

JISC DEVELOPMENT PROGRAMMES

Project Document Cover Sheet

DELIVERABLE

Project

Project Acronym	ASMIMA	Project ID	
Project Title	Adoption of Shibboleth for Multiple Identity Management Applications		
Start Date	April 2005	End Date	March 2006
Lead Institution	Cardiff University		
Project Director	Hugh Beedie		
Project Manager & contact details	Joan Wright wright@cardiff.ac.uk 029 2087 4496		
Partner Institutions	n/a		
Project Web URL			
Programme Name (and number)	JISC Core Middleware Programme (Shibboleth Early Adopters)		
Programme Manager	Nicole Harris		

Document

Document Title	Report on Knowledge Gained and Resilient Design Plans		
Reporting Period			
Author(s) & project role	Hugh Beedie, Project Director		
Date	15/8/2005	Filename	Knowledge and Resilience.rtf
URL			
Access	<input type="checkbox"/> Project and JISC internal		<input type="checkbox"/> General dissemination

Document History

Version	Date	Comments
1.0	15/8/2005	

Cardiff University

Project ASMIMA

(Adoption of Shibboleth for Multiple Identity Management Applications)

Report on Knowledge Gained and Resilient Design Plans

Dr H R A Beedie

Mr R Hebron

15 August 2005

Introduction

This document discusses the knowledge gained during the first 3 months of the Shibboleth Early Adopters project at Cardiff University about Shibboleth Origin (Identity Provider) architecture and processes, and proposes a resilient technical design for a pre-production Shibboleth service.

Non-technical issues relating to overall service availability, such as metadata definition and maintenance, liaison with trust federations, and production of operational procedures have not been addressed at this stage.

Background

A previously commissioned design study [Appendix A] discussed Shibboleth concepts and architecture, and proposed two options for a prototype system design. Option 2 was chosen as the preferred design by the Cardiff Shibboleth Project Team for reasons of final design simplicity. Issues relating to the design of a resilient system were also discussed in that design study. This paper describes the proposed resilient design and seeks to explain the reasoning behind its selection.

During project startup, it became apparent that there were two major reference implementation choices, from Athens and from the Internet2 project. The Internet2 project

was the only one to provide reference implementations of both Origin and Target (Identity Provider and Service Provider in the latest jargon), and was thus chosen as being most appropriate for Cardiff. This decision was further complicated by the timescale of the release of the next version (Beta V1.3). Initially, a pilot installation was built using V1.2. A subsequent V1.3 installation was found to be sufficiently compatible with a V1.2 Service Provider, and the final prototype and production services are now specified to use V1.3.

Knowledge Gained So Far - Lessons Learned

Early enquiries revealed that the statistics available regarding Cardiff's Athens usage were not in themselves sufficient to enable accurate estimates of the maximum 'hit rate' on a Shibboleth Identity Provider service. Our lack of understanding of Shibboleth and the Shibboleth-Athens gateway introduced further uncertainty, which made it difficult to specify the required hardware.

The following estimates were initially used to provide reassurance that a single server would be sufficiently powerful to satisfy the requirement. There are approximately 50,000 Athens accesses per month from Cardiff. This implies an average load of 5 accesses per minute (Say 20 days per month => 2,500 per day => 300 per hr => 5 per minute). Maximum loads were estimated at 5 times the average rate, so the maximum estimated load was approximately 1 Shibboleth authentication every 2 seconds.

Following a request to Athens for Cardiff's usage in the last year, an analysis was made of 2 significant 3 month periods. The highest authentication rate observed was 148 per minute (2.5 per second), with the next highest being 76 per minute. If no further information becomes available, we will aim to design the implementation to be able to deal with up to 10 authentications per second, thus allowing for significant future expansion in the use of Shibboleth.

Investigation of Shibboleth showed that there is no support for 'session or handle sharing'. This makes it difficult to provide a fully resilient system front-ended by a Layer 4-7 Application switch. However, given that a single authentication is only likely to take a few seconds to complete, and that the authenticated client then needs no contact with the local Shibboleth server, it will be reasonable to provide a service that fails over automatically, but with the potential loss of 'authentications in progress' only.

The decision to implement the pilot using newly released beta code was difficult, and problems were experienced as a result. However, the use of the beta V1.3 in the pilot phase should mean that the implementation phase will be taking place in a known environment.

The implications of testing of the pilot system in a production environment have been considered, and a potential minor problem has surfaced; how to test a newly updated full user interface in the live Athens Federation whilst hiding the service from normal users.

Resilient Design

The need for resilience in IT services is increasing as more users require 24X7 access to services. The highest level of resilience would typically involve clustered solutions, with automatic failover in the event of detectable hardware or software failure, leading to no loss in user service. As explained above, this is not easy to achieve with the current reference implementation of Shibboleth. Lower resilience levels involving redundant components or the use of standby servers may be sufficient, depending on the criticality of the service concerned. In the case of Shibboleth, the project team has agreed that a production service for Cardiff will be best provided by automatic failover to a standby system to deal with the worst case scenario of the local Shibboleth server failing during the night or weekend when there are no support staff present.

The resilient features proposed fall into two main areas; application hardening and hardware resilience.

Hardening the implementation

The following steps are recommended to harden the reference implementation:

- Perform user authentication in an Apache module rather than in Servlet code, this can be either through the use of basic authentication over SSL, or use of a custom module. Apache authentication can be referenced to an LDAP directory using Apache Modules. The authenticated user would then be passed through to the Handle Service .
- Enhance error reporting in the java code.
- Implement additional restrictions on access to the Attribute Authority.

Resilient architecture

For production implementation across the University, resilience of the solution becomes vital. This needs to be addressed in two ways:

1. System capacity to handle load.
2. Ability to seamlessly handle hardware, software or communications failure.

System capacity is not expected to be great since it is only called into operation when a user first accesses a Target site. The production system will be tested at authentication rates described above, and it is expected that a single server will be sufficient to cope with the predicted load.

The ability to handle Hardware/Software/Communications failure will be addressed by:

- Redundant components in the production server.
- A Hot standby server, with automated failover controlled by a Layer 4-7 switch.
- Introduction of fault tolerant links between elements of the implementation – e.g. the LDAP servers will also be front-ended by a Layer 4-7 switch to ensure that the failure of single LDAP server does not cause a halt in operations.

The Shibboleth AA appears to support connection pooling in the LDAP connections, and also to properly release contexts for the pool to manage. This would allow the pool to replace a bad connection with a good one.

A development server function will be required to provide a development and test environment for the Shibboleth service. It is likely that this will be implemented using a server virtualization product (such as VMWare) on the hot standby server to provide both standby and development functions in one physical server.

Redundant Components in the production server will include some or all of:

- Uninterruptible Power Supply (UPS)
- Dual hot-swappable power supplies
- Dual network interfaces
- Mirrored disks (RAID 1)

Monitoring for Resilience and Availability

The redundant components in the service must all be monitored so that failed components can be replaced before impacting service availability. A method of monitoring server function as opposed to major hardware failure will also be desirable to improve the range of failure modes that can be detected. Loading on the various critical components (such as CPU load) will also be recorded so that longer term trends can be identified before they impact on service availability.

Conclusion

The resilient pre-production design has been outlined and is now sufficient to specify the hardware to be ordered. Additional technical tasks have been identified (monitoring and the construction of a development environment).

APPENDIX A

Cardiff University Shibboleth Origin pilot Architecture report – Rob Hebron

Introduction

This document is intended to provide the basis of a pilot Shibboleth Origin implementation at Cardiff University. It seeks to :

1. Provide a brief project overview
2. Provide a brief overview of the Shibboleth architecture
3. Provide a system overview, including how the implementation will interact with systems already in place at the University
4. Indicate how the implementation can be integrated into possible future Single Sign On architectures
5. Indicate how the implementation can be integrated into a fault-tolerant architecture

Shibboleth is a federated authentication mechanism developed as an open standard by the Internet2/Middleware Architecture Committee for Education. It enables sites that hold user authentication details to securely verify the identity of individuals wishing to access content at another site that requires authentication. Core to the shibboleth design is that fact that user authentication tokens do not need to be replicated or reproduced at the content provider.

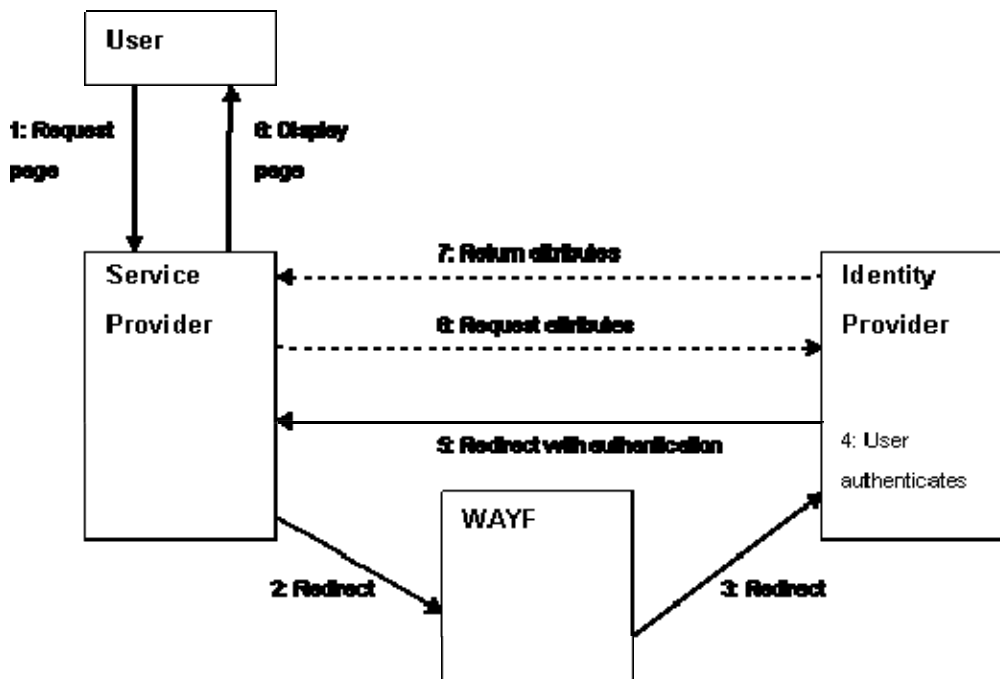
Shibboleth uses XML and, more specifically, SAML (Security Assertion Markup Language) for inter-server communications on user authentication and access rights. In addition, PKI (Public Key Infrastructure) is used to securely verify the participants in a Shibboleth federation. Reference implementations of the Shibboleth infrastructure are available, written in Java and running on Tomcat.

The Shibboleth implementation at Cardiff University is intended principally to provide authentication as an “origin” site to the Eduserv Athens content “target”. Athens is a JISC managed project to provide a single point of access to a wide variety of academic content. In

the medium-term , a Shibboleth implementation must integrate with a Cardiff University-wide single sign on and security system.

Shibboleth Architecture Overview

A simplified view of the Shibboleth architecture is reproduced from the MATU website below:

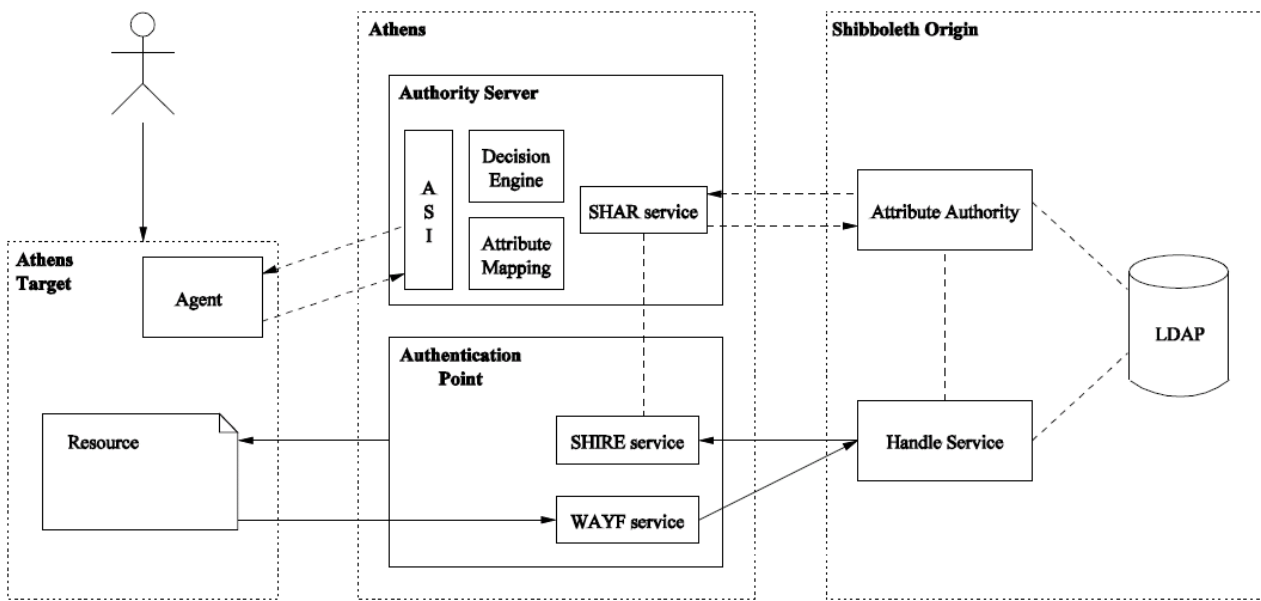


In the above diagram the following steps are described:

1. First of all, the user accesses a protected resource.
2. The resource redirects the user to the WAYF, so that he/she can select his home organisation. Depending on the policy of the federation, the user may be able to record this preference, perhaps in a cookie, for future use.
3. The user is then directed to his home organisation, which sends him to the authentication system for his organisation.
4. The user authenticates himself, by whatever means his organisation deems appropriate for this federation.
5. After successful authentication, a one-time handle or session identifier is generated for this user session, and the user is returned to the resource
6. The resource uses the handle to request attribute information from the Identity Provider for this user.

7. The organisation allows or denies the attribute information to be made available to this resource using the Attribute Release policy.
8. Based on the attribute information made available, the resource then allows or denies the user access to the resource.

A more detailed architectural overview is provided by the Athens Shibboleth documentation and is reproduced below.



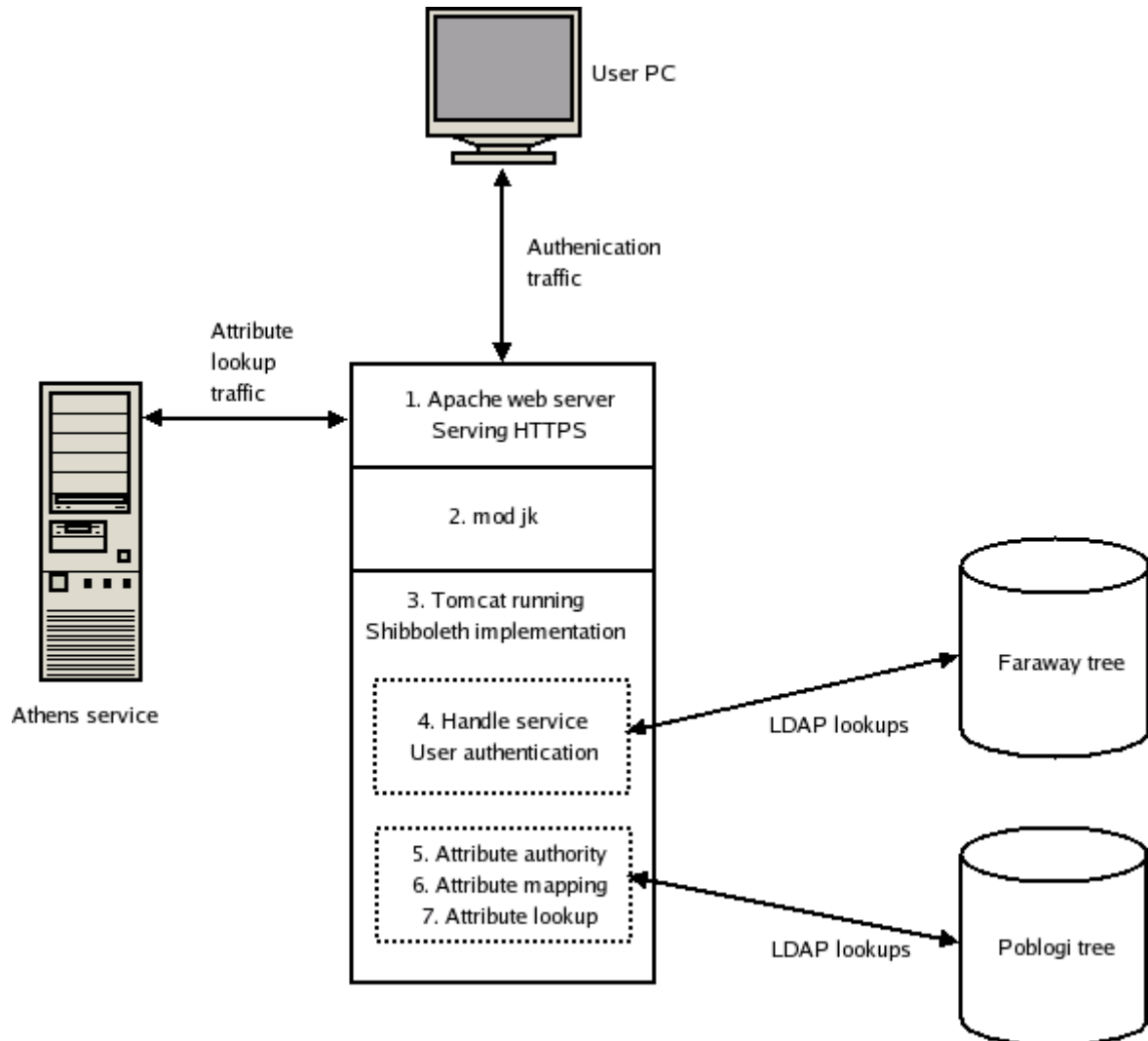
For more detailed Shibboleth architecture diagrams and terminology, see the `ng_core_architecture.pdf` file in the information pack (for Athens specific architectures) and documentation at <http://shibboleth.internet2.edu/shibboleth-documents.html>

Cardiff University implementation - system overview

From the Shibboleth architectural diagrams shown above, it is clear that the following need to be implemented at Cardiff University:

1. Handle service, including authentication of users
2. Attribute authority to provide attribute information on authenticated users to the Target site according to locally held policies
3. Means of connecting to existing source(s) for user authentication and attributes

Option 1: The diagram below shows a system implementing the above with no changes required in other infrastructure:



The above diagram describes the following elements

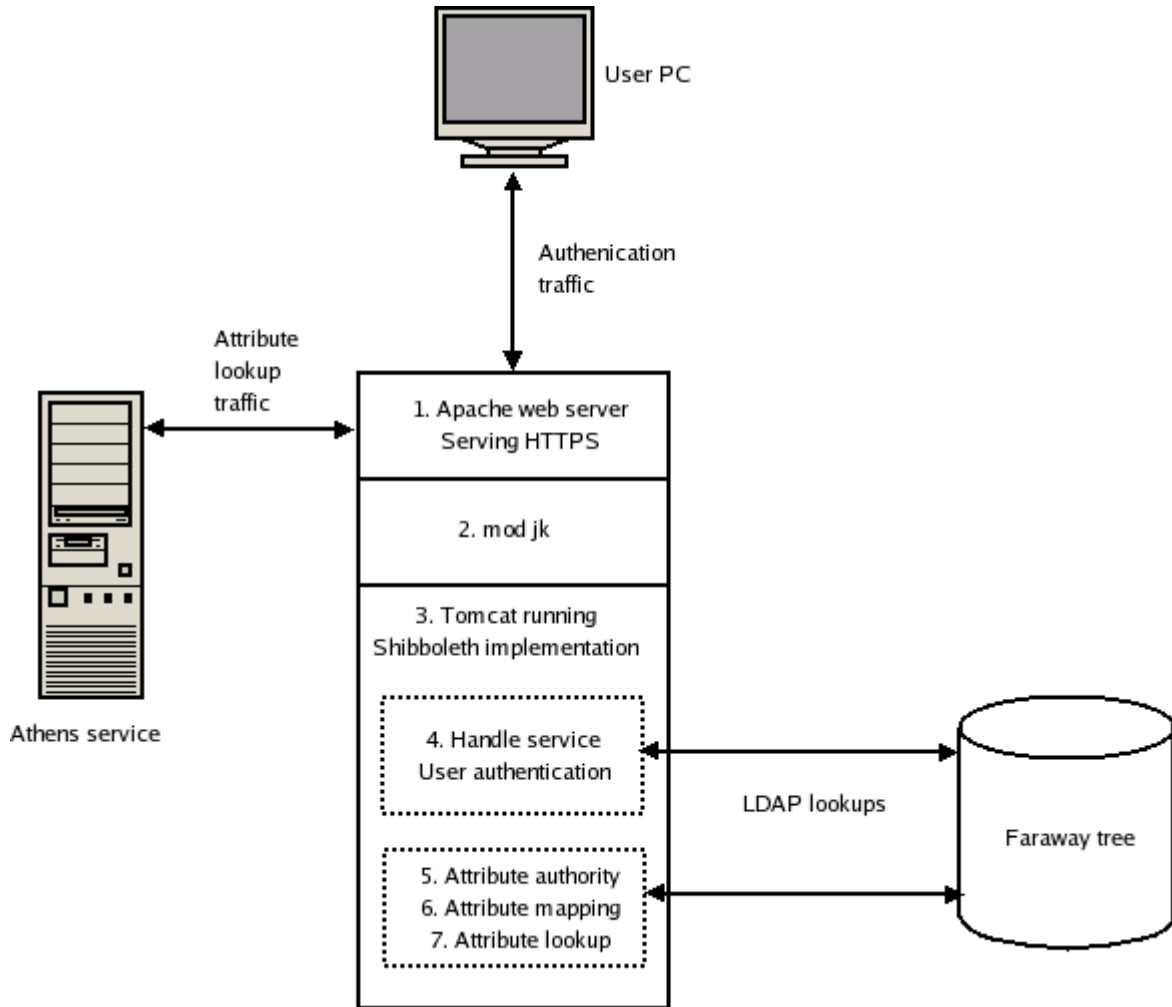
1. Apache webserver serving client requests over HTTPS
2. Forwarding of requests to/from Tomcat using modjk
3. Shibboleth implementation running in Tomcat Servlet Container
4. Handle service interfacing with Faraway tree for user authentication using LDAP
5. Attribute authority interfacing with Poblogi tree using LDAP for user attribute queries
6. Mapping of eduPerson attributes to equivalents held in Poblogi
7. Lookup mechanism to populate attribute values for those not available from Poblogi tree – e.g. Athens specific departmental descriptions

Linux is the recommended platform for running the implementation

Potential issues

- The attribute retrieval from the Attribute Authority to the Pologgi tree is included in the reference implementations, but assumes a one-to-one mapping between user accounts and objects in the LDAP directory. With the Pologgi tree one-to-many mappings exist, with a single identity mapped to more than one real account. This introduces complexity into the lookup process.
- While attribute mapping is included in the Shibboleth implementations, attribute lookup is not. For this reason, extension of the Pologgi schema to hold the attributes values which would otherwise require lookup should be considered.
- A possible solution to both of the above issues is to use custom code to perform the LDAP attribute retrieval. This would be queried by the attribute authority through a standard interface, and would in turn query the Pologgi tree and add any missing values from lookup tables. Since this would exist outside of the Shibboleth implementation, long term maintenance would not be a major issue provided a standardised interface is used between the attribute authority and the additional code.
- Current Athens extract determines user's department based on account context in JCCS tree (available in Pologgi attributes). These may have out of date names and/or may be incorrect for users. A decision is required regarding whether this should be continued, or new departments used instead.

Option 2: The diagram below shows an alternative design where changes to existing infrastructure are required:



The above diagram describes the following elements

1. Apache webserver serving client requests over HTTPS
2. Forwarding of requests to/from Tomcat using modjk
3. Shibboleth implementation running in Tomcat Servlet Container
4. Handle service interfacing with Faraway tree for user authentication using LDAP
5. Attribute authority interfacing with Faraway tree using LDAP for user attribute queries
6. Mapping of eduPerson attributes to equivalents held in Faraway tree
7. Lookup mechanism to populate attribute values for those not available from Pologgi tree – e.g. Athens specific departmental descriptions

The following changes in existing infrastructure would be required:

- The Faraway tree would need populating with attribute values currently held in the Pobllogi tree. This would also allow all required attributes to be populated, removing the need for attribute lookup
- Population of the Faraway tree could be done through DirXML via the JCCS tree, or with a scheduled job.

Option 2 provides the following advantages over option 1:

- Simplicity
- Removal of solution-specific code
- Ability to replace the Shibboleth server implementation with a drop-in replacement such as iChain

Option 2 is therefore recommended.

Hardening the implementation

The following steps are recommended to harden the reference implementation:

- Perform user authentication in an Apache module rather than in Servlet code, this can be either through the use of basic authentication over SSL, or use of a custom module. Apache authentication can be referenced to an LDAP directory using Apache Modules. The authenticated user would then be passed through to the Handle Service
- Enhance error reporting in the java code
- Implement additional restrictions on access to the Attribute Authority

Alternatively, the reference implementation could be replaced with a commercial alternative, such as iChain. However, the following should be borne in mind when considering this:

- iChain supports SAML (v1), but does not implement the Shibboleth framework
- Significant custom development would be required to support the Shibboleth framework on iChain
- Athens provides a SAML gateway in addition to the Shibboleth gateway, so the lack of support of the Shibboleth framework is not a show-stopper
- Both SAML and Shibboleth are emerging technologies and it is not believed that the

Novell iChain SAML extension would be as easy to debug and alter as an open source based solution. It should be noted that while the Novell SAML toolkit is released under an Apache style open source license, the SAML extension for iChain is not

Single Sign On integration

Single sign on architectures are briefly described in the definitions section below.

The solution described above could be integrated with a reverse proxy solution through:

- The use of basic authentication on Apache
- Basic authentication header injection on authenticated traffic passing from the reverse proxy to the Apache web server

The solution could be integrated with a central authentication server & ticketing system through:

- Use of Apache module within the solution to verify user authentication status with the authentication server and redirect to the authentication mechanism if the user is not already authenticated
- Use of LDAP authentication against the Faraway tree on the authentication server

Alternatively, a reverse proxy solution such as iChain that supported SAML/Shibboleth could be implemented.

Resilient architecture

For production implementation across the University, resilience of the solution becomes vital. This needs to be addressed in two ways:

- System capacity to handle load.
- Ability to seamlessly handle hardware, software or communications failure

System capacity is not expected to be great since it is only called into operation when a user first accesses a Target site. It can be enhanced through:

- Hardware capacity
- Use of load balancing through Layer 4-7 switching. This would not be simple to implement for reasons outlined below

The ability to handle Hardware/Software/Communications failure could be addressed by:

- Introduction of multiple servers running the Shibboleth Origin implementation
- Introduction of fault tolerant links between elements of the implementation – e.g. to ensure that the failure of single LDAP server does not cause a halt in operations
- Use of fault tolerant network architecture to route traffic intelligently to multiple servers in either an active/passive or active/active load balanced architecture

The use of load balancing between multiple Shibboleth servers introduces the problem of handle sharing between Shibboleth servers (handles being what are used to identify an authenticated user in the Shibboleth framework). This is significant because:

- Traffic comes from two sources – the client and the Shibboleth Target. These may be load balanced to different servers with a user authenticating on and receiving a handle from one server, and the Target site requesting attributes from another server that is unaware of the handle. This would result in attribute lookup failure. It may be possible to address this by load balancing based on handles issued, but without handle sharing being available, effectively means that an active/passive model must be used with load balancing not possible
- A Shibboleth server may fail between issuing a handle to an authenticated user and receiving attributes requests from the Target site. The request may be re-routed to another Shibboleth server, but this would be unaware of the handle. A Single Sign On architecture may hide the impact of this to the user by automatically re-logging on, but it could result in a increased initial load on the surviving Shibboleth server, and may result in session loss at the Target site . The impact of this is mitigated by the fact that the Shibboleth server is only involved when the user first attempts to access the Target site

The long term solution to the handle-sharing problems outlined above would be to introduce a session sharing mechanism into the implementation in order to provide a properly clustered service.

It should be noted that the above holds true for and iChain/SAML based implementation as well as Shibboleth.

Definitions

Shibboleth is an architecture that enables organisations to build single sign-on environments that allow users to access web-based resources using a single login. Shibboleth uses open standards (such as [SAML](#)) and was developed by the [Internet2 middleware group](#). OpenSAML, which is used by both the Internet2 and Eduserv/Athens implementations of the Shibboleth architecture is an open source implementation of SAML.

A **Shibboleth Target** holds content (text, documents, images, sound, video etc) that users wish to use, but are required to authenticate in order to do so. The Eduserv/Athens documentation also refers to a Target as a **Service Provider**

A **Shibboleth Origin** holds authentication details and mechanisms and has access agreements with one or more Shibboleth Targets. The Shibboleth Framework provides a means for Origin Sites to authenticate users wishing to access a Target, communicate this fact to the Target through user of a one-time token, and release attribute information about the user when queried by the Target. Attribute release is controlled by policies at the the Origin site in order to preserve privacy. The Eduserv/Athens documentation also refers to an Origin as a **Identity Provider**.

The **SAML** (Security Assertion Markup Language) framework is a standard ratified by OASIS that facilitates the exchange of authentication and authorisation information across disparate systems. This means service providers (the owners of web-based resources) can query identity providers (the owners of data about users) for information about their users and grant access based on that information. The framework consists of an XML language that defines the structure of messages that are exchanged by systems to share user data. The framework also defines rules about the content of these messages and how they should be exchanged.

SOAP (Simple Object Access Protocol) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.

The Hypertext Transfer Protocol (**HTTP**) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP has been in use by the World-Wide Web global information initiative since 1990.

HTTPS is a secure version of HTTP with an added layer of data encryption. Typically this uses public key infrastructure (PKI) cryptography for an initial key exchange between client and server, and symmetric encryption/decryption using this key for subsequent communications. Most HTTPS traffic uses the Secure Sockets Layer v 3 protocol, originally developed by NetScape. Since this was proprietary, Transport Layer Security has been developed as an alternative, but is far less widely used.

Web Servers are used to serve HTTP and HTTPS traffic (other protocols may also be served) to clients. Typically they can be extended from their original purpose of serving static content to become the front end for sophisticated dynamic back end applications. The leading web server in use today is Apache, from the Apache Software Foundation

A **Servlet Container** is used to serve web applications programmed in Java. Both the Internet2 and the Eduserv/Athens reference implementations of Shibboleth require servlet containers. There are numerous servlet containers available, some with additional Java 2 Enterprise Edition functionality, however the most widely used is Tomcat, an open source servlet container from the Apache Software Foundation.

An **Authentication Mechanism** is a generic term describing a framework which enables users (or automated processes) to be challenged to prove that they are who they are, and provide tokens to prove this. The most common is the user name/password combination. A simple implementation of authentication is a web server requiring the user name and password to be entered in a browser through the Basic Authentication mechanism, these are then encoded and sent to the server and checked against a static file containing user names and passwords. More sophisticated frameworks introduce encryption, the ability to check user credentials against existing organisational authentication authorities (such as LDAP directories). Further extensions may introduce the use of smart cards or biometrics.

Single Sign On is another generic term used to describe an architecture whereby a user is only required to log on once in order to access varied back end services, each of which has an authentication mechanism. There are three main groups of SSO architectures:

1. User name/password caching where the same user name/password pair are the same for accessing each back end resource. Typically a reverse proxy application sits between the user and the web servers hosting the secured resources. The user logs onto the reverse proxy and is then transparently logged on to back end web servers through basic

authentication header injection or form fill. The advantage of this is that the applications rarely need recoding.

2. User name/password caching where different user names and passwords are used for accessing back end resources. Typically a reverse proxy sits between the user and the back end web servers. The first time a user accesses a protected resource a challenge is sent to the user to enter a user name/password pair, which is then cached on the reverse proxy and used to provide the illusion of Single Sign On on subsequent requests to the back end server.
3. Centralised authentication servers are used in architectures where the user is challenged to authenticate on first access to any protected resource, and get issued a token (typically in the form of a cookie) when properly authenticated. This token is then presented to back end servers requiring authentication. These servers contact the central authentication server to verify that the token is valid. The principle advantage of these architectures is that user names and passwords are not stored anywhere between the client and the back end server, and for this reason they are considered the most secure option. The principle disadvantage is that the authentication mechanisms on the application servers require modification.

Guide to information pack prepared for Cardiff University

The information pack has principally been gathered from the following sites:

- Internet2 Shibboleth site at <http://shibboleth.internet2.edu/>
- Eduserv/Athens Identity Manager Shibboleth implementation at <http://www.athensams.net/shibboleth/AthensIM/index.html>
- Middleware Assisted Take-Up Service at <http://www.matu.ac.uk/>

In general, the quality of the documentation is fair for a project at this stage of development. There are internal inconsistencies and contradictions but, since the content is available from multiple sources, these are reasonably easy to resolve.

The reference implementations released by Internet2 and Athens should be considered beta quality and require testing and hardening to make either sufficiently stable and secure to be used in a production environment. Some guidance as to how this may be achieved is given in the documentation and earlier in this document.

In general, the Internet2 reference implementation is more mature than that from Athens. The latter is missing some key files from the source distribution (the only one valid for implementation in a production environment) and has only partial documentation. A working installation of the Athens implementation could only be achieved with close reference to that from Internet2.