

**Flexible Access Middleware Extensions to PERMIS (FAME-PERMIS): Final Report**

**Name of lead institution/organisation:** The University of Manchester/School of Computer Science

**Authors:**

Dr Ning Zhang, Dr Aleksandra Nenadic, Professor David Chadwick, and Dr Andrew McNab

**Full contact details for primary contact:**

**Name:** Dr Ning Zhang (the Project Manager)

**Position:** Academic Staff – Senior Lecturer in Distributed Systems Security

**Email:** [nzhang@cs.man.ac.uk](mailto:nzhang@cs.man.ac.uk)

**Address:** Room KB2.113, School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, England, UK.

**Tel:** (+44) (0)161 275 6117

**Fax:** (+44) (0)161 275 6204

**Web address of the project website(s):**

<http://www.fame-permis.org/>

**Project partners and outline project description:**

The FAME-PERMIS project is aimed at designing and developing middleware extensions to facilitate authentication strength linked flexible and fine-grained access control. The FAME-PERMIS middleware extensions integrate multiple authentication services that support the use of a wide range of authentication credentials including IP addresses, username/password pairs, certificate-based soft tokens, Java/smart cards in a single-factor or multi-factor fashion to achieve specific *authentication strength*, or *Level of Assurance* (LoA). This LoA is then fed into an authorisation decision engine, such as PERMIS, to decide the users' privilege rights. The extensions also provide Single Sign-On (SSO), extend the Shibboleth's Handle Service to include the LoA attribute for users, and use SAML as a message format for passing the user authentication information and attributes to Service Providers enabling secure but flexible Web based VO resource sharing. The middleware extensions consist of three parts, FAME, PERMIS and GridSite.

- **The School of Computer Science at University of Manchester (CS-MAN):** responsible for the design and development of the FAME system that integrates multiple standard web-based authentication services, derives a LoA based upon the authentication service/token used, and passes it to an authorisation decision engine using the Shibboleth infrastructure. In addition, FAME supports SSO.
- **The University of Kent (UoK):** responsible for the strengthening of the PERMIS authorisation decision engine by including the LoA value in its decision making process, so as to enable LoA-linked access control.
- **The High Energy Physics Group at the University of Manchester (HEP-MAN):** responsible for the extension of GridSite so as to provide support for XACML in GridSite alongside GACL, integrate FAME with GridSite, both for credentials obtained via Shibboleth with direct LoA information provided by FAME, and for existing X.509 credentials, and support for Shibboleth credentials within GridSite.

## Final Report

# Flexible Access Middleware Extensions to PERMIS (FAME-PERMIS)

### Table of Contents

Table of Contents .....	2
Acknowledgements .....	3
Executive Summary .....	4
1. Background .....	5
2. Aims and Objectives .....	5
3. Methodology.....	7
Overall Approach.....	7
Technical Design.....	8
Research .....	9
Case Studies .....	10
4. Implementation.....	11
Workpackages 1 & 2: FAME design and development.....	11
Workpackages 3 & 4: Adding LoA to PERMIS .....	12
Workpackage 5: GridSite extensions .....	14
Workpackage 6: Integration and case studies .....	19
Additional Contributions: FAME-CLEF .....	22
5. Outputs and Results.....	24
Project Website .....	24
Project Deliverables.....	24
Dissemination Activities.....	25
Publications .....	25
FAME-PERMIS Demo.....	25
Collaboration with NGS .....	26
6. Outcomes.....	27
7. Conclusions.....	28
8. Implications .....	28
Appendices .....	28
A1: FAME Design Final Report .....	28
A2: FAME User Guide.....	28
A3: GridSite User Guide.....	28
References .....	28

## **Acknowledgements**

The project is funded by the JISC Middleware Programme. We here would like to thank the JISC for the funding support, and the Programme Managers, Nicole Harris and James Farnhill, for their patience, support and understanding.

There are other people who made valuable contributions to this project directly and indirectly. We would like to particularly thank Li Yao for his excellent work on the design, and development of a PIN (Personal Identification Number) activated smart-card based authentication service, and Jay Chin for his technical help in managing the Shibboleth test-bed and contribution in extending the FAME solution so that it can also be used to protect stand-alone applications, in addition to protecting web-based applications. Special thanks also go to Prof. Alan Rector who has facilitated the involvement of the CLEF-services project into the case studies of the FAME-PERMISS project, and Prof Carole Goble for her in-kind support.

## **Authorship**

This report has been compiled from project deliverables and reports produced over the project period by:

- Ning Zhang
- Aleksandra Nenadic,
- David Chadwick,
- Andrew McNab, and

Ning Zhang  
Project Manager

## Executive Summary

Robust authentication and authorisation services are keys to the deployment of a secure virtual organisational (VO) environment where students, researchers, staff with different roles and responsibilities from different institutions are expected to share resources distributed in the Internet environment with components administered locally and independently. Entity authentication is the first line of defence in this VO environment, which is required to assure a service provider (SP) that a resource access is only granted to users whose identities have been verified. The level of assurance (LoA) in identifying a user, also referred to as the quality of an entity authentication process, reflects the degree of confidence in identifying the entity to which the credential was issued, and the degree of confidence that the entity using the credential is indeed the entity that the credential was issued to. Different authentication services and tokens provide different LoA. NIST (the US National Institute of Standard and Technology) [NIST06] has defined four levels of authentication assurance versus different authentication services and tokens.

Resources with varying sensitivity and/or risk levels are better served by different authentication methods. With this risk-based authentication/authorisation approach, an SP may specify a minimum LoA depending upon the resource sensitivity and/or risk levels, and require that the access is granted only if the LoA derived from an authentication instance satisfies the minimum LoA.

This project is aimed at designing and developing middleware extensions to realise this vision of risk-based authentication and LoA linked fine-grained access control. The extensions are in three parts, FAME [FAME], PERMIS [PERMIS] and GridSite [GridSite], linked by SHIBBLOLETH – an open source solution to support inter-institutional sharing of Web resources subject to access control [Shib04]. FAME integrates a wide range of standard authentication services that support the use of various authentication credentials including IP addresses, username/password pairs, and certificate-based soft as well as hard tokens such as smart/Java cards. FAME can easily be integrated with any authentication services with a Web front-end, e.g. Kerberos, NIS (Network Information Service), and authentication systems that use LDAP (Lightweight Directory Access Protocol) or Mysql. Upon successful authentication, FAME derives a LoA value based upon the authentication method/token used in the process, and passes it to an authorisation decision engine as a user attribute value, via the SHIBBLOLETH SAML message. PERMIS is a risk based authorisation decision engine that uses the LoA in the authorisation decision making process. PERMIS supports hierarchical role based access controls (RBAC). By converting the LoA into a user attribute, and defining the LoA as a hierarchical role in the PERMIS authorisation policy, PERMIS will make risk based authorisation decisions that are dependent upon the LoA value of the user. The PERMIS Policy Editor has been enhanced to allow managers to easily create policies that include the LoA as a hierarchical role. In other words, PERMIS now makes authorisation decisions based on the tuple: {Subject, LoA, Target, Action}.

The third component, GridSite, builds extensions to allow the integration of FAME-PERMIS into the GridSite infrastructure so that the GridSite community can also benefit from LoA lined fine-grained access control to Grid resources. Thus GridSite can not only use the X.509 DN to identify a user, but also, with the support for SHIBBOLETH, tie a username/password to that DN, using the certificate to create the account. The use of FAME in this context allows the representation of the quality of the authentication. The expressing requirements about this in GACL/XACML policies are also supported.

Both the software design and implementation adhere to relevant international standards [NIST06, Shib04]. Case studies are performed to ensure that the developed middleware satisfy both functional and non-functional requirements, including security, interoperability, portability, reliability, and usability. A live demo showing the integrated operation of FAME, PERMIS, GridSite, and SHIBBOLETH is available on the project website, <http://www.fame-permis.org/>.

The project has achieved all the aims and objectives initially set, i.e. those outlined above. In addition, the project also addressed an additional important issue that was identified during the design stage, i.e. the SSO (single-sign-on) property. Furthermore, we also identified an additional use case scenario during the case studies, in which a user may opt to use a standalone application (rather than a web browser) to access remote resources. Again, we have designed and built an extended solution making use of a software-configurable firewall. This extended solution, called FAME-CLEF, allows the use of FAME-PERMIS to protect resources accessed via standalone clients.

Last but not least, in addition to the software development, the project team has also published 5 academic papers, 3 in international refereed conferences and 2 in academic journals.

# Final Report

## Flexible Access Middleware Extensions to PERMIS (FAME-PERMIS)

### 1. Background

Robust authentication and authorisation services are keys to the deployment of a secure virtual organisational (VO) environment where students, researchers, staff with different roles and responsibilities from different institutions are expected to share applications, data and/or CPU cycles distributed in the Internet environment with components administered locally and independently. Authentication is the first line of defence in any secure systems, and it is particularly important in VO environments playing a critical role in the provision of a number of essential security services including authorisation, access control and accounting.

On the international front, VO, or Grid computing, security research has been mainly focused on inter-organisational entity authorisation and secure communication. The work on authentication has largely been limited to a certificate-based approach using X.509 identity certificates. The most notable grid security technologies and efforts are Grid Security Infrastructure (GSI) framework and Globus toolkits [Welc04], the Legion Security Model [Hump00], and the Shibboleth Project [Shib04]. The GSI/Globus API and the Legion Security Model adopt X.509 identity certificates, Public Key Infrastructure (PKI), and transport layer security protocol SSL/TLS (Secure Socket Layer/Transport Layer Security) to address the issues of authentication, single-sign-on, credential delegation, secure communication and identity mapping from a user in one domain to an account in another. Shibboleth does not directly address user authentication, but rather provides an open standard based protocol architecture for securely transferring a user's attributes from the user's home site to the remote resource site, and trusting the home site to correctly authenticate the user. The work on Grid authorisation has been much more disparate, and includes CAS, VOMS, PERMIS, CARDEA, etc. PERMIS provides a policy-based decision engine for user authorisation currently only based upon the attributes of three objects {Subject, Target and Action}, which provides no linkage between the user authorisation decision and the authentication assurance level.

Nationally, the current access management system available to the UK educational community is Athens that is a single-username/password solution. The existing middleware developed by the UK Grid community only supports the use of digital certificates in the form of software tokens. Both of these two systems are homogeneous in that a user has to obtain their respective authentication credentials in order to access the resources that they each manage. The next generation access control management system should at least be able to accommodate both of these authentication technologies, and preferably possess the following features:

- ◆ It should be heterogeneous supporting a range of authentication and authorisation technologies. In VO environments, users and service/resource providers are typically from different organisations, administrative domains or sites. The systems in different organisations and domains may use different security policies and employ different authentication technologies. In addition, users may wish to use wireless devices, such as laptops, PDAs (Personal Digital Assistants) and smart phones, to access the services, and authentication methods used by the wireless devices may vary. Thus there is a need for a dynamic authentication framework capable of supporting a wide range of authentication methods and devices.
- ◆ Authentication strength should be linked to access control decision making, and such a linkage is necessary for the provision of fine-grained access control and privilege allocation in VO environments in which different applications may have highly varied authentication requirements. Some services like e-catalogue services may be accessible to everybody who can be identified by the IP address of his/her machine, while others such as e-journal subscription or e-learning services may identify users based upon username/password pairs. Additionally, for the same user, it may be necessary to use authentication credentials with different strengths in different environments. For example, in a trusted environment such as a hospital, a medical doctor may use a weaker form of authentication to access his patients' records. But if the same doctor wishes to access the same patient records while he is on the train, then a stronger form of authentication credential should be used. Equally, a medical researcher uploading patient data into a central

biomedical data repository used for further research and treatment recommendation should use an even stronger form of authentication to ensure accountability and reliability of the data, in addition to data privacy and integrity.

- ◆ There needs to be support for user roaming.
- ◆ While several solutions exist to specify access policies in existing Grid projects, such as the Globus grid-mapfile, and XML policy languages from the PERMIS and GridSite projects, work is still needed to support general, standards-based interfaces. The use of proprietary access control interfaces presents particular problems for applications across institutional boundaries in some standard way. These policies must also support requirements on authentication strength set out above, as well as the identity asserted.

These observations illustrate that the current certificate-based "one-method-fits-all" authentication method and local-centred authentication policy is not suitable to the diverse and dynamic nature of VO environments and can not facilitate fine-grained access control, which takes into account the risk of falsified credentials. Furthermore, current systems cannot easily cope with dynamic configuration of policies (which can be in any language) and standard policies for passing between systems.

## 2. Aims and Objectives

The following details the project aim and objectives agreed at the start of the project.

This project is aimed at designing and developing Flexible Access Middleware Extensions to PERMIS (FAME-PERMIS) to facilitate uniform and multi-factor authentication and authentication strength linked fine-grained access control. It integrates different authentication methods with different levels of assurance and support the use of a wide range of authentication credentials including IP addresses, username/password pairs, certificate-based soft as well as hard tokens such as smart/Java cards. The extensions will be in three parts, FAME, PERMIS and GridSite, linked by SHIBBLOLETH – an open source solution to support inter-institutional sharing of Web resources subject to access control [Shib04]. The FAME system will be responsible for user identification/authentication via Web browsers, and the derivation of *authentication strength*, or *Level of Assurance* (LoA), based upon the authentication method/token used in the identification/authentication process. The LoA is then passed to an authorisation decision engine, such as PERMIS, via the SHIBBLOLETH SAML message. The latter then uses the LoA in the privilege allocation decision making process for the user. PERMIS will be enhanced to support LoA. In other words, PERMIS will now make authorisation decisions based on the tuple: {Subject, LoA, Target, Action}. GridSite extensions will be built so that GridSite users can also benefit from the LoA linked fine-grained access control provided by FAME-PERMIS.

To summarise, the principle objectives of the FAME-PERMIS project set in the original proposal are:

1. To develop the FAME login service supporting the integration of a wide range of authentication methods and services that use a variety of authentication tokens including IP addresses, username/password pairs, certificate-based soft and hard tokens such as Smart/Java cards. It will allow a Web browser user to authenticate her/himself using any one or more of these tokens, and secure and transparent local sharing of hardware authentication devices among wired as well as wireless terminals.
2. To implement the NIST recommendation [NIST04, NIST06] into the login service for the derivation of authentication strength, i.e. LoA, based upon the authentication token used in an authentication instance.
3. To develop APIs to serve authentication requests made through Shibboleth. The APIs will support two modes of working. In the first, the A/H/S may determine the LoA (or the authentication token required for the resource being accessed), and authenticate the user accordingly. In the second mode, the user is allowed to choose an authentication method, upon which the A/H/S calculates the strength prior to passing it to the authorisation decision engine.
4. To integrate the above components and produce our flexible authentication extension, the FAME system.
5. To develop a profile of the SAMLv2.0 protocol that allows us to pass the LoA in it (between the FAME and PERMIS systems).
6. To feed the LoA into PERMIS so as to enable authentication strength linked to access control.

7. To enhance the PERMIS Policy Editor to allow managers to easily create policies that include the LoA.
8. To enhance the PERMIS SAML interface to support the SAMLv2 profile developed in 5 above.
9. To enhance our existing Grid infrastructure to support PERMIS via its SAML interface, and to couple this with our existing fine grained access control.
10. To test and evaluate our FAME-PERMIS middleware solution using our Grid trial applications.
11. To report the work in writing to JISC.

The project has achieved the aims and objectives set in the initial project plan with the following changes.

We have re-named the FAME architectural components (but the project objectives and functionality of the components are not changed). Instead of calling them Device Manager, Network Manager and Fame Login Service, as detailed in the activities in Workpackage 1, and having separate APIs, as mentioned in Objective 3, the FAME components are now called FAME Single-Sign-On checker (F-SSO) and FAME Login Server (F-LS). These two FAME components, F-SSO and F-LS, jointly fulfil Objectives 1-4. This design revision is due to the observations: (1) different access devices actually use the same standard application for resource access, i.e. via the Web browser, and (2) SSO has been identified as an important and essential property for achieving security and usability. The revised FAME architecture is shown in Figure 1.

In addition, Objective 9 (Objective 6 in the original plan) has been adjusted owing to the following observations.

The original plan was to build SAML support into GridSite to support an external authorisation PDP such as PERMIS. However, since PERMIS was enhanced under the SIPS project to be called directly from Apache after Shibboleth authentication, there was no need to build the SAML support into GridSite. Instead PERMIS can be called directly from Apache. It was then relatively trivial to demonstrate PERMIS authorisation with GridSite instead of the current GridSite authorisation module. Combining the two authorisation modules together however depends upon which authentication method is used. This is because the way that Apache makes the authenticated name of the user available to the authorisation plug-ins is different for different authentication methods, e.g. Shibboleth and SSL client certificates.

In addition, the projects using GridSite are pursuing a model of X.509 user authentication and attribute certificates, GSI proxies and internal policy engines, and, in the case of EGEE, planning to add Shibboleth support. This model places more on support for Shibboleth, which JISC also wishes to support.

GridSite users use the X.509 DN to identify a user (and now with support for Shibboleth, it can tie a username/password to that DN, using the certificate to create the account), but the missing piece is a way of representing the quality of the authentication, and expressing requirements about that in GACL or XACML policies.

Therefore, in order to make the project outcome more beneficial to the GridSite user community, Objective 9 (i.e. Workpackage 5) has been adjusted to focus on the integration of GridSite, FAME, and PERMIS in the framework of Shibboleth.

### **3. Methodology**

#### **Overall Approach**

The project methodology involved divides the project work into three systems, with one person taking the lead for each: the FAME system (workpackages 1 and 2) was led by Ning Zhang, and developed by CS-MAN; the PERMIS extension (workpackages 3 and 4) was led by David Chadwick and done by UoK; the GridSite extension (workpackage 5) was led by Andrew McNab, and developed by HEP-MAN. We have spent 6 months for the system integration and case studies involving all the partners. This approach worked well and enabled us to fulfil our main objectives.

The critical success factors were correctly identified at the beginning of the project:

- We had ensured to employ RAs with proficient skills to work on the project.

- We have maintained good communication channels among project partners, especially during critical stages of design, components integration and case studies.
- We have also maintained good contacts with the JISC community, kept the community informed about the project progress, deliver the project deliverables on time, and actively attended all the programme meetings giving presentations and demos of the project output.
- We have been actively engaging with international standard bodies and organisations, such as OGF, noting any development in terms of SAMLv2, making sure that the design and implementation of the LoA passing mechanism are adhered to international standards.
- The NIST LoA definition is the only LoA technical definition that is currently available and well-known among international communities. To ensure software extensibility and interoperability, the design of the FAME architecture is such that FAME is not bound to any particular LoA definitions. In other words, FAME can accommodate any existing as well as future emerging LoA definitions or standards.
- We have employed security testing and case study methods to test and evaluate the software to ensure that the software developed satisfy both functional and non-functional requirements, including security, interoperability, portability, and usability.

In the following, we describe, in more detail, the methodologies used in terms of achieving security, usability, extensibility and interoperability.

### Technical Design

FAME integrates any authentication services with a Web-based front-end (i.e. authentication servers that can be invoked in the Apache server [Apache]). FAME does not re-implement existing authentication services<sup>1</sup>, but rather integrates itself with them and invokes them as configured in the Apache configuration file, *httpd.conf*. Currently, on our test-bed, FAME is integrated with six standard authentication services, namely, the Kerberos authentication service [Kerberos], LDAP authentication system [LDAP], SSL (Secure Socket Layer) certificate-based authentication service [SSL] with PKI credentials stored in Web browsers, and SSL certificate-based authentication service with PKI credentials stored in smart card tokens, the basic Apache username/password-based authentication system, and GridSite’s username/password-based authentication service, run at the GridSite’s location (HEP-MAN).

FAME supports SSO. FAME provides the SSO functionality through the use of a number of FAME tokens as outlined in Table 1. These tokens are generated from cryptographic hash functions, and are sent over SSL channels being protected using cryptographic keys. They are transferred among FAME architectural components (F-SSO, F-LS, and authentication services) either as cookies or URL parameters. Cookies are only used for passing information between two FAME internal components (namely, F-SSO and F-LS), while URL parameters are used to pass information between FAME internal component, F-LS, and external authentication services. A more detailed explanation of the contents of the tokens is given in the Appendix A1 – FAME Design Final Report.

Table 1: FAME Tokens

Token name	Passed From → To	Token protection	Token description
request-url cookie	F-SSO →F-LS	The token is passed only over an SSL-protected channel.	Generated by the F-SSO and scoped for the use by the F-LS. It contains the address of the originally requested url (of the Shibboleth’s HS), which was intercepted by the F-SSO.
sso cookie	F-LS →F-SSO	The token is passed only over an SSL-protected channel. It also contains a “cryptographic signature”	Generated by the F-LS and scoped for the use by the F-SSO after the user has successfully authenticated in order to implement SSO for subsequent resource

<sup>1</sup> With the exception of smart-card authentication service; this was re-implemented by the FAME team, as no open-source implementation was available.

		created as a keyed hash of the token data with a secret key known only to the F-SSO and F-LS.	accesses by the user via FAME. This token is not used by any external component.
<b>auth-request</b> token	F-LS → AS	Encrypted with the secret key shared between the F-LS and the AS and is only passed via an SSL-protected channel.	Generated by the F-LS and sent to the AS when requesting user authentication with the AS. It contains a random challenge <i>RC</i> that is used to authenticate the AS to the F-LS when response from the AS is received in the form of <b>auth-reply</b> token.
<b>auth-reply</b> token	AS → F-LS	Encrypted with the secret key shared between the F-LS and the AS, and only passed via an SSL-protected channel.	Generated by the AS once the user has been successfully authenticated and before the user is directed back to the F-LS. It contains a random challenge (that equals to the random challenge received from the <b>auth-request</b> token incremented by 1, i.e. <i>RC</i> + 1) and the <i>userID</i> of the authenticated user.
<b>auth-control</b> cookie	F-LS → F-LS	The token is passed only over an SSL-protected channel. It also contains a “cryptographic signature” created as a keyed hash of the token data with a secret key known only to the F-SSO and F-LS.	Generated by the F-LS and scoped for the use of the F-LS only. It contains the same random number sent to the AS in <b>auth-request</b> token and is used to verify the authenticity of the AS’s <b>auth-reply</b> token.

FAME derives a LoA based upon the authentication service and credential used in an authentication instance. For this purpose, FAME implements the NIST LoA definition (as this is the only standard available at the moment). However, FAME is not bound to the NIST definition. Any future emerging LoA definitions or standards are readily supported by FAME.

In addition, as described in the original proposal, FAME supports two modes of working. In the first, the IdP may determine the LoA (or the authentication token type as required for the resource access), and authenticate the user accordingly. In the second mode, the user is allowed to choose an authentication method, upon which FAME calculates the LoA prior to passing it to the authorisation decision engine. FAME achieves the first mode by only presenting the users with the authentication methods that satisfy the minimum LoA requirements as imposed by an SP, and achieves the second mode by default.

**Research**

In the original design, we mentioned that we would investigate the best ways of passing LoA (Level of authentication Assurance) from the Shibboleth IdP to the Shibboleth target (and from the authenticating service to the resource PDP in general). Since then, we have had discussions inside the GGF and with XACML experts and have picked up newer developments in the OGF community. Through these discussions and developments, we have seen there are two ways of achieving this. One is to encode the LoA as a user’s attribute and to pass it along with the user’s other attributes. The other is to pass it as part of the SAMLv2 protocol. The former method is preferred for two reasons. Whilst SAMLv2 supports the passing of authentication parameters as separate fields in the protocol, earlier versions of SAML and XACML do not support this. However, both support the passing of arbitrary user attributes in the request, as does SAMLv2. Therefore passing LoA as a user attribute will be transparent to the protocol that is actually used between the client and the server, and the PEP and PDP. This will necessitate adding the LoA attribute into the attribute policies of the target and origin sites, so that it can be requested and returned.

There are two alternative ways of implementing LoA attribute passing: the use of LDAP or CustomDataConnectors. With the LDAP approach, after the authentication server has finished

authenticating the user, it will add (actually replace) the current value of the LoA attribute in the LDAP entry of the user. Then when the Shibboleth AA server receives the SAML request, it will pick up the LoA attribute along with all the other ones and return them in the SAML response. A complication arises if the origin site is storing attribute certificates instead of plain vanilla attributes and these are being transferred by Shibboleth. In this case, we will need to enhance the PERMIS SAAM implementation to accept both attribute certificates and attributes in the same request. We are now looking at an enhanced design for SAAM (and the PERMIS call to GetCreds) that will support this. This is ongoing work and will be continued after FAME has completed.

In the second approach, a CustomDataConnector (CDC) can be built between the F-LS and the Shibboleth AA. The Shibboleth AA can be configured to release multiple attributes from multiple data-connectors, so that when a Shibboleth target SHAR requests the attributes of a user from the origin's AA, the AA sees what attributes should be released, and looks up what data-connectors are needed for this. The AA contacts each of these data-connectors in turn to release the necessary attributes. It then merges all the attributes together prior to returning them to the SHAR. The CDC facility is a standard Shibboleth extensibility feature for retrieving attributes from a data provider that does not have a standard data-connector implementation (i.e. LDAP, SQL database, etc) that come with the Shibboleth software. By building a CDC to the F-LS we will be able to return the LoA to PERMIS via the SAML response, without needing to store it temporarily in the user's LDAP directory entry.

We have chosen to implement the first, i.e. the LDAP, approach. This is because storing the LoA attribute together with the user's other attributes in the Shibboleth default store (LDAP), Shibboleth can be configured to pick up the LoA attribute via its usual configuration files (resolver.xml and arp.site.xml) in the same way as it is configured to pick up the other attributes. With this approach, all the user's attributes are stored at the same place and no further plug-ins are required to pick up the LoA attribute. So this approach is the cleanest solution. We have also had discussions within the OGF about the successor to the OGSA Authorisation protocol which is based on SAMLv1.1. When the FAME-PERMIS proposal was being written we anticipated this would be the SAML2 protocol, after it had been released. However, there appears to be strong support to replace SAML1.1 by XACML within the OGF. This case supports the first approach mentioned above, i.e. including the LoA as a user attribute. In this way FAME-PERMIS will be impervious to changes in the OGSA Authorisation protocol.

### **Case studies**

The project team spent six months on the integration of FAME, PERMIS, and GridSite in the Shibboleth infrastructure and on the case studies to test the interoperation and interoperability of the software developed. The test-bed runs a Shibboleth Identity Provider (IdP) service protected by the FAME in the School of Computer Science at the University of Manchester. The six standard authentication services that are integrated into the test-bed are the Kerberos authentication service, LDAP authentication system, SSL certificate-based authentication service with PKI credentials stored in Web browsers, and SSL certificate-based authentication service with PKI credentials stored in smart card tokens, the basic Apache username/password-based authentication system, and GridSite's username/password-based authentication service (the last authentication service is run at the GridSite's location, in the School of Physics at the University of Manchester).

The test-bed has two Shibboleth Service Providers (SPs); one is installed with the PERMIS authorisation engine, and is run at the Computing Lab, the University of Kent at Canterbury, and the other is installed with the integrated GridSite and PERMIS authorisation engines, and is run at the School of Physics, the University of Manchester. Both SPs control the accesses to their respective resources using users' attributes, including the additional LoA attribute, released by the FAME-enabled IdP via Shibboleth SAML messages upon the users' successful authentication.

In addition to joining the federated WAYF (Where Are You From) service run by the Internet2 [Internet2], we have also implemented our own WAYF service at UoM-CS for the purpose of these case studies.

Using this inter-site test-bed environment, we have designed three test scenarios, Test 1 tests the FAME-Shibboleth-PERMIS integration, Test 2 tests the FAME-Shibboleth-GridSite integration, and Test 3 tests Shibboleth-FAME-PERMIS-GridSite integration. We have also devised and used a number of allowable combinations of GridSite and PERMIS authorisation, as shown in the table below.

Table 2: Allowable combinations of GridSite and PERMIS authorisation

Allowable combinations	PERMIS Authorization	GridSite Authorization	Both Authorization
Apache Un/PW authentication	✓	✓	✓
X.509 PKC authentication	X	✓	X
Shibbolized FAME authentication and IdP attribute push	✓	✓	✓
Shibbolized FAME authentication and PERMIS AC pull	✓	X	X
Shibbolised FAME authentication and GridSite credential pull	X	✓	X
Shibbolised FAME authentication with PERMIS and GridSite credential pull	✓	✓	✓

These test scenarios are used to evaluate the correct functioning of each of the three systems and their mutual interoperability.

#### 4. Implementation

The project has three partners working on the three systems: FAME (Workpackages 1 and 2) by CS-MAN, PERMIS LoA extension (Workpackages 3 and 4) by UoK, and GridSite (Workpackage 5) extensions by HEP-MAN. These three systems are managed by Ning Zhang, David Chadwick, and Andrew McNab, respectively. We had regular project meetings, and lots of email communications among the project members for project coordination, management and discussions. We have spent the last six months of the project duration working closely on components integration and case studies (Workpackage 6).

##### Workpackages 1 and 2: FAME design and development<sup>2</sup>

The FAME architecture, as shown in the figure below, consists of the following components: a User-Agent, two internal FAME components and a set of external Authentication Services (ASes). The User-Agent (UA) is a Web browser that supports the use of cookies. The two internal FAME components are:

- (1) FAME SSO Checker (F-SSO), and
- (2) FAME Login Server (F-LS).

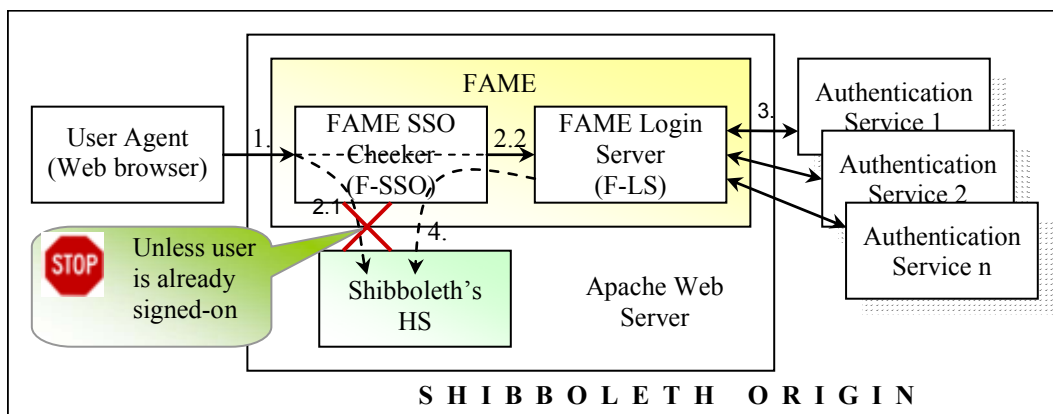


Figure 1: FAME architecture and components

The FAME SSO Checker (F-SSO) plays the role of a gate-keeper by controlling the access to the Shibboleth's HS (Handle Service). If a user has not been authenticated in the current session (as

<sup>2</sup> More design and development details are provided in the document (see Appendix A1): FAME Design Final Report.

determined by the absence of an **sso** cookie in the user's Web browser) the user will be redirected to the FAME Login Server (F-LS) and be forced to go through an authentication process. Otherwise, if the user has already been authenticated and issued with an **sso** cookie by the F-LS previously during this session, the access to the HS will be granted by the F-SSO and the current Shibboleth session will continue without re-authentication provided that the cookie is still valid.

The role of the F-LS is to redirect users' requests received from the F-SSO to an Authentication Server (AS) supported by the IdP (Identity Provider). If the IdP supports two or more ASes (the types of ASes supported is determined by the site's authentication policy and resource access policy), then the FAME interface window will allow the user to choose one from the list of ASes that are supported and/or satisfy the minimum LoA as required by the resource access, and the F-LS will redirect the user to the chosen AS. If the user fails the authentication process with the chosen AS, it is up to the AS to display an error message to inform the user. On the other hand, if the authentication is successful, the user will be redirected back to the F-LS by the AS. Back at the F-LS, the user will be issued with an **sso** cookie, which is scoped for use by the F-SSO only. The cookie contains a *userID* (an identifier of the user) received from the AS and the LoA derived by F-LS based upon the authentication method/token used by the user in this authentication instance. After the **sso** cookie is generated, the F-LS will redirect the user back to the URL originally requested by the user, i.e. the url of the Shibboleth's HS. This time, when the user tries to access the HS, the request will again be intercepted by the F-SSO. However, as now the user's request contains the **sso** cookie that has just been created by the F-LS, the F-SSO will let the user through to the HS. In addition, the F-SSO will populate the environment variable REMOTE\_USER with the value of the *userID* (contained in the **sso** cookie). The HS will use this identifier to identify the browser user, and to generate a handle for the user.

As mentioned in Section 3, FAME can be integrated with any authentication service that has a web front-end. Though the main focus of FAME is not on the re-implementation of currently available authentication services but rather on the integration of these services to support multiple and multi-factor authentication, we have also developed a custom-built Java smart card based Authentication Service for FAME software testing and evaluation as it is not available in existing open source arena.

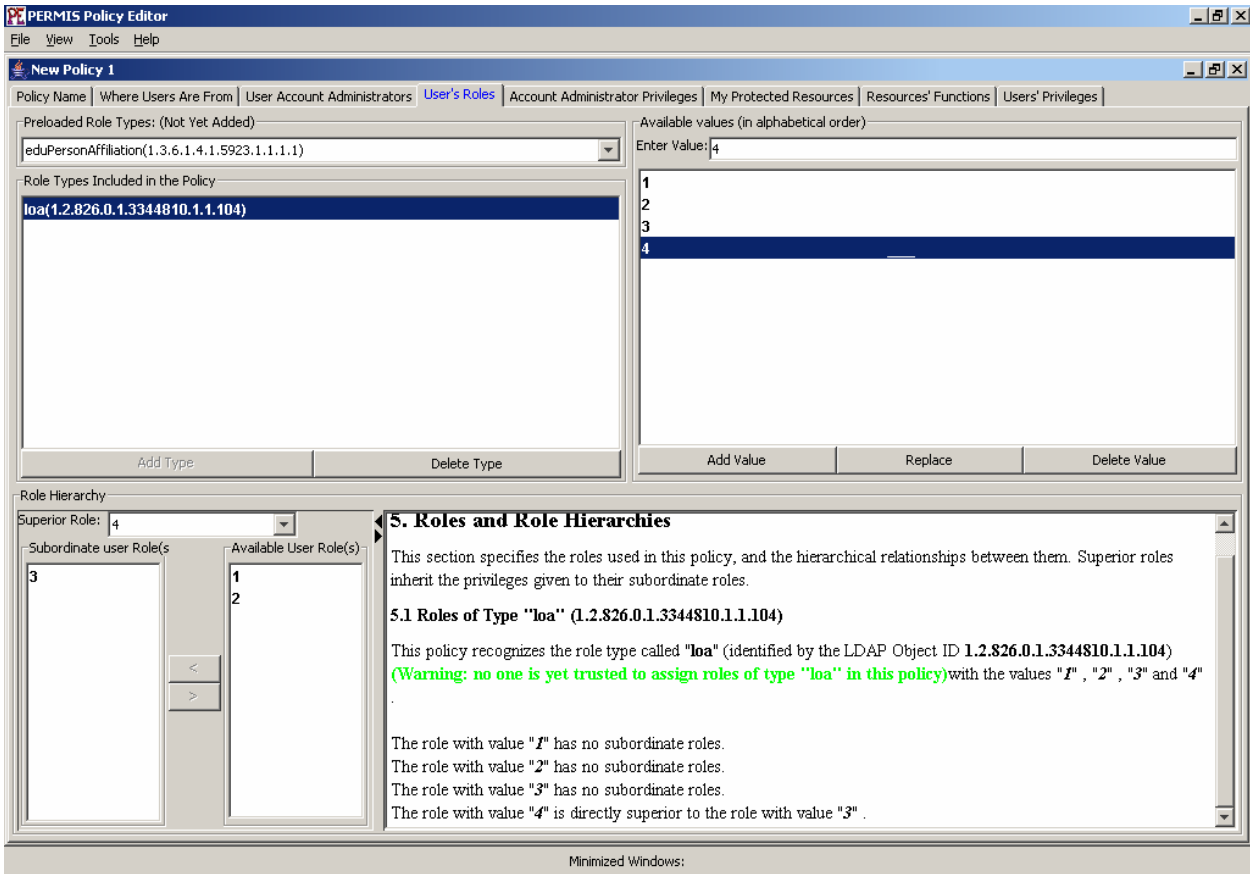
FAME is an Apache mod\_perl module, called Fame.pm, written in Perl. The integration with the Shibboleth HS, which is a Web servlet running inside a Tomcat-Apache Web Server assembly, is straightforward and proceeds as the integration of modules inside the Apache web server. The Fame.pm module is responsible for the authentication of a Web request (e.g. for accessing Shibboleth HS), and is invoked automatically by Apache for locations that use FAME for authentication protection. So the Shibboleth's HS should be configured to use the FAME module, Fame.pm, if it is to be protected by FAME.

It is worth noting that the issue of SSO (Single-Sign-On) was not mentioned in the original proposal. It was identified during the FAME design stage as an essential and important property to support user friendly Web accesses. We have taken this property into the design and implementation of FAME.

### **Workpackages 3 and 4: Adding LoA to PERMIS**

The original objective of WP3 was to develop a profile of the SAMLv2.0 protocol that would allow us to pass the LoA between the FAME and PERMIS systems. Due to the slow realization of SAMLv.2 into products and tools, we devised an alternative solution whereby the LoA was passed to PERMIS as an LDAP attribute along with the other subject attributes.

WP4 added built-in support for LoA to PERMIS. The main work was to redesign the PERMIS Policy Editor to allow the LoA to be easily specified as a hierarchical role attribute (see Figure 2 below), and to add a configuration capability so that the LoA could be preconfigured into the system (see Figure 3).

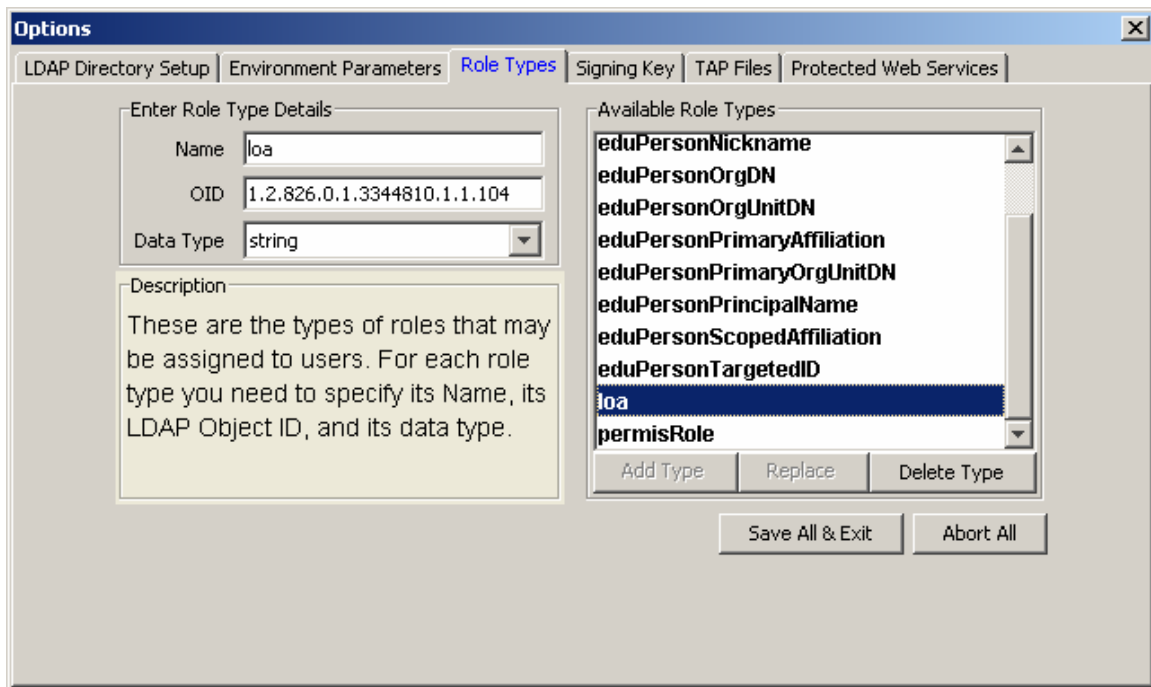


**Figure 2: The Policy Editor screen for including the LOA in a policy as a hierarchical attribute**

Key features to note from the Policy Editor screen-shot are:

- At the top left hand side, a picking list of *Preloaded Role Types (Not Yet Added)*, which lists the attribute and role types that have been configured into the Policy Editor tool, via the config screen shown in Figure 3, but that are not yet part of the current policy.
- A window immediately below the picking list, showing the attribute/role types that have been included in the current policy (the current window shows the loa attribute).
- At the top right hand side a field for adding values for an included role/attribute type, and beneath it a window showing which values have already been added for this type (the LOA values of 1 to 4 have been added to the current policy).
- At the bottom left hand side 3 windows for defining the role value hierarchy. The 3 hierarchy windows allow any complex value hierarchy to be constructed by picking a superior node in the hierarchy and then choosing which other values are directly subordinate to it. Once a value has been chosen it is removed from the set of available values, thereby stopping invalid hierarchies from being created.
- At the bottom right hand side a natural language description of the current role hierarchy policy. This is dynamically created from the underlying XML policy, and is updated every time the user changes any of the role values in any of the role types included in the policy. This allows the user to continually check that the policy says what he wants it to.

These windows have been chosen with maximum utility and user friendliness in mind. Each value in each window can be freely chosen by the policy writer, so that no constrictions apply to the created policy. Thus a role hierarchy for organizational roles can be created in just the same way as for LOA values.



**Figure 3: The Policy Editor Configuration Options tabs, with Role Types selected.**

Figure 3 shows how a new attribute or role type is added to the Policy Editor tool. A user needs to define the name of the attribute, its LDAP Object Identifier, and its data type (which is a choice from string, integer, Boolean, real, time, date or date+time).

It will be noticed that as well as coming with built in support for the LoA attribute, the Policy Editor also has built in support for all the attributes of the eduPerson schema. This should make it more user friendly for all the Shibboleth-PERMISS policy writers, since they should not need to refer to the eduPerson schema prior to writing their PERMISS policy.

There had been uncertainties in anticipation of which of the two standards, XACML or SAML2, would replace SAML1.1. This issue is relevant to the eventual mechanism chosen for the passing of LoA from IdP to SPs. When the FAME-PERMISS proposal was being written we anticipated this would be the SAML2 protocol, after it had been released. However, we had lots of discussions within the OGF about the successor to the OGSA Authorisation protocol which is based on SAMLv1.1, and there appears to be strong support to replace SAML1.1 by XACML within the OGF. To overcome this uncertainty, we have decided to choose the approach that encoded the LoA as a user attribute and to pass it along with the user's other attributes via the Shibboleth protocols. That is, we store the LoA attribute together with the user's other attributes in the Shibboleth default store (LDAP). Shibboleth can be configured to pick up the LoA attribute via its usual configuration files (resolver.xml and arp.site.xml) in the same way as it is configured to pick up the other attributes. With this approach, all the user's attributes are stored at the same place and no further plug-ins are required to pick up the LoA attribute. So this approach is the cleanest solution, making FAME-PERMISS impervious to changes in the OGSA Authorisation protocol.

#### **Workpackage 5: GridSite extensions**

GridSite<sup>3</sup> has been extended during the course of the project in two main ways, first by the addition of XACML<sup>4</sup> policy language support, and then by providing an interface to Shibboleth<sup>5</sup> identity providers and the FAME Level of Assurance information. Before describing the design of these additions, we outline the architecture of GridSite at the start of the project.

#### Original GridSite Architecture

<sup>3</sup> The GridSite Project website is at <http://www.gridsite.org/>

<sup>4</sup> XACML is defined by the XACML committee of OASIS: <http://www.oasis-open.org/committees/xacml/>

<sup>5</sup> Shibboleth is part of Internet2 middleware initiative: <http://shibboleth.internet2.edu/>

GridSite was initially developed as a content management system for the GridPP website<sup>6</sup> and it includes a standard set of tools for editing web pages and uploading files. Due to GridPP's involvement in international High Energy Physics grid projects such as EDG, LCG and EGEE<sup>7</sup>, the user community were all in possession of X.509 digital user certificates which are the basis of the security model shared by these projects. GridSite took advantage of this fact to implement an access control model based on X.509<sup>8</sup> certificates and their Distinguished Names (DNs).

This was initially implemented as a CGI filter run by the web server during each request for a URL, but this is an inefficient way of working, as it requires a new process to be forked for each request with overheads of loading and initialising the CGI executable and parsing a configuration file. Furthermore the policy engine used for access control was tightly integrated and not reusable by other pieces of grid middleware, even though they shared the same environment of users' DN and groups defined by lists of DN.

### GridSite 1.0 Modular Architecture

For this reason the GridSite 1.0 architecture was developed and this formed the basis of GridSite at the start of the FAME-PERMISS project.

The core element of this architecture is a library, libgridsite, which provides parsing of security objects used in the EDG, LCG and EGEE grids. These include the GSI Proxy X.509 certificate chains which ultimately became standardised in RFC 3820 by the IETF PKIX Working Group, the VOMS attribute certificate profile based on RFC 3281, the Grid Access Control Language (GACL) developed for GridSite, and requests to be signed during the GridSite Proxy Delegation protocol used by EGEE/LCG. This library is used by GridSite command-line clients, in GridSite's extensions to the Apache<sup>9</sup> web server, and in the EDG/LCG Logging and Bookkeeping server used to monitor jobs status.

One of the most powerful features of the Apache web server is its support for dynamically loaded modules which can be written by third-parties but have full access to the request processing chain within the Apache and Apache's internal data structures. Modules provide handling of the HTTP and HTTPS protocols, parsing of requests, generation of static content from files and dynamic content from executables, and make access control decisions. The GridSite 1.0 architecture took advantage of this flexibility by inserting additional credential parsing (such as GSI proxies) and access control options into the handling of requests, and this allowed GridSite to make authorization decisions about resources provided by third-party modules and scripts written in any language (this is similar to the approach now adopted by Shibboleth with mod\_shib).

The content management tools provided by GridSite 0.x were converted into a standalone CGI executable, gridsite-admin, which ran along side any other CGI scripts and on top of the access control structure enforced by mod\_gridsite.

The combination of flexibility and support for grid security credentials allowed the GridSite system to be reused as the basis of Web Service middleware, in particular in the gLite<sup>10</sup> Workload Management System which admits jobs into sites on the LCG and EGEE grids. The WLMS also makes use of GridSite as an HTTPS-based fileserver for moving files around within a site. In both of these cases, access policies are written in the GACL policy language in terms of X.509 and VOMS credentials.

### Credential Acquisition and Credential Types

The components of the GridSite system which was extended as part of the FAME-PERMISS project are the acquisition of credentials and their comparison against policies by the internal GridSite policy engine. Before covering these extensions in the following sections, we first describe this process in the 1.0 architecture.

---

<sup>6</sup> The Grid for UK Particle Physics collaboration website is at <http://www.gridpp.ac.uk/>

<sup>7</sup> The EU DataGrid project (now completed, and at <http://cern.ch/eu-datagrid>) was succeeded by the LHC Computing Grid (<http://www.cern.ch/lcg>) and the EU-funded Enabling Grids for E-Science (<http://public.eu-egee.org/>)

<sup>8</sup> Standards relating to X.509 are now defined by the PKIX Working Group of IETF:  
<http://www.ietf.org/html.charters/pkix-charter.html>

<sup>9</sup> The Apache Project <http://httpd.apache.org/>

<sup>10</sup> glite is the middleware of EGEE and is available from <http://www.glite.org/>

The GridSite module, `mod_gridsite`, acquires credentials from several different sources within an HTTP(S) request and its wider context. A `GRSTgaclCred` C structure is created to hold each credential, and access functions are provided so credentials can be treated as de-facto objects. The `GRSTgaclCred` structure includes the type of credential and a series of name/value pairs.

The following credentials are supported:

- The DNS hostname of the client making the HTTP(S) request, obtained by Apache's double reverse lookup procedure (find the DNS name corresponding to the IP number, then the IP number of that name, and then the DNS name of that IP number).
- The X.509 DN of any client certificate used to make the SSL/TLS connection as part of an HTTPS request. The certificate chain is examined by `mod_gridsite` for the presence of GSI proxy certificates and `mod_gridsite` prevents Apache's `mod_ssl` from marking these certificates signed by users as invalid. A GSI Proxy Level is included in the credential to indicate whether a full X.509 user certificate issued by a Certification Authority or a delegated GSI Proxy certificate was used.
- When parsing GSI Proxy certificates, `mod_gridsite` also searches for VOMS attribute certificates stored in certificate extensions. The Fully Qualified Attribute Names of these certificates are each stored as a credential.
- GridSite's GridHTTP one-time passcodes are HTTP cookies which can refer to a previously saved credential. This allows clients to make HTTPS requests for a URL and to authenticate themselves by X.509 certificate, but gives servers the option to return a one-time passcode and redirect the client to an HTTP copy of the URL as a more efficient way of transmitting large files. If GridSite detects a one-time passcode cookie, then the credential is recreated from the session cache if the passcode has not been used already and the session has not expired.

Each of the credentials which are acquired by `mod_gridsite` are stored in `GRSTgaclCred` objects, which are in turn stored as a list inside a `GRSTgaclUser` object and passed to the GACL policy engine.

#### GACL Policies and the GridSite Policy Engine

To make a policy decision, the policy engine has to determine which policy to use, and in the 1.0 architecture these were stored as `.gacl` files in the directory structure exported by Apache as URLs. To find the policy governing a script or a static file, the directory containing the script or file was inspected for a `.gacl` file, and if not found, parent directories were examined in turn until a `.gacl` was found or the top of the filesystem was reached. If no policy was valid, then the request was immediately denied.

If a policy is found, then it is parsed and converted into a series of C structures, starting with `GRSTgaclCred` objects to represent requirements on credentials held by the client. Generic credential types are introduced at this point:

- `any-user` which matches any user, with or without credentials. This allows for the creation of world-readable directories.
- `auth-user` which matches any identified user, and is suitable for situations in which user's identities are required for auditing or for holding users accountable for their use of a service, but without pre-authorizing them as individuals.
- `dn-list` which refers to the lists of X.509 DNs used to define groups of users in the EDG and LCG grids.

Credentials are contained by `GRSTgaclEntry` objects, which can contain lists of required credentials along with the permissions (such as "read") granted to users who satisfy all of the credentials listed, and as such an entry provides an AND condition. `GRSTgaclEntry` objects may also contain denials of particular permissions, and users matching denial conditions are denied that permission irrespective of what else is present elsewhere in the policy. This is intended as a way for local administrators to override permissions for problems users or groups.

A policy is represented by a `GRSTgaclAcl` object, which can contain multiple `GRSTgaclEntry` objects. Denies aside, the permissions granted by any of the entries are combined by a logical OR, and so it is

possible for a client to acquire basic permissions from a general entry and additional administrative privileges from a specific entry aimed at the user.

The task of the policy engine is to take the GRSTgaclUser object associated with a request and compare it against the GRSTgaclAcl object associated with the resource being requested. The result of the comparison is a list of permissions determining what the client can do, and these are evaluated by the components of GridSite responsible for carrying out each operation (for example, checking for the presence of read permission before disclosing the contents of a file).

#### FAME-PERMIS extensions to GridSite

##### **Support for XACML Policies**

The first extension was to add support for the standards-based XACML policy language alongside the GACL language developed for GridSite. Since GridSite contained both a policy parser and tools for creating policy files through a web interface, this involved both generating and parsing XACML. As GridSite and Apache are C-based, we wrote new functions to do this rather than use, for example, Sun's Java-based XACML library<sup>11</sup>.

We developed an XACML profile which allowed GridSite's policy concepts to be expressed in XACML, and either written out to an XACML policy from an existing GRSTgaclAcl structure, or loaded from XACML into the GRSTgaclAcl, GRSTgaclCred structures. A single policy parser was provided which detected the presence of GACL or XACML and proceeded accordingly.

This design allowed all of the existing GridSite machinery to function as before, and allowed users to install XACML support by simply upgrading their version of GridSite. If they choose to start generating XACML, using a single GridSite directive in the Apache httpd.conf configuration file, then GridSite will continue to act correctly even if this results in a mixture of GACL and XACML policies.

Avoiding the need for an external XACML parser has also considerably simplified the installation of GridSite, which has no dependencies other than packages required by Apache or supplied as part of commercial Linux distributions, and this is reported by users as one of the system's most attractive features.

##### **Support for Access Control to Virtual URL Spaces**

During FAME-PERMIS, we have also introduced a second method for finding the GACL or XACML policy file to complement direct searches of the directories exported by Apache. In addition to this view of the filesystem, Apache allows directives to be limited in scope to the URIs visible to clients rather than to directory hierarchies present in the underlying filesystem. We have made use of this mechanism by providing a directive which explicitly states the location of the .gacl or .xacml file governing the current scope, and this can be applied to a set of URIs using Apache <Location> ... </Location> container. By doing this, GridSite access control can be applied to virtual URLs generated by some content management systems such as MediaWiki<sup>12</sup> and Subversion<sup>13</sup>, which do not export identifiable areas of the underlying filesystem. This extension to GridSite is now used in production on [www.gridpp.ac.uk](http://www.gridpp.ac.uk) for the GridSiteWiki and System Administrators' Subversion repository.

This general class of system is also particularly common amongst the interactive services aimed at humans that Shibboleth aims towards, rather than the grid filesystems and web services which GridSite has been used for by third-parties previously.

##### **Support for Credentials from Shibboleth and FAME**

The final set of changes to the design of GridSite relate to FAME and Shibboleth support itself.

As well as the identity of the user, the FAME system's main output is the level of assurance associated with the user's authentication. We have incorporated support for this directly within GridSite's handling of existing credential types or where information is available from a FAME service

---

<sup>11</sup> Sun Microsystem's reference implementation of XACML is available at <http://sunxacml.sourceforge.net/>

<sup>12</sup> MediaWiki, the software used to manage Wikipedia and related sites, is available from <http://www.mediawiki.org/wiki/MediaWiki> GridSiteWiki is a modified version of MediaWiki with support for X.509 user certificates: <http://www.gridsite.org/gridsitewiki/>

<sup>13</sup> Subversion is a software version management system which makes use of HTTP/HTTPS and the Apache webserver: <http://subversion.tigris.org/>

via use of Shibboleth, and level of assurance is evaluated along with requirements placed on credentials in the access policies.

All of the existing credential types described previously have been assigned a level of assurance:

- DNS hostnames are level zero as they constitute anonymous access.
- X.509 user certificates from a certificate authority are level 3 (unless based on hardware tokens in which case level 4.)
- GSI proxy certificates are classified at level 2, as they are equivalent to revealing a password to a service provider to gain entry (i.e. they are vulnerable to theft by service providers, including bogus providers).
- Credentials obtained from one-time passcodes are also level 2, as they may not be replayed, but may be stolen before use if sites are compromised.
- VOMS attributes are level zero as possession of an attribute confers no greater authentication than the associated GSI proxy credential.

By fully including level of assurance support in the existing credential types, we can make use of it in access policies by the addition of a generic credential that may be present in the GRSTgaclEntry objects: <level>. If this credential type is present, then one of the user's credentials must provide at least the numeric value of the Level of Assurance which is stated.

This design has immediate benefits within the existing GridSite system: for example, that a policy may restrict access to a document to a set of users and they must present their X.509 user certificate and not a delegated GSI Proxy (of the type their jobs at remote sites possess) to access it; but that another policy in another area of the same webserver may give access to log files which the same set of users may be able to access from jobs and analyse using their delegated GSI Proxies. Previously, the Apache directive GridSiteGSIProxyLimit was the only way of controlling access by user certificate versus proxy, and as a server configuration file directive, cannot readily be changed by authorised users in the way that per-directory policy files can.

To obtain Level of Assurance information from FAME, support for Shibboleth is also required. This was achieved using Shibboleth's virtual HTTP headers mechanism, whereby the Shibboleth module, mod\_shib, obtains credentials from the relevant identity provider via a local Shibboleth daemon and then makes them available to other Apache modules as HTTP headers added to the request.

This approach resulted in a vulnerability that can occur on misconfigured servers with a mixture of Shibboleth and other access control mechanisms: it is possible for clients to include the same headers (as real HTTP headers) that Shibboleth would have generated artificially after the request was received. A knowledgeable user may then be able to falsely claim possession of the required credentials and gain access.

To circumvent this, we added an additional mod\_gridsite function which is called by Apache immediately after the receipt of the request but before mod\_shib modifies the list of headers. This mod\_gridsite function removes any instances of the headers which it will subsequently look for, preventing users from injecting them directly.

If authentication via Shibboleth has been carried out, then the headers User-Distinguished-Name and Nist-LoA may be present if the identity provider releases the user's DN and is aware of the NIST level of assurance information provided by FAME. (The exact names of the headers can be set by the Shibboleth AAP.xml file.)

If mod\_gridsite detects the presence of these headers, it uses them in preference to any X.509 certificate information provided by SSL/TLS, as this gives users the opportunity to choose which identity to use when authenticating to the identity provider. GRSTgaclCred structures are created with associated Level of Assurance information, and the resulting GRSTgaclUser can be evaluated against GACL/XACML policies as with any other credential type.

### Summary

This design, which is now fully implemented in GridSite 1.4, therefore extends the GridSite 1.0 architecture with XACML support, and makes the Level of Assurance an integral part of GridSite whether or not Shibboleth and FAME are being used directly. However, if a site and its users do opt to

support them, then they can be used alongside direct X.509 authentication and with full control over what levels of credentials are acceptable within the policies that authorised users can manage.

**Workpackages 6: FAME, PERMIS, GridSite, and Shibboleth integration and case studies**

Integration Environment

**FAME**

For a user to access to different authentication services, the user may have multiple identities each registered with a different authentication service. For example, a user may have a username identity registered with a username/password based authentication service, and a X.509 DN (Distinguished Name) identity registered with a X.509 certificate-based authentication service. FAME uses a database table to deal with this single-user-with-multiple-identity scenario. In addition to the users' multiple identities registered with different authentication servers, the FAME database also stores other configuration information required by the module, such as the URLs and LoA values of the Authentication Servers connected to FAME, secret keys shared between the FAME module and these Authentication Servers, and the secret keys used internally by FAME for generating FAME cookies. The multiple identities stored in the FAME database are connected to the users' unique LDAP identities stored in a Shibboleth's LDAP directory. This extension is necessary as a Shibboleth IdP service only keeps a user's unique identity in the LDAP directory, together with the user's other attributes, and is not aware that a user may have multiple identities registered with multiple authentication services. In this way, a user can have multiple authentication credentials, and can be authenticated using any one of these credentials depending on the sensitivity level of the resource to be accessed. Upon successful authentication, a user's alternative identities will be mapped by FAME onto the user's unique Shibboleth LDAP identity that is then passed to the Shibboleth HS (Handle Service). The HS will then use the user's LDAP identity to fetch user's additional attributes from the Shibboleth LDAP directory before passing them onto the requesting SP. The whole process is transparent to the users.

The following table further explains how FAME links a user's additional identity to his/her unique LDAP identity. The table has three fields: a *ldap\_id* field storing the user's LDAP identity, a *ldap\_attribute* field indicating the name of the LDAP attribute that is used to keep the user's LDAP identity, e.g. the user's username, and this can be any of the LDAP attributes such as uid, cn, sn, etc., and an *alternative\_id* field containing the user's alternative identities registered with the authentication services supported by the IdP. Each user may have several entries in this table, and each entry corresponds to the user's identity registered with one of the authentication services.

Table 3: FAME identity mapping table

<u>ldap_id</u>	<u>ldap_attribute</u>	<u>alternative_id</u>
<i>ldapuserA</i>	<i>uid</i>	<i>testuser</i>
<i>ldapuserA</i>	<i>uid</i>	<i>kerbuserA@CS.MAN.AC.UK</i>
<i>ldapuserA</i>	<i>uid</i>	<i>/C=GB/ST=Greater Manchester/O=University of Manchester/OU=Dept. of Computer Science/CN=FAME Project demo certificate v2/emailAddress=Alex@cs.man.ac.uk</i>

In this example, the single user identified by the LDAP identity, *ldapuserA*, stored as the LDAP directory attribute *uid*, has three entries containing three alternative identities the user has registered with three different authentication services. For example, the user's identity (i.e. username) registered with the username/password based authentication service is *testuser*, his identity with the Kerberos authentication service is *kerbuserA@CS.MAN.AC.UK*, and his identity with the X.509 certificate-based authentication service (expressed as his X.509 DN) is */C=GB/ST=Greater Manchester/O=University of Manchester/OU=Dept. of Computer Science/CN=FAME Project demo certificate v2/emailAddress=Alex@cs.man.ac.uk*. FAME uses the LDAP attribute *uid* to extract the user's LDAP identity that is to be used by Shibboleth whenever the user authenticates himself with any of the three authentication services.

In addition, we have extended the LDAP definition of the LoA attribute to also include specification information about a particular LoA definition used by FAME. In other words, the LoA attribute name now indicates the standard body and/or specification identifier of a particular LoA definition used. For example, the LoA definition currently used by FAME is defined by NIST. So the LDAP LoA attribute name is NIST-LoA. The LoA attribute value is an integer in the range of [0, 1, 2, 3, 4] as derived by FAME based upon the authentication credential and service used in a particular authentication instance. This LoA attribute modification greatly increases the flexibility and extensibility of the FAME solution. Any future development and/or change in authentication technologies and/or in the definition of authentication strength will not affect the deployment of FAME.

### **PERMIS and GridSite**

Both PERMIS and GridSite authorisation decision engines have now been extended to take the LoA attribute into consideration when making access control decisions. In other words, the authorisation decisions are now made based on the tuple: {Subject, LoA, Target, Action}.

PERMIS uses either X.509 attribute certificates pulled from an LDAP directory, or attributes pushed by the Shibboleth IdP to make an authorisation decision. Access control is provided by an Apache plug-in module called mod\_permis, which calls the PERMIS decision engine.

GridSite uses either X.509 credential certificates or attribute certificate credentials returned from an IdP as Shibboleth attributes for authentication and access control. Access control is now provided within the main GridSite Apache module. The module makes access policy decisions using the same GACL or XACML policy engine.

One clear requirement, however, is for policies to be able to restrict access depending on how the credential was established, since the same X.509 attribute certificate or Shibboleth attribute can now be obtained either via a digital certificate stored in the user's Web browser, or merely via a username/password combination and a Shibboleth IdP. To address this, both PERMIS and GridSite have used the FAME LoA approach.

GridSite has defined an additional credential type, nist-loa, within GridSite's GACL policy language and internal C credential structures to represent this. This has a numeric value corresponding to one of the four NIST LoA levels, and may be obtained directly as a Shibboleth attribute if using a FAME-enabled IdP, or set by GridSite based on the type of non-Shibboleth credential presented. Currently, level 2 is assigned to GSI Proxies and level 3 if a full X.509 certificate is presented. GridSite intends to add an option to configure the default for full X.509 certificates on a per-Certification Authority basis, to accommodate CAs which only issue certificates using hardware tokens and therefore qualify for level 4.

These additions to the GridSite Apache Web server module have implicitly added Shibboleth and NIST LoA support to the existing website and portal systems built on GridSite's security framework.

PERMIS does not need to define its own credential or attribute types. The PERMIS policy can be configured with any existing LDAP attribute definitions. Consequently, we have used the LDAP definition for the NIST LoA attribute to represent the LoA attribute. Because PERMIS supports hierarchical attributes, we have used each LoA value to represent one attribute in the attribute hierarchy. We define LoA=4 to be superior to LoA=3 to be superior to LoA=2 to be superior to LoA=1. Superior attributes inherit all the privileges of their subordinate attributes, recursively. In this way, a user who authenticates with LoA=4 gains access to all the resources that are available to LoAs of 3, 2 and 1.

We are able to make use of the modular access control framework provided by Apache to install the Shibboleth, GridSite and PERMIS Apache modules within the same web server, with a cascade of authentication and authorization steps. This allows credentials to be delivered directly using X.509 identity and attribute certificates, or obtained by contacting a Shibboleth IdP, or pulled as X.509 attribute certificates from an LDAP directory by the PERMIS PDP (Policy Decision Point), or DN-Lists to be fetched asynchronously by GridSite. However, not all combinations can be effectively used together, due to the different order in which the various plug-in modules are called by Apache. Shibboleth authentication and attribute pulling is always the first step, as this takes place during Apache's authentication phase. PERMIS authorisation is always the second step, as this takes place during Apache's authorisation phase. GridSite authorisation is always the last step, as this takes place during Apache's fix-ups phase. Consequently it not possible to use credentials pulled by either the PERMIS PDP or the GridSite PDP in the other's authorisation decision making. Nor is it currently

possible to use SSL client certificate authentication with PERMIS authorisation, since the distinguished name of the user is not made available by Apache until the fix-ups phase. Table 2 above shows the allowable combinations.

### Case Studies

With regard to the component integration and trial deployments, we have set up the following three test scenarios.

#### **Test Scenario 1 (the FAME-Shibboleth-PERMIS Integration)**

The Shibboleth SP run at Kent uses the PERMIS authorisation decision engine to control the access of its resource, i.e. five web pages of the same image but with 5 incremental levels of resolution/zoom. *Page 0* has the lowest resolution level, and therefore requires the least level of authentication strength, i.e. LoA=0 or no authentication, for its access, while *Page 4* has the highest resolution level, and therefore can only be accessed by users with the highest value of LoA, i.e. LoA=4. Intuitively, pages, 1, 2 and 3, are controlled by the minimum LoA values of 1, 2, and 3, respectively. *Page 0* with a LoA requirement of 0 is in fact a public page accessible to anybody, i.e. it does not require user identification and authentication. The idea here is that, in order to get a clearer view of the image, one has to be identified and authenticated using a credential with a higher value of authentication assurance.

Using the WAYF (Where Are You From) service provided by Internet2, a user trying to access the PERMIS-protected web pages is redirected to the FAME-enabled IdP for authentication (except for the *Page 0* which is publicly available). The user is then presented with all the authentication services supported by the IdP, namely, the Kerberos authentication service, LDAP authentication system, SSL certificate-based authentication service with PKI credentials stored in Web browsers, and SSL certificate-based authentication service with PKI credentials stored in smart card tokens, the basic Apache username/password-based authentication system, and GridSite's username/password-based authentication service. Upon the user's selection and successful authentication, the user will be assigned with a LoA value that is derived by the FAME module based upon the type of the credential used and the authentication service chosen in this authentication instance. For example, if the user is authenticated with the Kerberos service, then he/she will be given a LoA level of 2. If the user uses a smart-card token, then he/she will get a LoA level of 4. Once the LoA value is derived, it is passed on to the PERMIS decision engine run at Kent along with the user's other Shibboleth attributes using the Shibboleth protocol. The user is only granted with the access if and only if his LoA value is equal to, or greater than, the LoA value required by the resource. The demo is publicly available here: <http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISDemo/>.

#### **Test scenario 2 (the FAME-Shibboleth-GridSite integration)**

Test scenario 2 is set up to demonstrate the interoperability between FAME and GridSite components connected via Shibboleth. In this test scenario, GridSite acts as an SP hosting a web page that can be accessed by GridSite users either with X.509 credentials or username/password pairs. If a user authenticates with his X.509 credential, then the GridSite SP handles the authentication task without extra support. The GridSite is also provided with a username/password based authentication service to support users who do not have X.509 credentials, and this GridSite authentication service, along with the GridSite SP, is linked to the FAME module through Shibboleth. When using a username/password pair to access the GridSite SP, the user will be redirected by the WAYF service to the FAME-enabled IdP (run at UoM-CS) for authentication. There, from a list of available authentication services, the user can choose the GridSite username/password authentication service (run at UoM-Physics). Upon successful authentication with his username and password, the user will be redirected back to the FAME service that then assigns the user with a LoA value based on the authentication strength of the GridSite authentication service (set to level 2 during the demo). This LoA value is subsequently passed to the GridSite SP via the Shibboleth protocol. The user will then be presented with the test page that requires a minimum LoA level of 2. A Shibboleth session was established and the entire process was tested successfully.

#### **Test scenario 3 (Shibboleth-FAME-PERMIS-GridSite integration)**

This scenario is to demonstrate the co-existence of all three components, Shibboleth, PERMIS and GridSite, on the same host using remote Shibboleth-FAME authentication. The web resources protected are three sets of five web pages with 5 incremental levels of zoom, as described in Test Scenario 1 above. All the pages are managed by the UoM-Physics SP. The first set of five pages are

protected by the PERMIS authorisation engine. The second set of five pages are protected with the GridSite authorisation decision engine. The third set of five pages are protected with both GridSite and PERMIS authorisation engines. Each of the five pages in a set will require a different LoA for access, from 0 to 4, respectively, as described in Test Scenario 1. To gain access to the PERMIS protected pages the user must have a PERMIS role attribute; to gain access to the GridSite pages he must have a GridSite role attribute, and to gain access to the last set of five pages he must have both role attributes. This demonstration shows that a website can be protected by either GridSite authorisation, or by PERMIS authorisation, or by both authorisation PDPs working in conjunction with each other. This test scenario actually tests the third row of Table 2.

To summarise, this test scenario has demonstrated that:

- i) a user with LoA level 4 got access to all five web pages;
- ii) a user with LoA level 3 got access to *Pages 3,2,1* and 0;
- iii) a user with LoA level 2 got access to *Page 2* , 1 and 0;
- iv) a user with LoA level 1 got access *Page 1* , and 0;
- v) a unauthenticated user (i.e. with LoA level 0) got access only to *Page 0*.

The outcome of the case studies have demonstrated that whenever a user tries to access a page with a LoA lower than the minimum required by the page, the access request would be rejected. An access is granted if and only if a user has authenticated her/himself with an authentication service/token that satisfies the minimum LoA required by that Web page. The case studies have also demonstrated that the three components can inter-work and interoperate together to achieve LoA linked fine-grained access control.

Full details with regard to these case studies are described in our report on case studies, FAME-PERMISOct2006CaseStudiesReport.pdf, available at <http://www.fame-permis.org/publications/>.

#### **Additional contributions: FAME-CLEF**

In addition to being deployable under the Shibboleth infrastructure, FAME has been extended to protect standalone applications, e.g. those without a web front-end. The CLEF-services (Clinical e-Science Framework Services) project [CLEF] led by Prof Alan Rector from the University of Manchester was looking into the possibility of using FAME-PERMIS as their authentication and authorisation solutions. The project is funded by UK Medical Research Council. A set of stand-alone applications, collectively called CLEF Thick Client (CLEF-TC), have been developed by the CLEF-services team to access data in the CLEF Repository containing images, clinical and genomic data, and to allow clinicians to undertake statistical analyses and clinical research work. These applications are written mainly in JAVA, and are executed on users' PCs. Due to the way that these applications were developed, CLEF-TC can not be interfaced with a Web front-end, which means that CLEF-TC can not be Shibbolized. However, the CLEF team wishes that access to the CLEF Repository via their CLEF-TC be protected with the FAME services. Taking these requirements into consideration, we have extended the FAME solution and developed the FAME-CLEF solution, as shown below (also available at <http://www.fame-permis.org/links.html>). So FAME can now be used to protect stand-alone applications, such as CLEF-TC, in addition to those with a web front-end.

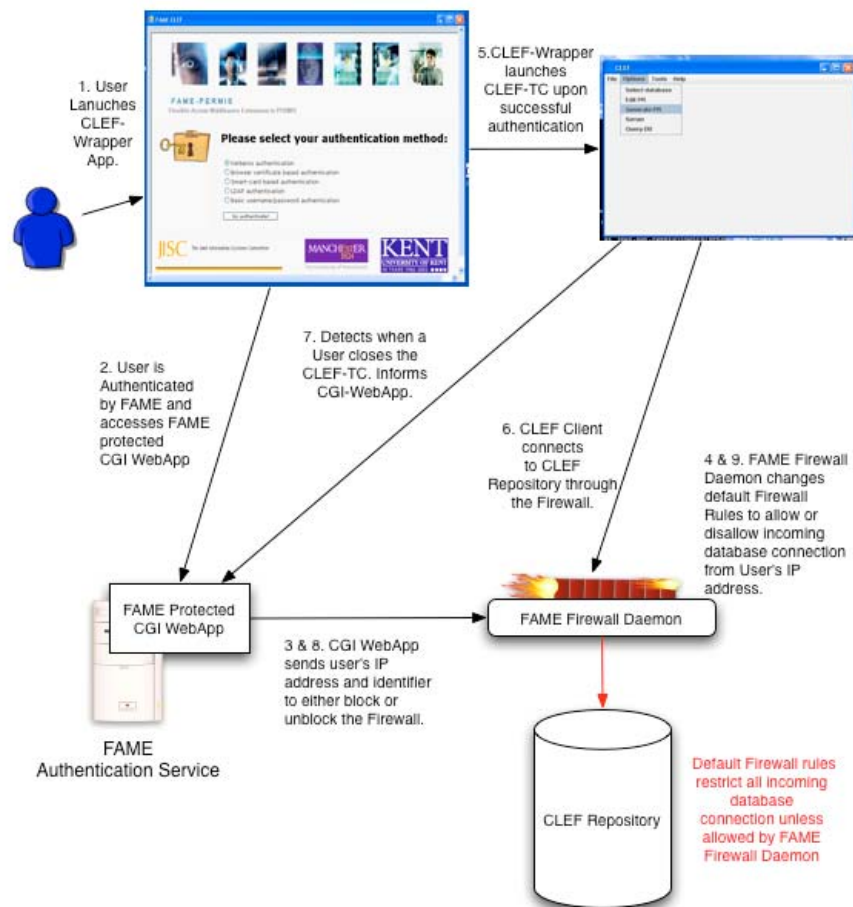
The FAME-CLEF architecture contains the following components.

**CLEF-Wrapper application:** The CLEF-Wrapper application is a .NET [7] Windows application that wraps up the CLEF-TC applications in order to enforce controlled access to these applications via FAME authentication services. It is run on a user's client machine and can be customised to suit any stand-alone applications. The user invokes it by double clicking on the CLEF-Wrapper executable. Once invoked, CLEF-Wrapper first opens a browser instance to establish a connection to the URL of the CGI-WebApp application (to be discussed shortly) that is protected by the FAME authentication service. FAME intercepts the connection request, and prompts the user for authentication. If the authentication is successful, CLEF-Wrapper is allowed to proceed to invoke CLEF-TC running on the user's host, and, at the same time, the connection to CGI-WebApp is allowed through, which proceeds to notify the Firewall to unblock the IP address of the user's host. CLEF-TC can then establish an access session via the Firewall to the CLEF Repository.

Otherwise, if the authentication is not successful, CLEF-Wrapper terminates the browser instance and CLEF-TC will not be invoked. Similarly, without successful authentication, CGI-WebApp will not be invoked (as prevented by FAME), and the Firewall will not be unblocked.

Another function performed by the CLEF-Wrapper is that it can detect the termination of an on-going access session between this host and the Repository (e.g. when the CLEF-TC application is closed). Upon the detection of the termination, CLEF-Wrapper sends a notification message to the FAME-protected CGI-WebApp that will inform the Firewall to block the concerned IP address. In addition, a timeout mechanism is also implemented in the Firewall, such that should a session inactive for a specified period, the Firewall will block the IP address automatically.

The CLEF-Wrapper application allows easy customisation through an XML configuration file that administrators can use to configure the target CLEF-TC applications and the URL address of the FAME-protected CGI-WebApp.



**Figure 4: The FAME-CLEF architecture**

**FAME-protected CGI-WebApp:** CGI-WebApp is a web application written as the Common Gateway Interface (CGI) script [13]. Its functions are, firstly, to notify the Firewall the identity and the host IP address of a user who has been successfully authenticated by FAME, and, secondly, to notify the Firewall when the user closes the session, which is notified by the CLEF-Wrapper run on the user's host. CGI-WebApp is protected by FAME and is run on the same Web server as FAME, so without successful authentication or a valid sso cookie, a user's HTTP request will not be allowed to hit the URL of CGI-WebApp.

As indicated above, in addition to notifying the Firewall a user's host IP address, CGI-WebApp also passes the user's identity extracted by FAME during authentication (e.g. a user's Kerberos Principal Name if the authentication method chosen by the user is Kerberos) for record-keeping and audit purposes.

**FAME Firewall Daemon:** The Firewall Daemon protects the CLEF Repository to ensure that only those requests made from the IP addresses notified by CGI-WebApp are allowed to get through to the Repository. By default, the Firewall blocks all incoming connections. Only when a user is successfully authenticated via FAME, will CGI-WebApp be invoked, which then retrieves the IP address of the user's host from the REMOTE\_ADDR environment variable<sup>14</sup> and obtains the user's identity from the REMOTE\_USER environment variable that is populated by FAME upon the successful authentication, and sends them to the Firewall Daemon via a secure XML-Remote Procedure Call (XML-RPC) invocation message. The Firewall, upon the receipt of this notification, will unblock the IP address, so that the request can get through to the Repository. This access permission will be denied as soon as the timeout expires, or when another notification message is received from CGI-WebApp.

The identity of a user, that is extracted by FAME during an authentication process and passed on by CGI-WebApp, is used by the Firewall Daemon to log the access record: the identity of the user making the access along with the date/time when the user's host IP address is unblocked.

## 5. Outputs and Results

The project team has made the following achievements.

### Project website

We have set up and been maintaining the project website, <http://www.fame-permis.org/>, that publishes all the deliverables including reports, published papers, software and user guides. In addition, the website has a live demo of the FAME-PERMISS software built on a real implementation of the Shibboleth infrastructure.

### Project Deliverables

Project deliverables produced:

- D1 & D3: Source code for the FAME system, fame.pm ([http://www.fame-permis.org/publications/fame\\_v1\\_1.zip](http://www.fame-permis.org/publications/fame_v1_1.zip)); this software has implemented all the functions as described for the original deliverables, D1 and D3.
- D2: Report on the design of FAME architecture and components, [http://www.fame-permis.org/publications/FAME\\_Design\\_and\\_Architecture\\_v1\\_0.pdf](http://www.fame-permis.org/publications/FAME_Design_and_Architecture_v1_0.pdf).
- D4: Report on the design of FAME architecture, components and APIs: FAME\_Architecture\_Components\_API\_v1\_1.pdf.
- D5: (Deliverable of WP3) Source code for the PERMISS Policy Editor that incorporates built-in support for the LOA attribute: available for download from [www.openpermis.org](http://www.openpermis.org) or as binary files from <http://sec.cs.kent.ac.uk/permis/>.  
(Deliverable of WP4) Source code for the PERMISS SAAM and PERMISS PDP that access and use the LOA attribute when making authorisation decisions, available for download from [www.openpermis.org](http://www.openpermis.org) or as binary files from <http://sec.cs.kent.ac.uk/permis>.
- D6: Source code support for NIST LoA in GridSite, gridsite-1.4.3.src.tar.gz, available at <http://www.gridsite.org/download/sources/gridsite-1.4.3.src.tar.gz>.
- D7: Source code for XACML support in GridSite, gridsite-1.4.3.src.tar.gz, available at <http://www.gridsite.org/download/sources/gridsite-1.4.3.src.tar.gz>.
- D8: The integrated system software release / demonstration, available here at: <http://www.fame-permis.org/publications.html>.
- D9: Report on case studies, <http://www.fame-permis.org/publications/FAME-PERMISSOct2006CaseStudiesReport.pdf>.
- D10: Final report (this document), software release (<http://www.fame-permis.org/publications/>) and user guide (attached to this document as appendices).

The differences between what has been delivered and what has been written in the original proposal:

---

<sup>14</sup> Apache Environment Variables – see [http://www.zytrax.com/tech/web/env\\_var.htm](http://www.zytrax.com/tech/web/env_var.htm).

- Due to the change in Objective 9, as mentioned in the ‘Methodology section above, D6 (original deliverable name: Source code for PERMIS/GridSite interoperability in GridSite) has been revised into: Source code support for NIST LoA in GridSite.
- In addition, we have developed a PIN activated smart card authentication service for FAME-PERMISS software evaluations and testing.
- In addition, we have developed an FAME extension, FAME-CLEF (documentation and source code are downloadable at <http://www.fame-permis.org/links.html> - see the link, FAME-CLEF), for achieving the vision of FAME, but for resources that can not be Shibbolethed or do not have a web front-end.

### Dissemination Activities

The project team has taken part in the following meetings and dissemination activities (giving presentations and/or poster and demonstrations):

- (1) JISC Joint Programmes Meeting, Brighton, 6<sup>th</sup> and 7<sup>th</sup> July, 2004.
- (2) Shibboleth Installation Workshop, London, Oct 19<sup>th</sup>, 2004.
- (3) JISC Core Middleware Programme Meeting, Loughborough, May 16-17<sup>th</sup>, 2004.
- (4) All Hands e-Science Conference 2005, Nottingham, Sept 19<sup>th</sup> -22<sup>nd</sup>, 2005.
- (5) JISC Core Middleware Programme Meeting, Lakeside, Lake Windermere, November 14-15<sup>th</sup>, 2005.
- (6) JISC Core Middleware Showcase Meeting held in London, 18<sup>th</sup> July 2006.
- (7) All Hands Meeting 2006 in Nottingham, September 2006.

### Publications

In addition, the project has generated the following journal and conference publications:

- (1) J. Chin, M. Parkins, N. Zhang, A. Nenadic, and J. M. Brooke, “GridFAME: Flexible Authentication Middleware Extension for Grids”, 2<sup>nd</sup> Workshop on Grid Computing & Applications, Biopolis, Singapore (GCA 2005), 5<sup>th</sup> May 2005, pp. 17–26, also available at: [http://www.fame-permis.org/publications/GridFame\\_2005.pdf](http://www.fame-permis.org/publications/GridFame_2005.pdf).
- (2) N. Zhang, L. Yao, J. Chin, Q. Shi, A. Nenadic, A. McNab, A. Rector and C. Goble, “Plugging a Scalable Authentication Framework into Shibboleth”, 14<sup>th</sup> IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2005), 13-15 June 2005 in Linkoping, Sweden, pp. 271-276, also available at: <http://www.fame-permis.org/publications/WET-ICE2005.pdf>.
- (3) J. S. Chin, M. Parkin, M., N. Zhang, A. Nenadic, J. M. Brooke, “Authentication Strength Linked Access Control Middleware for the Grid”, International Journal of Information Technology, 11(4): 1-12, 2005, also available at: [http://www.fame-permis.org/publications/JIT\\_2005.pdf](http://www.fame-permis.org/publications/JIT_2005.pdf).
- (4) N. Zhang, L. Yao, A. Nenadic, J. Chin, C. Goble, A. Rector, D. Chadwick, S. Otenko and Q. Shi; “Achieving Fine-grained Access Control in Virtual Organisations”, Concurrency and Computation: Practice and Experience published by John Wiley & Sons Ltd, Published On-line 12 Oct 2006, also available at: <http://www.fame-permis.org/publications/CCPE.pdf>.
- (5) Aleksandra Nenadic, Ning Zhang, Jay Chin, Carole Goble, “FAME: Adding Multi-Level Authentication to Shibboleth”, the 2<sup>nd</sup> IEEE International Conference on e-Science and Grid Computing, Dec 4-6, 2006, Amsterdam, Netherlands, pp.157-164, also available at: <http://www.fame-permis.org/publications/eScience2006.pdf>.

The project team has achieved the publication target set in the initial project plan.

### The Fame - PERMISS Demo

For the demo purposes, we have set up a Shibboleth Identity Provider (IdP) protected by the FAME multiple authentication services at the School of Computer Science, University of Manchester (UoM-CS). The FAME component here also provides the LoA derivation and single-sign-on services, and incorporates six different authentication systems that the users can choose from: Apache Basic Username and Password, Kerberos, LDAP, certificate-based SSL client authentication with either

browser certificates or certificates stored in smart-cards, and GridSite username-password authentication (hosted at Manchester School of Physics (UoM-Phy) and intended for GridSite users). These various authentication services provide different Levels of Assurance - ranging from 1 to 4 as follows:

- Apache Basic Username and Password - Level 1
- Kerberos, LDAP, GridSite - Level 2
- SSL and browser certificates - Level 3
- SSL and smart-card certificates (provided it can be proved the certificate came from a smart card) - Level 4.

We have set up test user accounts for the above authentication systems, which all map to the same entity (i.e. the same entity receives different LoA values when being authenticated with his/her different authentication credentials):

- For Apache Basic Username and Password
  - username: testuser
  - password: testpasswd
- For Kerberos
  - username: kerbuser
  - password: kerbpasswd
- For LDAP
  - username: ldapuser
  - password: ldappasswd
- For certificate-based authentication
  - We accept certificates issued by the HEPKI CA, University of Wisconsin (used by the Shibboleth community)

We are also running a Shibbolized Service Provider (SP) that is installed with the PERMIS authorisation engine and is run at the Computing Lab the University of Kent (UoK). The SP provides a set of five Web pages, named Page 0 - Page 4, containing the satellite images of the same location but with various level of zoom - the higher LoA value you have, the more detailed image will be presented to you.

Page 0 is publicly available to everyone, while Pages 1 - 4 require minimum LoA values of 1 to 4, respectively. So, if you try to access a page that requires, say LoA 2, you will be redirected to the WAYF page from which you select FAME as your origin institution. You are then redirected to the FAME-protected IdP at UoM-CS, where you should select an authentication method that will give you LoA Level 2 or higher (e.g. LDAP or Kerberos or certificate-based authentication, using test credentials as provided above). After successful authentication, you will be redirected back to the page you've tried to access, and if you have a sufficient LoA value, access to the page will be granted.

We have also implemented our own Where-Are-You-From (WAYF) service at UoM-CS that enables users to select their origin institution, where they will be redirected to for user identification and authentication.

You are welcome to try the demo available at: <http://www.fame-permis.org>.

### **Collaborating with NGS**

We have also made contact with Manchester Computing Centre that has been operating the CSAR (<http://www.csar.cfs.ac.uk>) National High Performance Computing service and the JISC-funded data node, both of which are part of the National Grid Service (NGS), to look into the possibility of integrating FAME-PERMISS into the NGS. This collaboration has led to the joined bid for the SHEBANGS project that is funded by JISC. SHEBANGS is aimed at providing credential translation services whereby Shibboleth-authenticated users can acquire (or delegate) temporary credentials to access the National Grid Service (NGS), for which FAME is used as an authentication solution. Through this collaboration, we have identified gaps in applying the NIST LoA definition to Grid applications; some of the credential types, such as those stored in on-line repository and proxy credentials, have not been addressed in the NIST LoA definition. At the time of this writing, the project

team is actively involved with the OGF community to address these gaps in collaboration with the International Grid community.

## 6. Outcomes

The outcomes of the FAME-PERMIS project can be summarised as follows:

- By plugging FAME into an IdP, a Shibboleth service provider using PERMIS as its authorisation engine will be able to achieve risk based authorisation based on the user's LoA value, and fine-grained access control to resources with varying levels of sensitivity.
- FAME can also be used to achieve risk-based authentication.
- Adding FAME/LoA support to an attribute based authorisation infrastructure that already supports hierarchical RBAC, such as PERMIS, was absolutely trivial once the LoA was converted into an LDAP attribute. In comparison, adding LoA support to GridSite GACLs was a major undertaking that caused the project to be delayed by 3 months.
- The XACML parsers and generators are installed at all LCG/EGEE sites, and the GACL/XACML based policy engine is used every time a job is accepted by a site for execution.
- The GridPP website (and other GridSite-based installations, eg Grid Ireland, Grid Support centre, as they upgrade to 1.4) can now be used as a Shibboleth service provider rather than purely using X.509 authentication.
- FAME/LoA support within GridSite now provides a clear way to distinguish between X.509 certificates (which users have in their web browsers) and GSI proxies (which their jobs hold at remote sites) in access policies.
- The project team is playing a leading role in defining standards for the use of authentication levels of assurance (LoA) to achieve fine-grained access control in VO (Virtual Organisation) and Grid environments. The Project manager, Ning Zhang, is now the Chair of a new Research Group, LoA-RG, being formed in OGF (Open Grid Forum).
- The FAME-PERMIS middleware solution for linking LoA to authorisation decision making is the first such effort in the world, and FAME is the first software to have implemented the NIST LoA recommendation.

Apache's quality and modular design is an excellent platform for building middleware on (mod\_gridsite, mod\_shib and mod\_permis, Fame.pm are all symptoms of this realisation.)

We should use open source software from the mainstream web world as much as possible (e.g. Apache again) and resist the temptation to write new servers from the ground up (e.g. GridFTP or Globus Gatekeeper).

Sites/users dislike products with long lists of unusual dependencies.

A note of caution: users and system administrators in particular, are wary of Shibboleth's redirecting of users between sites - it may make naive users vulnerable to "phishing" attacks similar to those used to obtain bank account details with faked bank websites. Furthermore, users do not see a benefit in the overhead Shibboleth imposes when they see essentially one identity provider and one service provider as relevant to their work.

One of lessons learnt (also predicted early on) was that taking part in standardisation activities and waiting for its outcome can be a time-consuming process.

Shibboleth is now being actively investigated by EGEE/LCG as a way of providing access to portals and other human-facing services, and the JISC-funded extensions to GridSite have left us in a good position to provide support for this move.

The complex nature of grid authentication and authorisation will be hidden from users accessing grids and replaced by the familiar access mechanisms provided by their institutions through Shibboleth.

The use of LoA has been adopted by UK and US governments in their e-Government initiatives, and JISC is now investigating the use of LoA in VO and federation contexts. The JISC-funded FAME-PERMIS project has left us in a better position to conduct the related projects, such as the recently funded ES-LoA project (E-infrastructure Security: the Levels of authentication Assurance).

The application of FAME to the Grid context via the SHEBANGS project has led us to identify a number of gaps in the NIST LoA definition. For example, there is currently no clear LoA definition with regard to the use of delegated credentials and credentials stored in an on-line repository. Collaborations with international communities are required in order to bridge these gaps, and this project has put us in a good position to lead this collaboration initiative.

## 7. Conclusions

We have achieved the aim and objectives set in the initial project plan, and, in addition, we have tackled well the issues that were not anticipated initially, e.g. the SSO issue and the additional use case scenario. We have also achieved impressive dissemination and publication records.

## 8. Implications

More work on the LoA definitions is required to bridge the gaps as introduced by Grid use case scenarios.

FAME should to be extended to support Portal and n-tier authentication use case scenarios.

The current version of SAML does not have any LoA class, and a standard approach to LoA representation in SAML is recommended as this would allow us to implement a standard negotiation mechanism to be used between an IdP and an SP.

On the attribute front, we identified that attributes from multiple authorities will soon be a common requirement, and this led to the recent funding of the Shintau project.

## Appendices

- A1: FAME Design Final Report
- A2: FAME User Guide
- A3: GridSite User Guide

## References

- [Apache] Apache project, available at <http://www.apache.org/>.
- [CLEF] CLEF project, available from <http://www.clinical-escience.org/>.
- [FAME] FAME-PERMISS deliverables and publications, available from <http://www.fame-permis.org/publications.html>.
- [FAME-CLEF] FAME-CLEF solution, available at <http://rpc170.cs.man.ac.uk/fame-clef/>.
- [GridSite] GridSite, available from <http://www.gridsite.org/download/sources/>.
- [Hump00] M. Humphrey, et al, "Accountability and Control of Process Creation in Metasystems", Network and Distributed System Security Symposium, 2000.
- [Kerberos] Kerberos: The Network Authentication System, available at <http://web.mit.edu/Kerberos/>.
- [Internet2] <http://www.internet2.edu/>.
- [LDAP] LDAP, Lightweight Directory Access Protocol, available at <http://www.openldap.org/>.
- [NIST04] W. E. Burr, et al, DRAFT Recommendation for Electronic Authentication, NIST Special Publication 800-63, Jan. 2004.
- [NIST06] William E. Burr, et al, Electronic Authentication Guideline, NIST Special Publication 800-63 version 1.0.2, April 2006.
- [PERMISS] PERMISS project, available at <http://sec.cs.kent.ac.uk/permis>.
- [PKI] X.509 Public Key Infrastructure: Certificate and CLR Profile, available at <http://www.ietf.org/rfc/rfc2459.txt>.
- [Shib04] Shibboleth project, available from <http://shibboleth.internet2.edu/>.

- [Welc04] V. Welch, et al "Security for Grid Services", available at <http://www.globus.org/Security/GSI3/GT3-Security-HPDC.pdf>, 10 Feb 2004.
- [WAYF] Internet2 WAYF service, available at <https://wayf.internet2.edu/InQueue/WAYF>.
- [SSL] SSL: Secure Socket Layer Protocol, available at <http://wp.netscape.com/eng/ssl3/>.