FAME-PERMIS Project
Report on Case Studies – Workpackage 6 Deliverable

October 2006

Ning Zhang\textsuperscript{1}
Alexandra Nenadic\textsuperscript{1}
Jay Chin\textsuperscript{1}
David Chadwick\textsuperscript{2}
Andrew McNab\textsuperscript{3}

\textsuperscript{1}School of Computer Science
University of Manchester
Oxford Road, Manchester M13 9PL

\textsuperscript{2}Computing Laboratory
University of Kent
Canterbury, Kent, CT2 7NF

\textsuperscript{3}School of Physics and Astronomy
University of Manchester
Oxford Road, Manchester M13 9PL
## Contents

1. Introduction
   1.1. FAME-PERMIS Project ................................................................. 3  
   1.2  Workpackage 6 Description ........................................................... 3  

2. Integration of FAME with PERMIS and GridSite
   2.1. Integration Environment .............................................................. 3  
   2.2. FAME ......................................................................................... 4  
   2.3. PERMIS and GridSite ................................................................. 5  
   2.4. The Shibboleth FAME-PERMIS-GridSite Integration ............... 6  
   2.5. Resource Access Procedure ...................................................... 7  

3. Using FAME to Protect Stand-Alone Applications
   3.1. CLEF-Services ........................................................................... 13  
   3.2. Integration of FAME and CLEF ................................................. 13  
   3.2. FAME-CLEF Architectural Components .................................. 14  
   3.3. CLEF Repository Access Procedure ......................................... 16  

References ............................................................................................... 17
1. Introduction

1.1. FAME-PERMIS project

The FAME-PERMIS (Flexible Access Middleware Extensions to PERMIS) project [4] is aimed at facilitating uniform and multi-factor authentication and authentication strength linked fine-grained access control. To achieve this aim, the project develops or extends three middleware components, FAME, PERMIS [9] and GridSite [5], in the SHIBBOLETH infrastructure [2] – an open source solution to support inter-institutional sharing of Web resources subject to access control [2]. FAME integrates a variety of authentication mechanisms, including IP addresses, username/password pairs, certificate-based soft as well as hard tokens, and derives an authentication level of assurance (LoA) based upon the authentication method/device used in an authentication instance. The LoA definition adopted by FAME is specified in the Electronic Authentication Guideline, recommended by the National Institute of Standards and Technology (NIST) [1]. The recommendation classifies authentication methods/tokens into four levels of authentication assurance, levels 1 to 4, in terms of the likely consequences of authentication errors and misuse of credentials. Level 1 provides the lowest level of assurance and level 4 the highest.

A LoA value for an authentication instance derived by FAME is passed to an authorisation decision engine, such as PERMIS or GridSite, run at an SP (Service Provider) via a Shibboleth SAML message. The latter then uses the LoA value in the privilege allocation decision making process for the requester. Both PERMIS and GridSite have been enhanced to support LoA linked authorisation and access control decision making. In other words, both engines have now been extended such that an authorisation decision is based on the tuple: \{Subject, LoA, Target, Action\}.

In addition, FAME also supports single-sign-on (SSO); upon a successful authentication of a user, the user’s session will be issued with an sso cookie by FAME. As long as (1) the cookie is not expired, and (2) the LoA value required for the resource access is equal to, or lower than, the LoA value assigned to the user by FAME during the authentication process, the user will not need re-authentication for multiple resource accesses.

1.2. Workpackage 6 Description

The objectives of Workpackage 6 are, firstly, to integrate the components developed/extended in the project, namely, FAME, PERMIS and GridSite, in the Shibboleth infrastructure, and, secondly, to conduct case studies to evaluate the interoperation and efficacy of these components in achieving LoA linked fine-grained access control. This document reports the outcome of the integration and case studies.

Two case studies have been performed. The first focused on the integrated operations of FAME authentication and PERMIS and GridSite authorisation decision making via the Shibboleth infrastructure. This case study has demonstrated that the three components can inter-work and interoperate together to achieve LoA linked fine-grained access control. The second case study was motivated by our user partner, the CLEF-services project. The outcome from the second case study demonstrates that, in addition to controlling the access to Shibbolized Web-based resources, the FAME component can also be used to control the access to resources that do not have a Web front-end, e.g. those through the use of stand-alone applications.

2. Integration of FAME with PERMIS and GridSite

2.1. Integration Environment

For the purpose of the integration, we are running a Shibboleth Identity Provider (IdP) service protected by the FAME authentication, LoA derivation and single-sign-on services run at the School of Computer Science, the University of Manchester (UoM-CS). FAME integrates, and interoperates with, any authentication services that have a Web front-end. Currently, FAME is integrated with six standard authentication services, namely, the Kerberos authentication service [14], LDAP authentication system [11], SSL certificate-based authentication service [15] 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, the School of Physics and Astronomy, the University of Manchester (UoM-Physics).

We are also running two Shibbolethed Service Providers (SPs); one is installed with the PERMIS authorisation engine, and is run at the Computing Lab, the University of Kent (UoK), and the other is installed with the integrated GridSite and PERMIS authorisation engines, and is run at the UoM-Physics. 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 [6], we have also implemented our own WAYF service at UoM-CS for the purpose of our case studies.

2.2. FAME

For a user to access to different authentication services so as to obtain different values of LoA (depending on the authentication tokens/methods used) in order to access resources with varying levels of sensitivity, 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 [10]. FAME uses a database 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 used 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 [11]. 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 it 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 1: FAME identity mapping table

<table>
<thead>
<tr>
<th>ldap_id</th>
<th>ldap_attribute</th>
<th>alternative_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>ldapuserA</td>
<td>uid</td>
<td>testuser</td>
</tr>
<tr>
<td>ldapuserA</td>
<td>uid</td>
<td><a href="mailto:kerbuserA@CS.MAN.AC.UK">kerbuserA@CS.MAN.AC.UK</a></td>
</tr>
<tr>
<td>ldapuserA</td>
<td>uid</td>
<td>/C=GB/ST=Greater Manchester/O=University</td>
</tr>
</tbody>
</table>
In this example, the single user identified by the LDAP identity, \texttt{ldapuserA}, stored as the LDAP directory attribute \texttt{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 \texttt{testuser}, his identity with the Kerberos authentication service is \texttt{kerbuserA@CS.MAN.AC.UK}, and his identity with the X.509 certificate-based authentication service (expressed as his X.509 DN) is /\text{C}=\text{GB}/\text{ST}=\text{Greater Manchester}/\text{O}=\text{University of Manchester}/\text{OU}=\text{Dept. of Computer Science}/\text{CN}=	ext{FAME Project demo certificate v2/emailAddress}=\text{Alex@cs.man.ac.uk}. FAME uses the LDAP attribute \texttt{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 \texttt{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.

2.3. \textbf{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 \texttt{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, \texttt{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 [12] 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 1 below shows the allowable combinations.

<table>
<thead>
<tr>
<th>Allowable combinations</th>
<th>PERMIS Authorization</th>
<th>GridSite Authorization</th>
<th>Both Authorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Un/PW authentication</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>X.509 PKC authentication</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Shibbolized FAME authentication and IdP attribute push</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shibbolized FAME authentication and PERMIS AC pull</td>
<td>✓</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Shibbolised FAME authentication and GridSite credential pull</td>
<td>X</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Shibbolised FAME authentication with PERMIS and GridSite credential pull</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

2.4. The Shibboleth FAME-PERMIS-GridSite Integration

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.

2.5. Resource Access Procedures

This section describes a use case scenario on resource access using the FAME-PERMIS solution. The description is based on test scenario 3 discussed above, i.e. the web pages are protected by both PERMIS and GridSite authorisation engines. Other use case scenarios, such as those depicted by test scenarios 1 and 2, are similar to this one.

1. A user navigates his Web browser to the resource page at http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISGridsiteDemo/ (see Figure 1). There, the user is presented with links to PERMIS-protected pages, GridSite-protected pages and PERMIS and Gridsite-protected pages.

![FAME-PERMIS-GridSite demo page](image)

By clicking the link PERMIS and GridSite protected pages, the user is directed to the page containing links to 5 web pages (see Figure 2).
The public page is freely accessible, i.e. it requires no user authentication, and navigating that link the user is presented with Page 0 (see Figure 3).

However, if the user tries to access any of the remaining four pages, the Shibboleth infrastructure steps in. Let us say that the user wants to access Page 2 via link ‘LOA 2’ ([http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISGridsiteDemo /LOA2.html](http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISGridsiteDemo /LOA2.html)) from the main resource page. The Shibboleth SHIRE component at the SP detects that the user has not been
authenticated, so it directs the user to the WAYF service page at https://rpc56.cs.man.ac.uk/perl/WAYF.pl (shown in Figure 4).

![WAYF Service](image)

**Figure 4: WAYF (Where Are You From) selection page**

From the WAYF page, the user chooses FAME as his origin institution and gets redirected to the IdP at https://rpc56.cs.man.ac.uk/shibboleth-idp/SSO/ for authentication. This IdP is FAME-enabled, i.e. the Shibboleth HS is protected by FAME.
2. The user is presented with the FAME authentication page (Figure 5), where a list of available authentication services supported by the IdP are shown. Once the user has made the choice, FAME will connect the user to the chosen authentication service.

3. The user goes through the authentication procedure, as shown in Figure 6. Let us assume that, in this instance, the user has chosen the LDAP authentication system. Upon successful authentication, the user will be assigned a LoA value of 2 by FAME. This value is saved by the IdP for the user’s current session, which can be retrieved by the SP along with the user’s other attributes later on. In addition, FAME will issue the session an sso cookie so that as long as the cookie is not expired and as long as the LoA value issued for the session is not lower than what is required by the resource being accessed, the user will not need re-authentication.
4. The Shibboleth HS at the IdP creates a reference (i.e. a *handle*) for the user, which is included in the HTTP request. The user is then re-directed back to Page 2 ([http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISGridsiteDemo/LOA2.html](http://issrg-beta.cs.kent.ac.uk:8080/FamePERMISGridsiteDemo/LOA2.html)) that was initially requested by the user. This time, the Shibboleth SHIRE component at the SP detects the handle carried in the request and passes the handle to the SHAR component that uses the handle as the reference to retrieve the user’s attributes from the Shibboleth IdP.

The IdP releases all the attributes about the user that the SP is entitled to know, including the LoA attribute. The SP SHAR component passes these attributes to two authorisation decision engines – PERMIS and GridSite. Both engines process the received attributes in line with their access control policies. If and only if the user’s attributes have satisfied both sets of policies specified for the resource access by both decision engines, the user will be granted with the access (Figure 7). Note the increased level of zoom in Figure 7 as compared to Page 0 shown in Figure 3. It is worth noting that had the user had a LoA value higher than 2, the user would still be granted access to the page, as level 2 is the minimum LoA requirement for Page 2. However, in cases where the user has been authenticated with a method/token having a LoA value of 1, then the access to the page would be rejected. All the interactions between the SP and IdP during the above described process are transparent to the user.
This case study has demonstrated that:

1. FAME can successfully interact with both PERMIS and GridSite via Shibboleth and pass them the additional attribute LoA using the Shibboleth protocol.

2. Both PERMIS and GridSite authorisation engines are capable of processing the LoA attribute and make access control decisions based on it in line with their respective access control policies, such that:
   
   i) a user with LoA level 4 gets access to all five Web pages;
   ii) a user with LoA level 3 gets access to Pages 3, 2, 1 and 0;
   iii) a user with LoA level 2 gets access to Pages 2, 1 and 0;
   iv) a user with LoA level 1 gets access Pages 1 and 0;
   v) an unauthenticated user (i.e. with LoA level effectively equal to 0) can only get access to Page 0.

3. PERMIS and GridSite authorisation engines can co-exist on the same web server and interoperate to protect the resources hosted on the server. In this case, only when both engines grant the user access, can the user proceed. Should one engine refuse to grant the access, the access request will fail.
3. Using FAME to protect stand-alone applications

This use case is motivated by a problem raised by our user partner, the CLEF-services (Clinical E-Science Framework stage II) project [3].

3.1. CLEF-Services

The CLEF-Services project provides knowledge resources and electronic health records to clinical professionals and biomedical scientists, health care enterprises and the public at large. Resources it provides include a combination of results of clinical, genetic and genomic research and evidence-based health care and are stored in the central CLEF Repository. Some of these resources are highly confidential and should not be made available to the general public and thus must be protected from uncontrolled access.

To access the data in the Repository containing images, clinical and genomic data, and to allow clinicians to undertake statistical analyses and clinical research work, the CLEF-services project has developed a set of stand-alone applications, collectively called CLEF Thick Client (CLEF-TC). These applications are written mainly in JAVA, and are executed on users’ PCs. Due to the way that these applications are designed, 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 designed a FAME-CLEF architecture [8] so that FAME can be used to protect stand-alone applications, such as CLEF-TC, in addition to those with a web front-end.

3.2. FAME-CLEF Architecture

FAME is a Web-centric flexible authentication system capable of supporting LoA lined fine-grained access control for Web resources. The FAME-CLEF architecture is aimed at harnessing the full security capabilities of Web-centric FAME into the CLEF suite of thick clients (CLEF-TC) while offering a seamless integration process with no further modifications of CLEF-TC.

The essence of the FAME-CLEF integration, as shown in Figure 8 below, is to combine the use of FAME and a firewall to control the access to the remote CLEF Repository, which is accessed from the CLEF-TC standalone applications on user hosts. The Repository is located behind the Firewall on a remote CLEF server. For a user to be allowed to go through the Firewall to access data in the CLEF Repository via CLEF-TC, the user would have to first pass through the FAME authentication procedure. Once the authentication is successful, the Firewall will be signalled to temporarily unblock the IP address of the user’s host executing the CLEF-TC, and thereby all subsequent access requests to the CLEF Repository sitting behind the Firewall will be allowed to pass the Firewall. As soon as the access session closes (e.g. when the user closes the CLEF-TC), the Firewall will be set back to the ‘block’ state and further accesses from the given IP address will be forbidden until the user authenticates him/herself again.
3.3. FAME-CLEF Architectural Components

The FAME-CLEF architecture (as shown in Figure 8 above) 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 ongoing 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.

**FAME authentication service.** The FAME authentication service is an Apache-Perl Web Server module. It can be configured to integrate and interoperate with a wide range of authentication services with varying levels of authentication strengths; in fact, it can be integrated with any authentication services or modules with a web front-end. FAME can be used to protect Web resources, e.g. via protecting the Shibboleth HS, that require authentication strength linked fine-grained access control.

In this case study, FAME is used to protect the CGI-WebApp application that is designed to securely notify a Firewall protecting the CLEF Repository (1) the identities of, and IP addresses of hosts used by, the users who have gone through authentication with FAME; and (2) the events when the users close their CLEF-TC/Repository sessions.

For a single CLEF-TC/Repository session, FAME-protected CGI-WebApp may be contacted twice. On the first occasion, the CGI-WebApp is invoked (via the CLEF-Wrapper) to notify the user’s host IP address and his/her identity to the Firewall so that the latter can unblock the IP address. When the user finishes with the session and/or terminates the session, the CGI-WebApp will be contacted again (also via the CLEF-Wrapper), but this time it is invoked to notify the Firewall that the session is closed, so that the Firewall will re-block the IP address.

A special single-sign-on (SSO) cookie has been designed and used such that the cookie is embedded inside the user’s HTTP request when the user makes the first (i.e. firewall unblocking) request. Thus, when the same user makes the second (i.e. firewall blocking) request, re-authentication is no longer necessary. The validity period of this sso cookie is set to the same value as the timeout value used by the Firewall. So if a user’s session lasts longer than the validity period set in the sso cookie, the Firewall will timeout and block the user’s host IP address, and the user would have to re-authenticate before the Firewall is unblocked again.

**FAME-protected CGI-WebApp.** CGI-WebApp is a web application written in 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 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.

### 3.3. CLEF Repository Access Procedure

The following explains the steps of the CLEF Repository access procedure.

1. A user double clicks a desktop icon that invokes our custom CLEF-Wrapper application.
2. CLEF-Wrapper opens a browser instance to try to access the URL of CGI-WebApp. Since CGI-WebApp is FAME-protected, the user must go through an authentication process with FAME successfully before the user’s request is allowed through to the URL of the CGI-WebApp.

![Figure 8: CLEF-Wrapper invokes FAME prompting user for authentication](image)

3. CGI-WebApp is invoked, and it retrieves the IP address of the user’s host from the REMOTE_ADDR environment variable and obtains the user’s identity from the REMOTE_USER environment variable, and sends them to the Firewall Daemon via a secure (encrypted) XML-Remote Procedure Call (XML-RPC) invocation message.
4. The Firewall Daemon receives the notification message, and modifies the existing firewall rules to unblock the IP address of the user’s host so that the user can make the connection to the CLEF Repository. In addition, the Firewall Daemon logs the user’s identity and the access connection time for audit purpose.
5. The CLEF-Wrapper on the user’s host executes the CLEF-TC client.
6. The CLEF-TC gains access to the CLEF Repository protected by the Firewall.
7. The CLEF-Wrapper will also detect if the user has finished with the current access session and/or has closed the CLEF-TC client, in which case it will issue a logout message to the FAME-protected CGI-WebApp.
8. The CGI-WebApp notifies the Firewall Daemon which will then proceed to block further accesses by the corresponding IP address.

The FAME Firewall Daemon also has a mechanism to detect periods of inactivity for an allowed IP address which can be due to a CLEF-TC client aborting abnormally or user’s leaving their PC on for long period of time. If such an event is detected, the FAME Firewall Daemon will refresh the firewall rules to deny connections from the IP addresses concerned.

This case study has demonstrated that:

(1) FAME is an independent component that can be successfully deployed in an environment outside the scope designed for the FAME-PERMIS project and it can be used both within and outside the Shibboleth framework.

(2) FAME can be used to provide authentication not only for Web-based resources, but for standalone applications as well.

References