, with different parameters, must be performed at A2 for each of the applications. A2 must also know A1’s
6.2.2 Environment issues

An issue associated with AD FS was encountered when changing the time of the server. Changing the time backwards creates a problem with AD FS’s configuration database and AD FS will cease working, claiming that the database is corrupt. The database is not corrupt but some values in the database have to be reset, namely, the value LastPublishedPolicyCheckTime in the tables IdentityServerPolicy.Authorities, IdentityServerPolicy.MetadataSources, and IdentityServerPolicy.Scopes. Setting them to NULL will fix the issue and AD FS will work again. If no extensive configuration has been made, another solution is to simply reinstall AD FS.

6.3 .NET application

With the WIF SDK and some configuration, WS-Federation support can be added to a .NET web application. The configuration is specified in the web.config file.

6.3.1 Configuring authentication and registering WIF modules

In order to prepare the application for claims-based authentication, all unauthorized users must be denied access and all other types of authentication must be disabled. Figure 12 displays the configuration directives that achieved this.

```xml
<authorization>
    <deny users="?" />
</authorization>
<authentication mode="None" />
```

Figure 12: Configuration directives to deny unauthorized users and disable other authentication.

The next step is to register the necessary modules. The WSFederationAuthenticationModule module redirects unauthenticated users to the IdSP
and parses security tokens returned from it; users who are already authenticated are managed by the `SessionAuthenticationModule` module. Figure 13 shows how to add these modules in the configuration section for IIS. The `ClaimsAuthorizationModule` module can also be registered in the same way, but this application will not make any authorization decisions.

```xml
<modules>
  <add name="WSFederationAuthenticationModule" type="Microsoft.IdentityModel.Web.WSFederationAuthenticationModule, Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" preCondition="managedHandler" />
  <add name="SessionAuthenticationModule" type="Microsoft.IdentityModel.Web.SessionAuthenticationModule, Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" preCondition="managedHandler" />
</modules>
```

Figure 13: Registration of necessary modules.

### 6.3.2 Configuration WIF modules

Figure 14 shows how to add a configuration section for WIF in the configuration file.

```xml
<configSections>
  <section name="microsoft.identityModel" type="Microsoft.IdentityModel.Configuration.Microsoft.IdentityModelSection, Microsoft.IdentityModel, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
</configSections>
```

Figure 14: Adding a configuration section for WIF.

The `microsoft.identityModel` is used to configure WIF and is divided into several subsections. In the `claimTypeRequired` section, optional and required claims are specified. A security token, without all the required claims, is invalid. The `audienceUris` contains all valid URIs for the application. The `AudienceRestrictionCondition` (see section 4.5) in the SAML assertion must match one of the URIs in order to be valid. The integration with the AD FS server is configured in the `federatedAuthentication` section. Here, the WS-Federation Passive profile is enabled and is configured by enabling passive redirect and specifying the URL of the AD FS server and the URI of the application itself. See Figure 15 for an example. A token-decrypting certificate can be specified in the `serviceCertificate` section and IdSPs public keys are configured in the `issuerNameRegistry` section.
6.3.3 Request validation

By default, web applications reject the HTTP POST containing the security token since it is expressed in XML. One option is to turn off the request validation, but that can compromise the security of the web application. A better option is to use a custom request validator that accepts security tokens and redirects all other requests to the standard validator.

Such a request validator has been implemented. For each request, it checks whether it is a HTTP POST request containing the parameter wresult. If not, it redirects validation to the standard request validator. But, if the request contains the wresult parameter it will attempt (using the WIF SDK) to build a valid W-RSTR message from the parameter’s value. If the creation of the W-RSTR message is successful, the request is valid and the validation terminates, however, if the message creation fails, then the request will be redirected to the standard request validator.

6.3.4 Deployment

To deploy the application, it is added as an Application to the default web site in IIS. For file access purposes, the application will run under its own Application pool. The application pool must run with version 4.0 of the .NET framework and this application pool uses an integrated Managed Pipeline.

In order for the application pool user to be able to read the required files, it must be added to the access control list (ACL) of some files. The files required in order to be able to read are those in the application directory and the file where its private certificate is stored. The certificate file can be found by using the command “certutil.exe –store my”, which will reveal its container name. If, for example, the application pool is named WIFPool, the user which is required to be added in the ACLs is “IIS AppPool\WIFPool”.

Figure 15: Configuration of the WS-Federation Passive profile.
6.3.5 Application issues
A problem encountered in the application is that, if the application is accessed without a trailing slash, WIF gives an error. For example, accessing the application through “https://address/” works, but accessing it through “https://address” will give an error. The error can be fixed by customizing the Application_Error function in global.asax to check for the error code, and then redirecting to the same address with a trailing slash.

6.4 Java application
The greatest task in the Java application development is to create the WS-Federation Passive profile implementation. Furthermore, an authentication handler will be implemented to ease the use of the WS-Federation implementation in conjunction with AD FS.

6.4.1 Overview of the WS-Federation design
An overview of the most important parts of the WS-Federation is illustrated in Figure 16. As described in chapter 4.5 of the report, WS-Federation makes use of WS-Trust, and WS-Trust makes use of SAML.

![UML class diagram](image)

Figure 16: An UML class diagram illustrating the WS-Federation Passive profile implementation.

The PassiveRequestor class is the front end of the implementation. It redirects users for authentication and parses the returned security
tokens. To achieve this, it uses a RequestSecurityTokenResponseHandler (that handles the W-RSTRs) which in turn uses a RequestedSecurityTokenHandler. SAMLTokenHandler is an implementation of the RequestedSecurityTokenHandler interface and handles SAML assertions. These classes will be described in the chapter that follows.

6.4.2  WS-Federation implementation

The PassiveRequestor class takes care of the WS-Federation protocol and is responsible for redirecting users for authentication, plus checking for returned security tokens. It requires three parameters: the address of the IdSP endpoint, the RPs URI, and the public key of the IdSP. The address and the URI are used to create the request to the IdSP, while the URI and the key are used to validate the returned security token. The sendRequestSecurityToken and processRequestSecurityTokenResponse functions can be used to handle the authentication procedures.

The purpose of the RequestSecurityTokenResponseHandler is to parse the W-RSTR, extract the RST, and hand it to the RequestedSecurityTokenHandler class. The only information in the W-RSTR that is signed is the RST. Hence, no information apart from the RST can be trusted. The only unsigned information that is used, in order to construct an appropriate RequestedSecurityTokenHandler, is the TokenType element. The WSIT API is used to parse and handle the W-RSTR. Another important task for the class is to validate W-RSTR responses to ensure they only contain the necessary elements. Using a strict schema will minimize the attack surface as an attacker is unable to make any significant alterations to the W-RSTR. Part of the XML Schema used for validation can be viewed in Appendix A, Figure A.1.

The RequestedSecurityTokenHandler is an interface to group handlers for requested security tokens. An implementation of the interface is the SAMLTokenHandler class. It uses the WSIT API to parse the SAML assertion, validate it, and extract claims from it. The validation parameters are passed down from the RequestedSecurityTokenResponseHandler by the validate function. It also validates the structure of the assertion, using the XML Schema that is, partly, presented in Figure A.2, Appendix A. The class currently validates the signature and that the assertion is intended for the application. It is also responsible for decrypting the RST if it is encrypted and the decryption key is currently loaded from a Java
keystore in this class. The getClaims function will return name-value pairs of the extracted claims.

6.4.3 Authentication handler implementation

The authentication handler assists in the integration of the WS-Federation implementation with AD FS. It consists of two classes and a configuration file. Identity is a class that holds the identity of a claims-based authenticated user. The AuthenticationHandler class handles the authentication (all it requires is the location of the configuration file) and sets and instance of the Identity class in the session upon successful authentication. Figure 17 shows how the AuthenticationHandler can be used in an HttpServlet to authenticate users.

```java
try {
    AuthenticationHandler authHandler = new AuthenticationHandler(request, response);
    authHandler.authenticate();
} catch (AuthenticationException ae) {
    // User redirected. Do nothing.
    return;
}
```

Figure 17: Usage of the AuthenticationHandler.

The authenticate function is presented in Figure 18. It checks whether a session already exists, or if a security token is present in the request, otherwise it redirects the user to AD FS. It uses the WS-Federation implementation to check for security tokens and redirect users.

```java
public void authenticate() throws ServletException, IOException,
    AuthenticationException, AuthenticationFaultException {
    try {
        checkCookie();
    } catch (AuthenticationException ae) {
        // No previous session found
        // Check for authentication in GET or POST data
        try {
            checkParam();
        } catch (AuthenticationException ae2) {
            // Redirect to AD FS for authentication
            redirect();
            throw new AuthenticationException("Redirected for authentication");
        }
    }
}
```

Figure 18: The authenticate function.

```xml
<config>
  <SP>
    <realm>https://webserver.ex-company2.com/ClaimsAwareJavaWebApp</realm>
  </SP>
  <IdSP>
    <name>Company2IdSP</name>
    <endpoints>https://sts.ex-company2.com/adfs/ls/</endpoints>
    <publickey>company2idsp.cer</publickey>
  </IdSP>
</config>
```

Figure 19: Example of the configuration file.

The configuration file for the AuthenticationHandler is illustrated in Figure 19. The public key is an X.509 certificate file.
6.4.4 Application issues

A problem encountered with the SAML implementation in the WSIT API was that some, valid, assertions failed to validate. The culprit was the parsing of an assertion, as some attributes become moved. Figure 20 shows a part of an assertion that is wrongly parsed, and Figure 21 shows the result of the parsing. It can be seen that $a:OriginalIssuer$ and $xmlns:a$ are moved inside the document, which breaks the signature verification.

This issue can be bypassed by validating the assertion without using the SAML implementation to parse it.
7 Security analysis

A security analysis will be performed of the experimental environment. This chapter will describe the security mechanisms in the environment, the performed security tests, and other security considerations.

7.1 Security mechanisms

The security of the environment depends on the ability of the RPs to trust that the security tokens they accept are issued by the IdSP, intended for them, and carried by the correct user. This means that no one should be able to forge a security token, steal a security token, use security token from one RP to access another RP, or trick the IdSP into issuing a token for another RP than that authenticated for the user.

HTTPS is used as the transport security to avoid eavesdropping, replay and man-in-the-middle (MITM) attacks. Although attacks against Secure Socket Layer (SSL) and Transport Layer Security (TLS) exist, see [47] and [48], they are not exclusive for this environment and they can also be avoided. The use of HTTPS will also defend against phishing attacks since the user is able to identify the web sites.

Regardless of HTTPS, some parties will still be able to access the security tokens since they are relayed through the browser and sent to RPs. Therefore, the security tokens are encrypted using XML Encryption and signed with an XML Signature. In certain circumstances, the XML Encryption can be broken, see [49]. For the attack against XML Encryption to work, an application has to provide detailed error messages of the decryption. Although this attack is easily avoided, RPs will be able to decrypt the messages, and they are also able to encrypt them again using another RP’s public key. The XML Signature protects the integrity of the message, that is, the receiver can verify the sender. Bypassing the signature protection of a message will allow an attacker to gain access to an application as another user. Several attacks, that are presented in [50] and [51], can bypass the protection signatures and will be tested in the security test.
7.2 Security tests

The security tests are performed in order to test some security aspects of the environment. Notably, the tests will determine whether the security tokens can be forged, altered, erroneously acquired, or used in a DoS attack.

7.2.1 Forging a security token

This section describes the tests used to determine whether it is possible to forge or alter a security token.

Simply sending a forged security token to the applications or the IdSP does not work because it lacks a valid signature. Another possibility is to alter an already signed security token and, while keeping the signed data intact, insert new data in the security token. An element wrapping attack is an example of this and several techniques are described in [50] and [51]. An example of a basic element wrapping attack is shown in Figure 22. In the figure, the signed element has been moved to a dummy element, and if the validator extracts this element by means of the URI, the validation will be successful. The vulnerability exploited by the attack is that the signature validation finds the element by means of the URI, while the remainder of the application finds it by means of a path.

```xml
<root>
  <protected URI="fake">
    <user>Attacker</user>
    <signature>
      <reference URI="real" />
      <signatureValue>...</signatureValue>
    </signature>
  </protected>
  <dummy>
    <protected URI="real">
      <user>RegularUser</user>
      <signature>
        <reference URI="real" />
        <signatureValue>...</signatureValue>
      </signature>
    </protected>
  </dummy>
</root>
```

Figure 22: An element wrapping attack.

Several tests have been made to execute different element wrapping attacks but all have failed. The XML Schema validation in the applica-
tions stops custom elements from being inserted, effectively preventing the attack since no false assertion can be put into a custom element. The ASP.NET application and AD FS accept altered assertions inserted in the Advice element of another assertion. However, the application only extracts claims from the signed element so it cannot be exploited. The Java application only accepts one assertion element and it must be correctly signed.

Even if the executed tests have not discovered any vulnerability, there is still a possibility that they do exist. If this is the case then an RP owner will have a greater chance of exploiting them than a regular user. It would be difficult for a regular user to execute an element wrapping attack since the SAML assertion is encrypted, but an RP has access to unencrypted tokens and thus, has a better chance of success.

### 7.2.2 Erroneously acquiring a security token

This section will describe the tests used to determine whether the IdSP can be tricked into issuing a security token for another RP than that for which the authentication occurs.

Several attempts have been made in order to trick the IdSP into providing the RP with a security token for another RP. The WS-Federation protocol specifies that the wreply parameter can be used in a request for a security token. A naïve implementation of the protocol might trust the wreply parameter, but AD FS only considers it if it matches a configured endpoint for the RP.

One other opportunity is for an RP to add the URI of another RP to itself in the IdSP configuration. This might be possible since the IdSP can update the RP configuration automatically from a metadata file at the RP. However, AD FS does not allow two RPs to possess the same URIs. This can be circumvented with some URL encoding but since the requests are made through HTTPS, AD FS will decode the requests to a correct URI. For example, it is possible for an RP (A) to successfully register a URI of another RP (B) with a %0d appended to it. A request with this URI to AD FS will, however, issue a token for RP-B but will also return it to RP-B. It is possible that some option exists to exploit this but none has been found. Turning off the automatic updates of RPs’ configurations will eliminate the threat.
7.2.3 Denial of service attacks

The purpose of the tests described in this section is to determine whether the security token can be used to launch a DoS attack.

Since no unsigned data is processed or parsed, only large amounts of data can be inserted into the unsigned portion of the security token. This is not an exclusive attack to this environment and since the servers limit the size of the requests no tests have successfully achieved a DoS condition.

Another attack surface exists if the application validates the referenced URIs before the SignedInfo element. That would allow resource heavy transformations to be inserted, which the application would perform before the validation fails. However, the applications perform the validations in the correct order. In the security token for the ASP.NET application, several Advice elements can be inserted successfully. This will cause the WIF to parse all advice elements, executing the readAssertion and readAdvice functions for each Advice element. AD FS limits the depth of the XML tree when parsing and, hence, only a few Advice elements can be inserted. Another possible DoS attack is to include a large external file in the URI of the reference element, but none of the implementations appear to accept an external file.

Inserting a Document Type Definition (DTD) with many entities or recursive definitions can force the XML parser to consume a great deal of resources when these entities are used in the document. Neither the applications nor AD FS, allow DTDs which makes this attack impossible. The canonicalization (normalization) of the document can be exploited by inserting a significant amount of extra XML that is required to be processed. This works with the .NET application and AD FS, but measurements have not revealed any significant impact on the processing time.

7.3 Security considerations

The security of the system relies on the mechanisms described in the introduction to this chapter. Even if the security tokens are secure, other security threats exist. The environment might even cause it to be easier to use several common vulnerabilities.
The applications themselves might introduce vulnerabilities. Even if the authentication logic is externalized from the application, other threats such as cross-site scripting (XSS) and cross-site request forgery (CSRF) must still be accounted for. Both applications use cookies for sessions, and XSS exploits can endanger the security of the applications by stealing cookies. Stealing cookies by means of Javascript is aggravated by the fact that the cookies are HTTP only, and thus inaccessible from Javascript. However, exploits exist that are able to overcome that obstacle. With CSRF, a website owner might be able to execute URL requests as the user who visits the site. This kind of environment might heighten a user’s vulnerability to CSRF attacks because of the SSO feature. It is possible to protect against CSRFs attacks, but XSS vulnerabilities in the application (or another application running on the same domain) might allow an attacker to circumvent the protection mechanisms.

Another vulnerability associated with the environment is that an RP can proxy requests to the IdSP instead of redirecting the user to the IdSP, effectively creating a form of MITM attack. As the RP will have a valid certificate, the user’s browser will trust the website and the user might expect a login page, especially if it is the first RP that has been visited. The only difference for a user is that the URL in the browser is still that of the RP.
8 Results

The results of the functional test and the security analysis are presented in this chapter.

8.1 Functional test

The function test shows that the environment is fulfilling all functional requirements. A user is able to authenticate in his/her own domain and then access the deployed RPs. Two RPs have been implemented, one in Java and one in .NET.

Figure 23: Alice from Company 1 has authenticated successfully to the .NET application using Internet Explorer 8.0.

Figure 23 show a user from Company 1 who has successfully authenticated to the .NET application, and Figure 24 shows a user from Company 2 accessing the Java application.

With the RPs deployed in Company 2, Figure B.1 and B.2 in Appendix B show the traces (cleaned up) of the HTTP requests when users, from Company 1 and Company 2 respectively, access an RP. The difference between the two web traces is that the user from Company 1 is redi-
rected to the IdSP of Company 1 while the user in Company 2 can authenticate with the IdSP in Company 2. Figure B.3, in Appendix B, shows the HTTP traces of a user accessing an RP when it has already been authenticated to the AD FS. The abundance of POST requests to the AD FS server should be noted. In this case the user is not required to submit a repeat login form.

### 8.2 Security analysis

The overall security of the environment is good. The security tests have not revealed any vulnerability that defeats the purpose of authentication, but the security token might be exploited in order to ease a DoS attack of the .NET application. Table 1 summarizes what the different roles in the environment can exploit.

A user is a regular user in any domain; an RP owner is able to control a RP that regular users visit and a website owner controls a website, without authentication, that users visit.

<table>
<thead>
<tr>
<th>Role</th>
<th>Forge security token</th>
<th>Wrongfully acquire security token</th>
<th>Denial of service</th>
<th>Cross-site request forgery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular user</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Relying party owner</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Possibly</td>
</tr>
<tr>
<td>Website owner</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Possibly</td>
</tr>
</tbody>
</table>

Table 1: A summary of different roles’ exploit possibilities in the environment.

Since the security tests revealed that the applications will only extract claims from properly signed elements, the IdSP’s private key is necessary in order to forge a security token. Although no test was able to successfully trick the AD FS into issuing an unintended token, it was revealed that an RP could add URIs to itself that did match another RP’s URI under certain conditions. It is possible that this could allow an RP to acquire valid tokens for another RP. The possibility for the RP and website owners to execute CSRF attacks depends on the security of the application, but is included at this point as the environment’s SSO capability increases an attacker’s potential to perform such an attack. Since a valid signature is required in order to perform the DoS attacks, only an RP owner, who has access to unencrypted tokens, can launch such an attack.

A comparison of the processing times between a regular request and the two DoS attacks for the .NET application can be seen in Figure 25. In the
Advice element DoS, 500 Advice elements have been inserted into the security token and in the Canonicalization DoS, 10 000 attribute elements have been inserted.

Figure 25: Comparison of request timings between requests.

The more potent of the two DoS attacks is definitely the Advice element DoS. It takes the server more than 500 times longer to process an Advice element DoS request, as compared to that for a regular request. However, it should be noted that a single Advice element DoS request will not lead to a successful DoS attack, but, utilizing these types of requests instead of regular requests will greatly assist an attacker. Since AD FS limits the number of Advice elements that can be inserted to around ten, the attack does not have a significant impact on the processing time.
9 Conclusions

The aim of the thesis was to investigate the effects that an FIdM system with SSO capabilities will have for an organization by deploying and evaluating such a system. To achieve this, a virtual environment with two organizations had to be created. In the environment, users had to be able to authenticate between the organizations using AD FS and be able to access two applications, with one created in .NET, and the other created in Java. A security analysis had to be performed in order to identify the risks and vulnerabilities of the environment.

The result of the functional test shows that an experimental environment with two organizations and two applications has been implemented successfully. With the use of AD FS, users from both organizations can authenticate in their own organization and are then able to access the services of the organizations without any repeat authentication. Both the .NET application and the Java application are able to accept claims issued by AD FS as authentication. The result of the security analysis shows that the overall security in the system is good. The use of AD FS does not allow anyone to bypass the authentication process. However, the security tokens might assist an attacker to successfully launch a DoS attack against the .NET application.

In general, the results obtained were as expected. However, the fact that WIF has restrictions for the XML parser in AD FS, but not in the .NET application was unexpected. This led to the result that AD FS is protected from the Advice element DoS attack, while the .NET application is susceptible to it.

This contributions made by this thesis are in relation to a security analysis of AD FS and WIF as well as an implementation of WS-Federation in Java, together with a helper class for using the implementation together with AD FS and an easing of its integration for developers in future projects.
9.1 Effects of federated identity management

By centralizing the users’ identities, FIdM will have a positive impact on an organization’s user management. Adding, removing, and changing the identities of the users will only have to be performed in one place. This will save time and will minimize errors. The federated characteristic will allow organizations to manage only their own users. Users from partners that require access to an organization’s system are able to be managed by the partners. Overall, FIdM will reduce the cost of user management.

Externalizing authentication from applications will also be positive for developers as they need not implement any functionality for managing users within the applications. It is highly probable that users accept the introduction of FIdM in an organization as the SSO functionality will reduce the amount of passwords, as well as the number of login procedures, required in their work. This will lead to fewer password resets and interruptions.

The results of this thesis indicate that these effects are possible. Even though the tests have been performed in a small scale environment, no limitations or difficulties have been encountered that would suggest otherwise. Although AD FS has been used, the same effects can be achieved with other FIdM systems. FIdM will, in addition, not impair the security of the environment, but will rather assist it as password management for the users is improved. An important concern regarding the reliability of the system is that the IdSP can become a single point of failure. As the security tests have been performed with AD FS, no conclusions regarding the security of other FIdM systems can be drawn. However, it does show that it is possible to make the systems secure.

It must also be noted, that while the authentication security is externalized from the application development, other security aspects are not. There are still many security threats against an application that must be accounted for.

This thesis has focused on a B2B scenario, in which the user has a distinct identity, controlled by its organization. A user’s organizational and a user’s private identities are two different aspects. In this scenario, it is clear that the organization is responsible for the authentication of the user.
9.2 Recommendations for future work

This thesis has focused on the authentication part of FIdM. However, the aspect of authorization is also important and requires additional research. Future work will be able to research the possibility, effects, and security of centralized authorization, or how authorization decisions based on claims will impact upon existing applications. Signing off is another aspect that should be researched. This work has focused on the act of signing on, but it also led to questions regarding how the sign off should function in a SSO system. More specifically:

- Is it possible to have a single sign off function that users in the system naturally understand?
- How should a user determine whether he/she has been signed off from all applications?
- Where should a user initiate the single sign off?

Another point of interest concerns how to handle applications that accept anonymous functions, for example, a commentary function at a corporate intranet. If the commentary function accepts both anonymous and authenticated users, how will the SSO be handled by the application?

Additionally, the user experience of FIdM and SSO must be researched. This thesis has focused on the technical aspects of deploying a FIdM system. But will the users:

- still trust the security?
- understand that they are signed into all application they can access?
- accept stronger password policies enforced upon them, since their password management is eased, or will they create dangerous shortcuts?

The security analysis also reveals a point for further research. The possibility for an RP to add a URI that evaluates the equivalent URI of another RP might pose a security threat. Research should be conducted as to whether or not this is able to be exploited, as the feature of automatically
configuring RPs from metadata files has the ability to ease the deployment of a system.

The WS-Federation implementation and the authentication handler could be improved. The WS-Federation implementation requires further testing and by extending it, this could make it useful for more scenarios. It is currently only tested with AD FS, but other systems might use parts of the protocols not used by AD FS. Extending the authentication handler to interoperate with other systems will make it more useful in heterogeneous environments.

9.3 Recommendations regarding integrating federated identity management into an organization

Organizations can deploy FIdM to ease their user management and cut costs which are related to it. This work has not found any reason to not recommend it. In fact, the FIdM technology evaluated in this thesis has proved to be sufficiently mature for both a B2B scenario and inside an organization. Preceding the implementation of an FIdM system, the recommendation is to perform an analysis with regards to how the authorization in the system will be managed.

The solution will require a new infrastructure and the porting of existing applications. Depending on the size of the organization, and the amount applications, this will involve a significant amount of time and effort. Depending on the technical environment, new technology might be required to be implemented in order to enable claims-based authentication to take place. However, there appear to be no difficulties which are unable to be overcome in deploying a FIdM system with AD FS.

To make the system secure and to limit the attack possibilities, some security considerations must be taken into account. It is always necessary to use the available security mechanisms. For example, even though security tokens are encrypted in transport with HTTPS, it is also necessary to encrypt the security tokens with XML Security so as to limit the possibilities for an intermediate receiver to alter and exploit the tokens. The current recommendation is to not automatically update RP or IdSP configurations from metadata files. Finally, since the security of the system is very much dependent on the certificates, they must be transferred securely.
When the security protocols to be used in the environment are implemented, the recommendation is that they should be used very strictly and input from users should never be trusted. The security test and iterations of the implementation of WS-Federation for Java has led to this conclusion and is supported by the security research presented in section 3.2, namely, that the more flexibility and unnecessary functionality being supported, the greater are the number of options open to an attacker, thus making it less secure.

The security analysis presented in this thesis only concerns the actual security implications involved in introducing AD FS as system for FIdM. Using this system together with real world applications may introduce additional threats to the overall security of the system. Therefore, a complete security analysis of the system is recommended.
References


Federated identity management - AD FS for single sign-on and federated identity management
Carl Wikblom

References
2012-08-28


Retrieved 2012-04-10.

Retrieved 2012-04-10.


Retrieved 2012-04-10.
References

Retrieved 2012-04-10.

Retrieved 2012-08-02.

[29] Microsoft, “Understanding the AD FS Web Agent Role Service”,
Retrieved 2012-04-10.

Retrieved 2012-04-10.

Retrieved 2012-04-10.

Retrieved 2012-04-10.

Retrieved 2012-04-10.

[34] Microsoft, “Understanding Claims”,
Retrieved 2012-04-10.

http://docs.oasis-open.org/wsfed/federation/v1.2/os/wsfederation-1.2-spec-os.html#_Toc223175002
Retrieved 2012-08-02.
Federated identity management - AD FS for single sign-on and federated identity management

Carl Wikblom

References

2012-08-28

Retrieved 2012-08-02.

Retrieved 2012-08-03.

Retrieved 2012-04-11.

Retrieved 2012-04-11.

[40] Shibboleth Wiki, “IdPAddAttribute”, https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAddAttribute
Retrieved 2012-04-11.

Retrieved 2012-04-11.

Retrieved 2012-04-11.

Retrieved 2012-04-11.

Retrieved 2012-08-02.
Retrieved 2012-08-02.

Retrieved 2012-08-02.

Retrieved 2012-08-03.

http://vnhacker.blogspot.se/2011/09/beast.html
Retrieved: 2012-08-03.


Appendix A: XML Schemas used for validation in the Java application

```xml
<xs:element name="RequestSecurityTokenResponse" type="wst:RequestSecurityTokenResponseType"/>
<xs:complexType name="RequestSecurityTokenResponseType">
  <xs:sequence>
    <xs:element ref="wst:Lifetime" minOccur="0" maxOccur="1"/>
    <xs:element ref="wst:AppliesTo" minOccur="0" maxOccur="1"/>
    <xs:element ref="wst:RequestedSecurityToken" minOccur="0" maxOccur="1"/>
    <xs:element ref="wst:TokenType" minOccur="0" maxOccur="1"/>
    <xs:element ref="wst:RequestType" maxOccur="1"/>
    <xs:element ref="wst:KeyType" minOccur="0" maxOccur="1"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="RequestedSecurityTokenType" type="wst:RequestedSecurityTokenType"/>
<xs:complexType name="RequestedSecurityTokenType">
  <xs:sequence>
    <xs:element ref="wst:EncryptedData" minOccur="0" maxOccur="1"/>
  </xs:sequence>
</xs:complexType>
```

Figure A.1: Part of the XML Schema, showing the allowed structure, used to validate the W-RSTR.

```xml
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
  <sequence>
    <element ref="saml:Conditions" minOccur="0" maxOccur="50"/>
    <choice maxOccur="0">
      <element ref="saml:Statement"/>
      <element ref="saml:SubjectStatement"/>
      <element ref="saml:AuthenticationStatement"/>
      <element ref="saml:AuthorizationDecisionStatement"/>
      <element ref="saml:AttributeStatement"/>
    </choice>
    <element ref="ds:Signature" minOccur="0"/>
  </sequence>
  <attribute name="MajorVersion" type="integer" use="required"/>
  <attribute name="MinorVersion" type="integer" use="required"/>
  <attribute name="AssertionID" type="ID" use="required"/>
  <attribute name="Issuer" type="string" use="required"/>
  <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

Figure A.2: Part of the XML Schema, showing the allowed structure, used to validate the SAML assertion. It can be noted that no Advice element is allowed.
Appendix B: Web traces of the login sequences

GET /CfInsAwareAspwebApp/ HTTP/1.1
Host: webserver.ex-company2.com

HTTP/1.1 302 Found

POST /adfs/ls/ls?login=https://sts.ex-company2.com/CFIdfInsAwareAspwebAppADFS/6f4e72f21bf4faa3189769e50f1f2689&ct=2012-09-01T09:59:30Z HTTP/1.1
Host: sts.ex-company2.com

HTTP/1.1 302 Found

GET /adfs/ls/ls?login=https://sts.com/CFIdfInsAwareAspwebAppADFS/6f4e72f21bf4faa3189769e50f1f2689&ct=2012-09-01T09:59:30Z HTTP/1.1
Host: sts.ex-company2.com

HTTP/1.1 200 OK

POST /adfs/ls/ls?login=https://sts.com/CFIdfInsAwareAspwebAppADFS/6f4e72f21bf4faa3189769e50f1f2689&ct=2012-09-01T09:59:30Z HTTP/1.1
Host: sts.ex-company2.com

HTTP/1.1 200 OK

https://sts.ex-company2.com/adfs/ls/

POST /adfs/ls/ls?login=https://sts.com/CFIdfInsAwareAspwebAppADFS/6f4e72f21bf4faa3189769e50f1f2689&ct=2012-09-01T09:59:30Z HTTP/1.1
Host: sts.ex-company2.com

HTTP/1.1 200 OK

POST /CfInsAwareAspwebApp/ HTTP/1.1
Host: webserver.ex-company2.com

HTTP/1.1 302 Found

GET /CfInsAwareAspwebApp/ HTTP/1.1
Host: webserver.ex-company2.com

HTTP/1.1 200 OK

Figure B.1: Web trace of a user in Company 1 authentication to a RP in Company 2.
Federated identity management  -  AD FS for single sign-on and federated identity management  
Carl Wikblom  

References  2012-08-28  

---

GET /ClaimsAwareaspwebapp/ HTTP/1.1  
Host: webserver.ex-company2.com  

HTTP/1.1 302 Found  

---

GET /adfs/ls/?wa=wsignin1.0&wtrealm=https://webserver.ex-company2.com/ClaimsAwareaspwebapp HTTP/1.1  
Host: webserver.ex-company2.com  

HTTP/1.1 200 OK  

---

POST /adfs/ls/?wa=wsignin1.0&wtrealm=https://webserver.ex-company2.com/ClaimsAwareaspwebapp HTTP/1.1  
Host: webserver.ex-company2.com  

wa=wsignin1.0&result=w-RSTR  

HTTP/1.1 302 Found  

https://webserver.ex-company2.com/ClaimsAwareaspwebapp/  
GET /ClaimsAwareaspwebapp/ HTTP/1.1  

HTTP/1.1 200 OK  

---

Figure B.2: Web trace of a user in Company 2 authentication to a RP in Company 2.

---

GET /ClaimsAwarejavawebapp/ HTTP/1.1  
Host: webserver.ex-company2.com:8443  

HTTP/1.1 302 Moved Temporarily  

---

GET /adfs/ls/?wa=wsignin1.0&wtrealm=https://webserver.ex-company2.com/ClaimsAwarejavawebapp/ HTTP/1.1  
Host: webserver.ex-company2.com  

HTTP/1.1 200 OK  

---

POST /ClaimsAwarejavawebapp/ HTTP/1.1  
Host: webserver.ex-company2.com:8443  

wa=wsignin1.0&result=w-RSTR  

HTTP/1.1 200 OK  

---

Figure B.3: Web trace of a user in Company 2, who is already authenticated at AD FS, accessing a RP.