UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

ACCESS CONTROL SYSTEM FOR THE EPIDEMIC MARKETPLACE

Carlos André Galinha Jorge dos Santos

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

2013

UNIVERSIDADE DE LISBOA
Faculdade de Ciências
Departamento de Informática

ACCESS CONTROL SYSTEM FOR THE EPIDEMIC MARKETPLACE

Carlos André Galinha Jorge dos Santos

DISSERTAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA
Especialização em Arquitectura, Sistemas e Redes de Computadores

Dissertação orientada pela Prof. Doutora Maria Dulce Pedroso Domingos
e pelo Prof. Doutor Mário Jorge Gaspar da Silva

2013

Agradecimentos

Ao longo destes nove meses deparei-me com vários problemas, muitos dos quais nunca teria sido capaz de resolver sozinho. Estes problemas não foram só relacionados com o meu trabalho mas também problemas pessoais e por isso agradeço aqui a todos os que me ajudaram e que me deram força durante este período.

Quero dar um agradecimento especial à Dulce pela orientação e pelo seu pragmatismo que me trouxeram aqui onde estou, mostrando-se sempre disponível para me esclarecer. Quero agradecer ao Mário, ao Zamite, e ao Paulo Graça pela ajuda que me deram ao longo deste percurso.

Quero agradecer ao Comité Europeu pelo apoio financeiro ao EPIWORK sob o Seventh Framework Programme (Bolsa #231807), e à FCT pelo apoio financeiro do Programa de Financiamento Multianual.

Quero agradecer à minha família, principalmente ao meu pai e à minha mãe pelo suporte moral incondicional e pelos valores morais que desde novo me embutiram.

Quero agradecer também a um conjunto de amigos que ao longo do meu percurso académico me proporcionaram momentos inesquecíveis. Quero agradecer ao Cabaço, ao Pedro Marquês, ao Reis, ao Monteiro, ao Faria, ao Faísca, ao Telmo, ao Marcos, ao Guns, ao Antunes, ao Tareco, ao JP e ao Francisco Cunha. Quero também deixar um agradecimento especial ao Eduardo Matos, ao Saraiva, ao Eduardo Ferreira e ao Alexandre da Cunha.

Por último mas de forma nenhuma menos importante quero agradecer à minha namorada Tânia que me apoiou incondicionalmente e sempre com um sorriso.

A todos estes e aos demais agradeço por tudo, obrigado.

À Maria e à Margarida.

Resumo

A Epidemic Marketplace (EM) é uma plataforma de integração e partilha de dados epidemiológicos. As questões da privacidade constituem sempre um aspecto muito delicado nos repositórios de plataformas desta natureza, já que envolvem a partilha de dados sensíveis. Os utilizadores requerem que lhes seja assegurado o acesso aos seus dados de acordo com políticas de acesso bem definidas. Para suportar tal requisito, o modelo de controlo de acesso suportado pela EM é baseado em grupos (GBAC). Numa primeira versão da plataforma, os recursos apenas podiam ser partilhados com grupos estáticos, o que limitava a expressividade das especificações. Além disso, a plataforma tinha problemas de desempenho que derivavam de uma implementação inicial, não escalável, do sistema de controlo de acesso. Neste trabalho, apresentam-se as soluções desenvolvidas para aumento da escalabilidade da EM e fornecimento de mecanismos mais expressivos para a partilha de recursos através da especificação de grupos dinâmicos.

Dada a popularidade das redes sociais, a utilização dos grupos dinâmicos foi estendida para possibilitar a sua integração com estas redes, permitindo que os utilizadores da EM criem grupos baseados em ligações das redes sociais.

A EM foi desenvolvida no âmbito do projecto Europeu Epiwork, que teve como objectivos monitorar surtos epidemiológicos, guardar os dados recolhidos e utilizá-los em modelos matemáticos destinados a simular e a melhor entender a disseminação de doenças.

Palavras-chave: Epidemiologia, Controlo de Acesso, Controlo de Acesso Baseado em Grupos, Redes Sociais, Serviços Web, Partilha de Informação

Resumo estendido

A Epidemic Marketplace (EM) é uma plataforma de integração e partilha de dados epidemiológicos. Contudo, os donos dos dados não os partilham facilmente devido à sensibilidade / confidencialidade de alguns destes dados, mas sobretudo, devido ao seu valor científico que lhes garante a singularidade do seu trabalho. Para suportar tal requisito, o modelo de controlo de acesso suportado pela EM é baseado em grupos (GBAC). No entanto, quer o modelo, quer a sua implementação encontram-se numa fase embrionária.

A EM tem como base um repositório Fedora Commons e um motor de indexação Solr, o Fedora comunica com o Solr através do Fedora Generic Search service (GSearch). Sobre esta camada encontra-se a camada dos serviços web, esta camada está exposta à internet e serve de intermediário entre a camada interior e clientes HTTP, aplicações do Epiwork, e outros fornecedores de dados. Ambas as camadas do repositório e dos serviços web comunicam com um servidor LDAP onde são guardados os utilizadores e os grupos da EM. As aplicações do EM assim como o site da EM comunicam também com o servidor LDAP.

O modelo de controlo de acesso da EM é baseado em grupos, estes grupos têm três tipos de visibilidade: Privada, em que apenas o criador do grupo consegue ver o grupo; Visibilidade para o Grupo, em que todos os membros do grupo conseguem ver o grupo e partilhar recursos com aquele grupo; e Pública, em que todos os utilizadores do repositório podem ver o grupo e partilhar recursos com ele. Além disto, os utilizadores da EM podem ainda partilhar com os seus grupos colecções de recursos.

Os recursos da EM são guardados no repositório Fedora e contêm vários datastreams, dos quais nós distinguimos dados de metadados. Metadados são os datastreams que contêm informação sobre os dados. Nomeadamente os datastreams 'EM', 'RELS-EXT', 'Request', 'DC', 'FESLPOLICY' são considerados metadados pois contêm informação sobre o recurso e sobre os seus dados.

O sistema de controlo de acesso da EM baseia-se nas políticas de acesso definidas no repositório. Estas políticas encontram-se na linguagem XACML. Uma política em XACML possui dois componentes importantes: o alvo, onde se podem definir pedidos de decisão para o recurso identificado (sendo que neste caso o recurso é um termo genérico); e a regra, que é composta por um alvo, um efeito e uma condição.

Na segunda versão da EM (EMv2), era possível encontrar algumas lacunas na implementação e alguns problemas de desempenho. Além disso também não era possível partilhar recursos com utilizadores sem ser pelo intermediário de um grupo. Além disto, os grupos eram estáticos e por isso os seus membros tinham de ser definidos individualmente. Todos estes aspectos motivaram uma evolução do modelo e do sistema de controlo de acesso.

Comecei por estender o modelo de controlo de acesso para permitir a partilha de recursos com utilizadores individualmente. Alterei a estrutura dos nós no servidor LDAP de forma a resolver alguns problemas, alterei a forma como as políticas de controlo de acesso eram guardadas no Fedora melhorando bastante o desempenho dos serviços web de acesso aos recursos da EM, criei indíces de alguns atributos do LDAP que eram utilizados nas buscas de utilizadores em grupos melhorando bastante o desempenho dos serviços de controlo de acesso e de acesso aos recursos, e normalizei os formatos e conteúdo de entrada / saída dos serviços web. Ainda relativamente aos serviços web, melhorei o serviço de upload através da utilização de threads melhorando bastante também o seu desempenho. Criei ainda serviços web como o upload binário que permite ao utilizador carregar datastreams para um recurso da EM, e resolvi problemas de segurança no serviço de criação de grupos. Consolidei também o sistema de controlo de acesso da EM, adicionando funcionalidades para gerir grupos, tais como a adição e remoção de membros de grupos, e ainda as funcionalidades de criação / remoção de grupos.

De modo a flexibilizar a definição de grupos, estendemos o modelo de controlo de acesso da EM com grupos dinâmicos. Ao contrário do que acontece com grupos estáticos onde os membros são enumerados individualmente, o conjunto dos membros destes grupos é definido com regras. Dada a popularidade das redes sociais, a utilização dos grupos dinâmicos foi estendida para possibilitar a sua integração com estas redes, permitindo que os utilizadores da EM criem grupos baseados em ligações das redes sociais.

A EM foi desenvolvida no âmbito do projecto Epiwork, que teve como objectivos monitorizar surtos epidemiológicos, guardar os dados recolhidos e utilizá-los em modelos matemáticos destinados a simular e a melhor entender a disseminação de doenças.

Palavras-chave: Epidemiologia, Controlo de Acesso, Controlo de Acesso Baseado em Grupos, Redes Sociais, Serviços Web, Partilha de Informação

Abstract

The Epidemic Marketplace (EM) is a platform for integrating and sharing epidemiological data. Privacy issues are always a delicate matter when users intend to store sensitive data in such repositories. The users require assurance that their data access will always be in compliance with defined policies. The access control model of the EM uses Group-Based Access Control (GBAC). However, in an initial version of the platform resources could only be shared with static groups, leading to a lack of expressiveness. In addition, the EM platform had performance limitations that derived from using a non-scalable access control system implementation which could only perform simple access control changes. This work reports how performance issues with the platform have been solved and its scalability improved. In addition, EM users have the possibility of sharing their resources with dynamic groups, which, being rule based, provide more expressive mechanisms to share data. Given the current popularity of Social Networks, dynamic groups have been integrated with Social Networks, enabling EM users to create groups based on Social connections, obtained from Social Networks. Such groups rely on user approval for granting EM access to Social Network data. The EM has been developed in part within the EU-funded Epiwork project, whose main concerns include monitoring epidemiological outbreaks, storing that data and feeding it to mathematical models for simulating and better understanding the dissemination of diseases.

Keywords: Epidemiology, Access Control, Group Based Access Control, Social Networks, Web Services, Information Sharing

Contents

List of Figures

Listings

Chapter 1

Introduction

In scientific collaborative repositories access control has always been a sensitive matter, in the Epidemic Marketplace (EM) this is no exception. The EM is part of the Epiwork project, which is divided into seven Work Packages (WPs). The EM belongs to the WP3 and its purpose is to store epidemiological data. Access to this data has to be restricted by access control policies.

The EM uses a Discretionary Group-Based Access Control (GBAC) to ensure to its users that accesses to their data always follows the policies they define. It is very important to create a trust relation with our users by ensuring both that their data is secure from unauthorized access and that they are always in control of the policies that define access to their data.

Despite the EM being an integration and epidemiological data sharing platform, data owners will not share their data easily given its sensibility / confidentiality, but mostly because of the scientific value which grants them the singularity of their work.

This first chapter has the objective of presenting the motivation and the objectives of the project which originated this dissertation . I also present the contributions that resulted from our work, ending with the listing of the structure of the next chapters.

1.1 Motivation

The EM uses a GBAC model in order to provide its users with the ability to share their resources with each other. The GBAC model has shown its value in the EM. In scientific repositories it is necessary to promote information sharing and the GBAC model fits this need because it provides the users with effective means to share the resources they own with other users.

In a first phase the EM began by having a discretionary GBAC. In its second version / release, the Epidemic Marketplace version 2 (EMv2), it evolved into a decentralized discretionary GBAC. However, while the number of rules and resources increased, the performance decreased. Particularly the access control related web services, which were very slow. This problem had to do with the architecture of the Access Control approach at that time. For each resource shared in the EM, multiple resources had to be created to store access control rules. These rules define the policies of the resource being shared. This lead to a non-scalable Access Control system and motivated an evolution on the EM AC model.

Also, the platform's abstract character suggests the use of an access control model that eases the sharing of information while taking advantage of the ever growing use of Social Networks. This could increase the EM's expressiveness when it comes to sharing resources. The EMv2 provides their users with effective means to share resources, however they could be enhanced. The sum of all these enhancements was released in EM version 3 (EMv3).

1.2 Objectives and Approach

The objective of this work is to create an access control system that ensures data owners the privacy of their data while easing and promoting its sharing in a controlled environment.

Next I present the specific objectives that originated this dissertation, the challenges they present and the methods to approach each one of them.

1. **Solve the EM performance issues:** the EM presented performance issues in several services. Challenges:

    - Identify the source of the performance issues.
    - Identify the changes that need to be implemented in the services that are affecting the perfomance.
    - Implement the changes that need to be made.

    Approach:

    - I began by measuring response times in our web services, after this step I was able to identify the Fedora Commons repository and the LDAP server as the backend services that were affecting the performance of the EM. Then I analyzed those services to identify the source of the problems and fixed them.

2. **Consolidate the decentralized GBAC model of the EM:** it should allow users to manage the groups they create. Challenges:

    - Develop web services to allow users to edit and to remove groups.
    - Develop web services to allow users to add and remove members from the groups.
    - Develop the web interfaces for these web services and integrate them in the front-end.

    Approach:

    - I began by modeling the format of the inputs and outputs of the web services. Then I designed the process and at last developed the modeled web services.

3. **Extend the access control system of the EM to allow the share of resources with single users:** it should allow users to share resources they own with single users without having to create groups. Challenges:

    - Study current access control approaches based on sharing with single users.
    - Choose an approach that fits the requirements of the EM.
    - Implement that feature, extending the EM access control model.

    Approach:

    - I began by searching for approaches that are currently used in such cases and identified one that suited the EM. Then I implemented this feature.

4. **Extend the GBAC model of the EM to allow users to share resources with dynamic groups:** it should allow users to create groups whose membership is defined by rules. Challenges:

    - Study current access control approaches based on sharing with dynamic groups.
    - Choose an approach that fits the requirements of the EM.
    - Implement that feature, extending the EM access control system.

    Approach:

    - I began by searching for approaches that are currently used in such cases and identified one that suited the EM. After identifying an LDAP-based solution that seemed to satisfy this objective.

5. **Extend the GBAC system of the EM to allow users to share resources with Social Network groups:** it should allow users to share their resources with dynamic groups whose membership is based in their social networks connections. Challenges:

    - Study current access control approaches based on dynamic groups with social information.
    - Study the current approaches to integrate social networks with other platforms.
    - Identify the approaches that make sense in the context of the EM.
    - Develop a social network dynamic group approach for the EM.
    - Implement that solution in the EM.

    Approach:

    - I began by searching for current access control approaches based on dynamic groups with social information, then I identified the necessary tools to perform such integration. After this I identified an approach that made sense in the EM's context, developed a solution based on my approach and implemented it in the EM.

In order to achieve such objective I will enhance the expressiveness of the access control system by extending the current EM model to include dynamic groups.

With this objective in mind we intend to integrate Social Networks concepts in a discretionary access control system based in groups.

1.3 Contributions

In this work, I identified the drawbacks that were affecting the EM's performance. I've done this by implementing a new access control structure that dealt better with the growth of workload. This changes were made in the web services by creating a new access control web service, at the Fedora repository by changing the organization of the resources policies, and in the LDAP server by changing its structure.

In a first iteration of enhancements to the access control system, I consolidated the EM's GBAC by implementing the features that were missing. After being satisfied with the performance and behavior of the newly added features, I moved on to the second iteration, on which I enhanced the EM's expressiveness by allowing EM users to create dynamic groups, groups whose membership is defined by rules. In a third and last iteration, I extended the EM's access control system to allow users to share their resources with their Social Connections (connections from their Social Networks).

In addition, I developed new web services such as the binary upload which allowed EM Users to upload datastreams to the EM. This was an important feature for the integration with the Gleamviz team. I developed the tag cloud web service which returned the epidemiological terms that had more occurrences in the EM's resources, this web service was later discontinued. I also developed the Epimarketplace Crawler (EMCrawler), which crawls the web for articles of interest in order to add them to the EM's repository.

1.4 Document Structure

This document is organized as follows:

- Chapter 2 (Epidemic Marketplace) – In this chapter I present the Application Architecture of the EM; the Access Control Model and some useful concepts and terms for the full understanding of the Epidemic Marketplace Platform; the Access Control System with a brief introduction to XACML and an extended overview of the EM's Access Control Architecture. Also I present some of the work regarding the access control model that was developed prior to my integration in the EM, namely the second instance of its access control.

- Chapter 3 (Consolidation) – In this chapter I present the extensions I added to the EM's Access Control Model; the Access Control System Changes regarding the LDAP's structure, the Fedora policies, the LDAP indexes and the Web Services normalization; I also present the Access Control Extensions and explain how they improved the EM's performance.

- Chapter 4 (Dynamic Groups) – Here I present the access control enhancements that I developed regarding dynamic groups, the generic dynamic group model; the definition of dynamic groups with LDAP attributes; and the definition of dynamic groups with Social attributes.

- Chapter 5 (Conclusion) – In the conclusion I will make an overview of my work and criticize it with a very pragmatic approach.

Chapter 2

Epidemic Marketplace

In this chapter I present the Application Architecture of the EM, the EM's Access Control model and other important access control models, the EM's Access Control System, and the access control model of the EMv2 which were developed prior to my integration in the project.

2.1 Application Architecture

The Epidemic Marketplace's architecture has to ensure to its users that their data is kept safe and undisclosed. The EM's application architecture is divided into three layers, in which, all interact directly with the LDAP server. The EM's application architecture is shown in Figure 2.1.

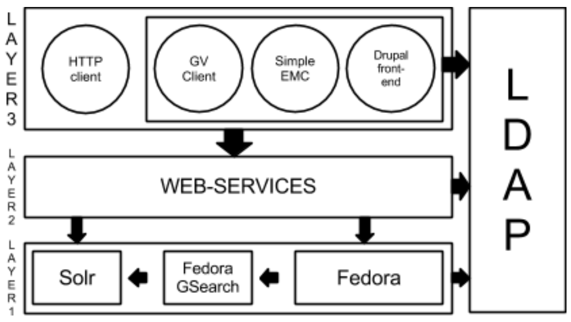
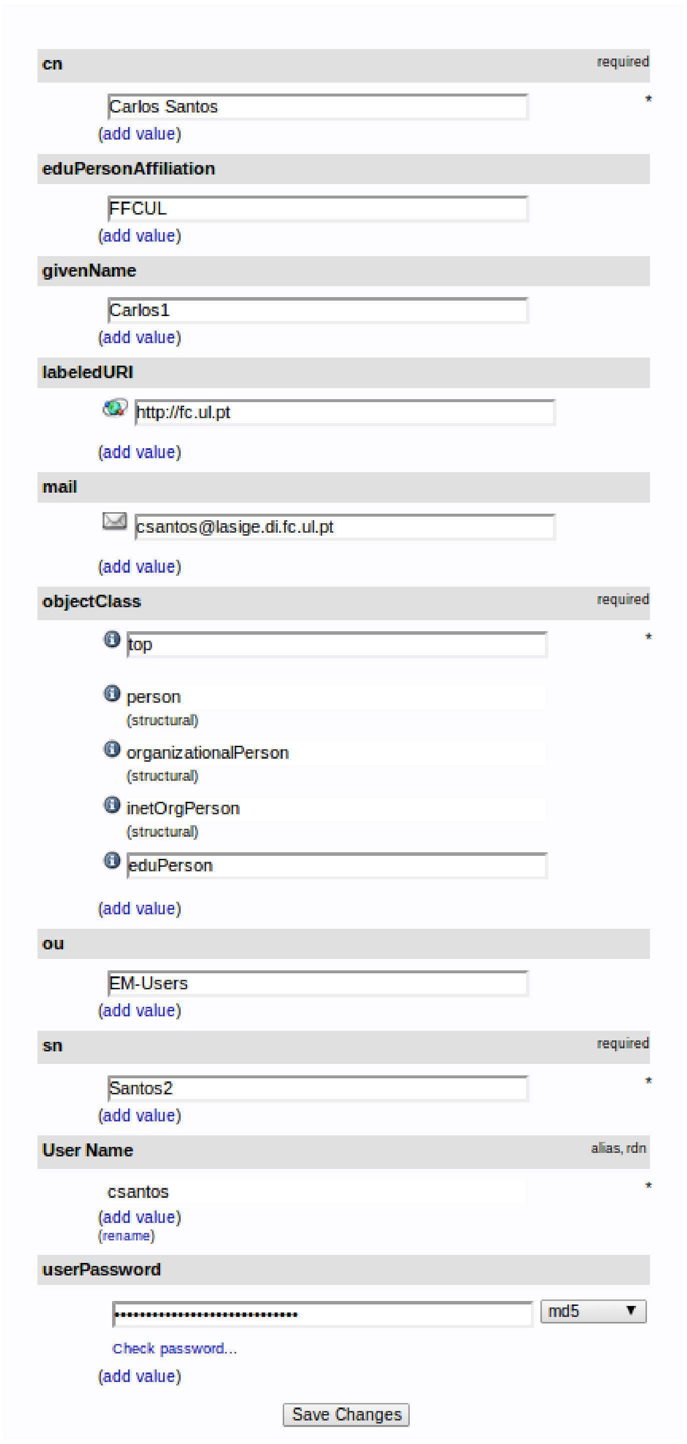


Figure 2.1: EM's application architecture

2.1.1 LDAP

LDAP is a directory oriented service. In an LDAP server, entries are stored hierarchically following a tree model. Each entry consists in a set of attributes with values. Entries can refer to people, organizations, groups of people, etc.. Two important terms in the LDAP domain are the DN (Distinguished Name), which is the full path of the entry (eg: cn=csantos,ou=EM users,dc=ldap,dc=epiwork,dc=eu), and the CN (Common Name) which is the common designator of the name and unlike the DN is not required to be unique.



cn required

Carlos Santos *
(add value)

eduPersonAffiliation

FFCUL
(add value)

givenName

Carlos1
(add value)

labeledURI

http://fc.ul.pt
(add value)

mail

csantos@lasige.di.fc.ul.pt
(add value)

objectClass required

top *
person
(structural)
organizationalPerson
(structural)
inetOrgPerson
(structural)
eduPerson
(add value)

ou

EM-Users
(add value)

sn required

Santos2 *
(add value)

User Name alias, rdn

csantos *
(add value)
(rename)

userPassword

•••••••••••••••••••••••••••••••• md5 ▼
Check password...
(add value)

Save Changes

Figure 2.2: LDAP user example

A typical Client-LDAP work-flow begins with the Client binding with the LDAP server. Then the Client performs a set of operations (such as modify, delete, search) and then the Client unbinds with the LDAP server completing the work-flow.

EM users are stored in an LDAP server. In Figure 2.2 we can see the users' representation in LDAP. Users have ten fields, the *CN* (Common Name) which is used both internally and at the web services level; the *eduPersonAffiliation*, where the user can add its affiliations (e.g. FFCUL); the *givenName* which is the name used at the interface level, it's this name that is used in the front-end; the *labeledURI* field which allows the user to associate a URI with his account; the *mail* where the user can store his email; the *objectClass* which defines the entry's type; the *ou* (Organizational Unit) which is used to categorize entries, in this case the *ou* represents the EM-Users; the *sn* (surname); the *User Name*; and at last the *userPassword*.

2.1.2 Layer1

In this layer there are three applications: a Fedora server, which is where the resources are stored; the fedora generic search application, which is responsible for communicating with the last application of this layer: the Solr search engine. The Solr search engine provides a powerful set of tools for high performance searches. In this section I detail these three applications.

Fedora Repository

EM uses a Fedora Commons [2] solution integrated with LDAP. Fedora Commons is our back-end repository, where resources are stored and managed. User related information, such as authentication credentials, as well as groups are stored in the LDAP server. Fedora uses a Digital Object Model to store the data.

The Fedora's Digital Objects are composed as follows:
- Persistent ID (PID) - The Digital Object Identifier.
- Object Properties - System properties to manage and track the object.
- Datastream(s) - Content Item(s).

An EM resource is an abstraction which is represented as one Fedora Digital Object. Each EM resource has a set of mandatory Datastreams. Fedora reserves five Datastream Identifiers for itself: 'DC', 'AUDIT', 'RELS-EXT', 'RELS-INT' and 'FESLPOLICY'. On top of those Datastreams, EM resources have another two reserved Datastream Identifiers: 'EM' and 'Request', which contain the resource's metadata.

Every EM resource has one EM datastream, this is the datastream on which we keep the resources's metadata. Not all resources have a Request Datastream, only those that are requests for resources (I will elaborate this concept later on). The contents of EM datastreams are in eXtensive Markup Language (XML). One example is shown bellow:

Listing 2.1: EM's datastream example

```
<?xml version="1.0" encoding="UTF-8"?>
<em:em xmlns:em="http://epiwork.di.fc.ul.pt/metadata/" em:
   version="2.0">
  <em:title>My test resource</em:title>
  <em:identifier>empid:1053</em:identifier>
  <em:generalDescription>
    <em:abstract>Small abstract text</em:abstract>
    <em:citation>citation text</em:citation>
    <em:description>description text</em:description>
    <em:DOI>DOI</em:DOI>
    <em:format>Format</em:format>
    <em:ISBN>978-3-16-148410-0</em:ISBN>
    <em:ISSN>ISSN</em:ISSN>
    <em:language>en</em:language>
    <em:pubmedID>is PubmedID</em:pubmedID>
    <em:subject>Meta-Information</em:subject>
    <em:type>Mobility data</em:type>
    <em:URL>http://fc.ul.pt</em:URL>
    <em:version>1.0</em:version>
  </em:generalDescription>
  <em:date>2012-07-17T00:00:00Z</em:date>
  <em:dateSubmitted>2012-07-17T14:54:56Z</em:dateSubmitted>
  <em:creator>
    <em:name>Paulo</em:name>
    <em:organisation>FFCUL</em:organisation>
    <em:URL>http://fc.ul.pt</em:URL>
  </em:creator>
  <em:organisation>
    <em:name>is Organisation Name</em:name>
    <em:URL>http://www.fc.ul.pt</em:URL>
  </em:organisation>
  <em:uploader>
    <em:name>Paulo Graa</em:name>
    <em:organisation>FFCUL</em:organisation>
  </em:uploader>
  <em:time>
    <em:from>1982-01-01T00:00:00Z</em:from>
    <em:to>1983-01-01T00:00:00Z</em:to>
    <em:moment />
  </em:time>
  <em:source>
    <em:name>source name</em:name>
    <em:URL>http://www.fc.ul.pt</em:URL>
    <em:description>source description</em:description>
  </em:source>
  <em:biologicalInformation>
    <em:diagnosticMethod>http://ncicb.nci.nih.gov/xml/owl/EVS/
       Thesaurus.owl#C64382</em:diagnosticMethod>
    <em:disease>http://purl.obolibrary.org/obo/DOID_0050143</em:
       disease>
    <em:symptom>http://purl.obolibrary.org/obo/SYMP_0000249</em:
       symptom>
    <em:drug>http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#
       C74599</em:drug>
    <em:host>http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#
       C74505</em:host>
    <em:pathogen>http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.
       owl#C74505</em:pathogen>
    <em:transmission>http://purl.obolibrary.org/obo/
       TRANS_0000008</em:transmission>
    <em:vaccine>http://purl.obolibrary.org/obo/VO_0012214</em:
       vaccine>
    <em:vector>http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.
       owl#C74505</em:vector>
  </em:biologicalInformation>
  <em:environment>http://purl.obolibrary.org/obo/ENVO_00000275</
     em:environment>
  <em:demography>http://purl.obolibrary.org/obo/OMRSE_00000002</
     em:demography>
  <em:socioEconomicCondition>http://ncicb.nci.nih.gov/xml/owl/
     EVS/Thesaurus.owl#C100743</em:socioEconomicCondition>
  <em:location>
    <em:country>PT</em:country>
    <em:place>Lisbon</em:place>
    <em:URI />
  </em:location>
  <em:bibliographicCitation>
    <em:citation>is citation</em:citation>
    <em:DOI>DOI</em:DOI>
    <em:PubmedID>pubmed id bibliographic</em:PubmedID>
  </em:bibliographicCitation>
  <em:rights>
    <em:rightsHolder>is rights holder</em:rightsHolder>
    <em:copyright>is copyright</em:copyright>
    <em:disclaimer>is disclaimer</em:disclaimer>
    <em:license>My license</em:license>
  </em:rights>
</em:em>
```

The Fedora Security Layer (FeSL) is the Fedora's authorization and authentication system. FeSL authentication is built on the Java Authentication and Authorization Service

(JAAS) [5], which authenticates EM's users against their LDAP credentials. FeSL extends and improves the OASIS XACML (see the XACML section 2.3.1).

Whenever Fedora receives a request for accessing a Digital Object's data, it binds with the LDAP server application to see if the requester belongs to any of the groups that the resource is shared with (if any). Then, using the Fedora Security Layer (FeSL), authorization might or might not be given for that user to access that Digital Object.

In order to fetch Digital Objects from Fedora, it provides a basic search engine which allows searching and browsing its repository. Even though this basic search is provided, the EM uses the Fedora Generic Search Service (GSearch) which is used as a middleware between the Fedora repository and the Solr [9] search engine.

GSearch and Solr

The Fedora Generic Search service (GSearch) makes searching digital contents in Fedora relatively easier. GSearch comes with a Solr plugin which interacts with this last. Solr is then used to power-up the EM's search feature.

Solr is a powerful open-source software developed by the Apache Lucene project. Some of the features Solr has that meet the EM's needs are a powerful full-text search, faceted search and highlighting.

Solr provides user friendly indexation with the aid of XML schema files (XSD) where one can easily define the fields to index and the type of those fields.

2.1.3 Layer2

This is the web services layer, the EM's Web Services are a critical feature of the Epidemic Marketplace. They are the middleware layer that intermediates between the Fedora Repository and the outer layer. Every request that is made to the Fedora Repository has to pass through the web services layer. The web services were developed in Python.

Web Services

The Web Services are **RESTful**, they use *HTTP methods* explicitly, are *stateless*, their API is *hypertext driven*, expose *directory structured-like URIs*, and transfer *XML*, *JavaScript Object Notation* (JSON), or both thus following the SOAP specification.

The Web Services are responsible for intermediating with HTTP Clients, Epiwork Applications and other data suppliers:

- **HTTP Clients -** These are the user applications that insert/retrieve epidemiological data in/from our platform.
- **Epiwork Applications -** Applications such as the Simple EM Client and the Gleamviz Client.
- **Other data suppliers -** Tools such as *online news*, *RSS feeds*, *ProMED Mail*, and other event generators.

2.1.4 Layer3

The third layer includes the EM's Drupal-based front-end which provides EM's users with a user friendly User Interface (UI), the Simple EM Client which is a Python tool that allows users to bypass the EM's UI, and the GLEaMviz Client whose simulations rely on the availability and behavior of the EM's web services.

Drupal front-end

EM provides a Drupal-based user interface front-end for interactive upload and manipulation of resources. The front-end is Drupal-based because it allows a modular development, which accelerated the development of the EM. Whenever we updated some software that interacted with Drupal, we just had to update the module (sometimes this wasn't even needed).

The website uses many modules, some are contrib (modules developed by other peers that were made available in the community), but most are custom modules that we developed in order to deal with our platform specific needs.

Simple EM Client

The Simple EM Client is a tool that enables an advanced user to bypass the Drupal front-end without having to deal directly with the Web Services. The Simple EM Client is built in Python and communicates with the EM through HTTP requests.

GLEaMviz Client

GLEaMviz [3] is a client-server software system that can model the world-wide spread of epidemics for human transmissible diseases like influenza like illnesses (ILI), offering extensive flexibility in the design of the compartmental model and scenario setup, including computationally-optimized numerical simulations based on high-resolution global demographic and mobility data. GLEaMviz uses the EM's web services to store/retrieve its simulations.

2.2 EMv2 Access Control Model limitations

In this section I describe the EMv2's group-based discretionary access control model. In this model, users assign permissions to groups of users over their own resources. However, the model wasn't fully implemented. Some of the features were missing, there was no feature for editing a group once it was created. Also resources could only be shared be shared with groups, a resource couldn't be shared directly with a user.

Also the model lacks of expressiveness because users could only share resources with static groups where they have to define all the members individually.

2.3 EM's Access Control Model

In this section I present two main access control models, the Discretionary Access Control (DAC) and the Role Based Access Control (RBAC) models and their variations. Also, I present the Access Control Model of the EM and the underlying concepts.

2.3.1 Access Control Models

An access control model implies the use of access control policies in order to define the system's behavior. Access control models define the formal representation of the access control policies and their working behavior. Access control languages allow to express access control policies. Regarding the definition of access control models for information platforms and scientific repositories we consider two main access control models:

- the **DAC** is based on the user identification and on authorizations (permissions assigned to that user) [20]. In this model resources have owners associated with them. In the *Ownership DAC* model, resource owners have the ability to set permissions over their resources. On the *Decentralized DAC* model, the owners have the power to delegate the ability to set permissions to other users.
  The traditional Group-Based Access Control [4] model is an evolution of the DAC model on which users can share resources which they own with groups of users.
  - **Group-Based Access Control (centralized)** The Centralized Group-Based Access Control approach allows users to share their resources with groups of users, however they rely on administrators for the creation and maintenance of those groups.
  - **Group-Based Access Control (decentralized)** On the other hand, a Decentralized Group-Based Access Control allows users to define groups to share the resources they own, thus achieving the same granularity as the traditional GBAC with better efficiency. There is no need for administrators intervention.

- on the **RBAC** model, the concept of Role is introduced. This is a widely-accepted approach [10] where each role has a set of permissions associated with it, then roles are assigned to users, and so the users use the set of permissions associated with the role(s) assigned to them.
  In this case, when access is solicited by an user or process, the user/process will use the permissions that are assigned to all the roles that are associated with him/it.

2.3.2 Concepts

Datastream is a component of a Fedora digital object. In the EM's perspective there are two sets of datastreams: *data* and *metadata*.

Metadata datastreams are finite and well defined: *EM*, *DC*, *Request*, *RELS-EXT* and *FESLPOLICY* are metadata datastreams. These datastreams are managed by the EM, EM users can't interact directly with them.

The **data datastreams** are all other datastreams, datastreams that the user uploads to their resources.

A **Principal** represents a user or a group of the repository.

An **Object** represents an entity a user may or may not have access to perform actions on, including collections, resources and their *metadata*.

Groups are categories of principals.

Actions are the types of access that users can perform on objects. In this model we consider the actions: read, write, create and delete. Object owners are granted all actions for their owned objects. The read action allows read-only accesses to objects such as:

1. Read action for a collection authorizes users to list the collection, and to view the contained metadata and resources.

2. Read action for a resource enables users to view the resource content and its metadata.

3. Read action for metadata enables users to view the metadata.

The write action is used to edit the contents of existing collections and resources:

1. Write action for a collection authorizes users to edit the collection, i.e. the contained metadata and resources.

2. Write action for a resource authorizes users to edit that resource content and its metadata.

3. Write action for metadata authorizes users to edit the metadata of an object.

The create action allows users to upload data datastreams to objects.
The delete action allows users to remove any object. In addition contained objects are also removed when the container object is deleted.

Group Management is decentralized and discretionary. Group owners have permissions to edit and delete his created groups. Furthermore, we define that groups have three types of visibility: Private, where only the group creator is able to see the group; Group Visible, where all the members of a group can see the group and share resources with that group; and Public, where everyone in the repository sees the group and can share resources with that group.

A **Permission** is defined by an action and an object, the action can be each one of the following: read, update, create or delete. The permission can be described by the pair *(action, object)*.

Authorization is the process of enforcing permissions over principals[17].

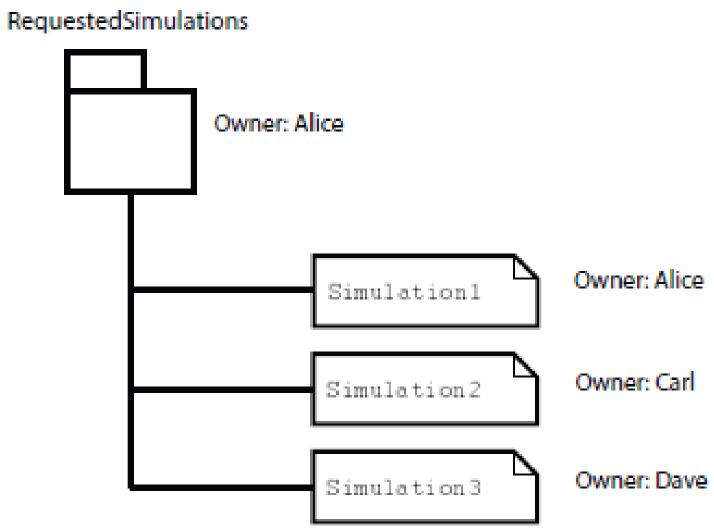


Figure 2.3: Collection usage example

Take the following scenario as an example: A team of experts is hired to create a set of simulations for a public health institution. *Alice*, a member of the team of simulation experts creates a group *SimulationExperts* = {*Alice*,*Carl*,*David*} which consists of the members of the team. *Alice* then creates a collection named *RequestedSimulations* and grants read and insert permissions to the *SimulationExperts* group, enabling them to read and create simulations inside the collection. Each of the experts creates a simulation resource inside the collection (see Figure 2.3). Because collection access rights are implicit for contained resources, *Alice* also has all access control rights over the created simulations.

Because our model is owner centered, *Alice*, the owner of the collection, can then set further access control permissions for her collection. Because the public health institution requires that the simulations be shared with them privately, *Alice* simply creates a new group *PHIUsers* = $\{Bob, Frank\}$ which consists of members of the public health institution (or simply use a pre-existing group if the institution created one) and gives them read access to her collection. Because container access rights are implicit on contained objects, *Bob* and *Frank* have read permissions over the contained resources without having to request them individually.

2.4 EM's Access Control System

In this section I present the Access Control System with an introduction to XACML and an overview of the EM's Access Control Architecture.

2.4.1 XACML

Policy languages have been studied for a long time. There are also some languages that can express access control policies such as XACML (eXtensible Access Control Markup Language)[11], X-RBAC (XML Role-Based Access Control)[15], KAoS(Knowledgeable Agent-oriented System)[14], Rei[16] and Ponder[18]. Among them, XACML is the current OASIS standard specification.

The OASIS eXtensible Access Control Markup Language (XACML) describes both a policy language and an access control decision request/response language. XACML defines an access control policy language implemented in XML which enables the formulation of queries to determine whether or not a given action should be allowed to a subject over a resource. The XACML format consists of two major components:

- **Target:** The set of decision requests, identified by definitions for resource, subject and action, that a rule, policy or policy set is intended to evaluate.
- **Rule(s):** A target, an effect and a condition. A component of a policy.

All the policies in the EM are permissive, they only contain positive rules. By default, everything that isn't explicitly permitted is forbidden. Bellow is a XACML rule example that gives *read* access to the *metadata* of the resource to the group *testgroup-267* and to the user *csantos*.

Listing 2.2: XACML rule example

```
<Rule Effect="Permit" RuleId="group@testgroup-267,user@csantos;
    meta;r">
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
        ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
          string-at-least-one-member-of">
        <ResourceAttributeDesignator AttributeId="urn:fedora:
            names:fedora:2.1:resource:datastream:id" DataType="
            http://www.w3.org/2001/XMLSchema#string"></
            ResourceAttributeDesignator>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
            string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">EM</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">DC</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">Request</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">RELS-EXT</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
          string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:names:
            fedora:2.1:action:id" DataType="http://www.w3.org
            /2001/XMLSchema#string"></ActionAttributeDesignator>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
            string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">readds</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
          or">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
          string-at-least-one-member-of">
        <SubjectAttributeDesignator AttributeId="memberOf"
            DataType="http://www.w3.org/2001/XMLSchema#string"></
            SubjectAttributeDesignator>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
            string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">testgroup-267</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
          string-at-least-one-member-of">
        <SubjectAttributeDesignator AttributeId="urn:oasis:names:
            tc:xacml:1.0:subject:subject-id" DataType="http://www.
            w3.org/2001/XMLSchema#string"/>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
            string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
              XMLSchema#string">csantos</AttributeValue>
        </Apply>
      </Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

The major actors in the XACML domain are the **PEP**, the **context handler**, the **PDP** and the **PAP**.

- The **policy enforcement point (PEP)** is the entity that performs **access control**, by making **decision requests** and enforcing **authorization decisions**.

- The **context handler** is the entity that converts **decision requests** in the native request format to the XACML canonical form and converts **authorization decisions** in the XACML canonical form to the native response format.

- The **policy decision point (PDP)** is the entity that evaluates **applicable policy** and renders an **authorization decision**.

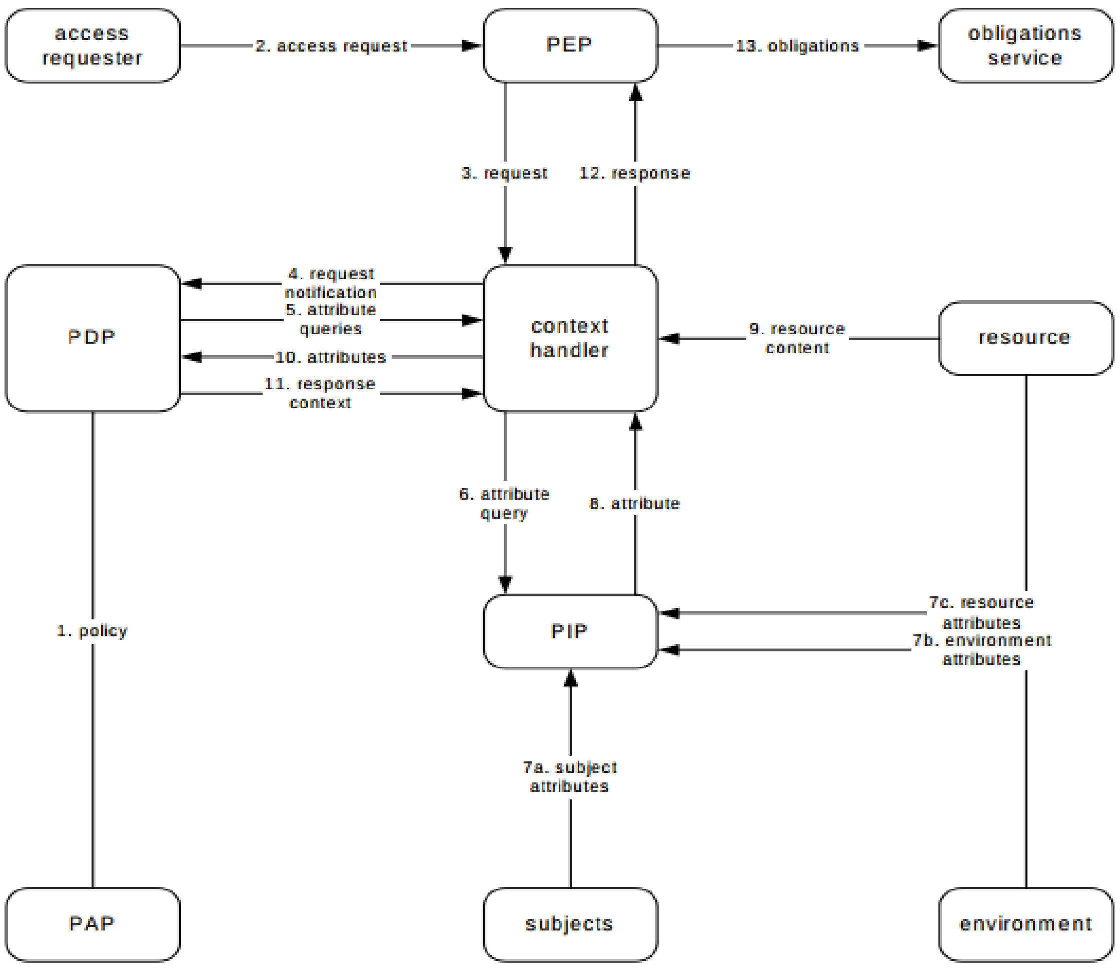


Figure 2.4: OASIS data-flow diagram

- The **policy information point (PIP)** is the entity that acts as a source of **attribute** values.
- The **policy administration point (PAP)** is the entity that creates a **policy** or **policy set**.

The major actors in the XACML domain are shown in Figure 2.4.
The data-flow operates by the following steps:

1. **PAPs** write **policies** and **policy sets** and make them available to the **PDP**. These **policies** or **policy sets** represent the complete policy for a specified **target**.
2. The access requester sends a request for access to the **PEP**.
3. The **PEP** sends the request for **access** to the **context handler** in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**.

4. The **context handler** constructs an XACML request **context** and sends it to the **PDP**.
5. The **PDP** requests any additional **subject**, **resource**, **action** and **environment attributes** from the **context handler**.
6. The context handler requests the attributes from a **PIP**.
7. The **PIP** obtains the requested **attributes**.
8. The **PIP** returns the requested **attributes** to the **context handler**.
9. Optionally, the **context handler** includes the **resource** in the **context**.
10. The **context handler** sends the requested **attributes** and (optionally) the **resource** to the **PDP**. The **PDP** evaluates the **policy**.
11. The **PDP** returns the response **context** (including the **authorization decision**) to the **context handler**.
12. The **context handler** translates the response **context** to the native response format of the **PEP**. The **context handler** returns the response to the **PEP**.
13. The **PEP** fulfills the **obligations**.
14. (Not shown) If **access** is permitted, then the **PEP** permits **access** to the **resource**; otherwise, it denies **access**.

2.4.2 EM's Access Control Architecture

In this subsection I explain how and where access control policies and EM groups are stored.

Fedora Policies

The policies that define access to EM resources are kept in Fedora inside the resources' FESLPOLICY datastreams. Each resource has one FESLPOLICY datastream. This datastream is considered *metadata* and as stated above about these types of datastreams, it cannot be managed directly by the users of the EM. Users must use the web services that the API of the EM provides to them.

These policies are composed of a target, which specifies the target resource, and a set of rules that define access to that resource.

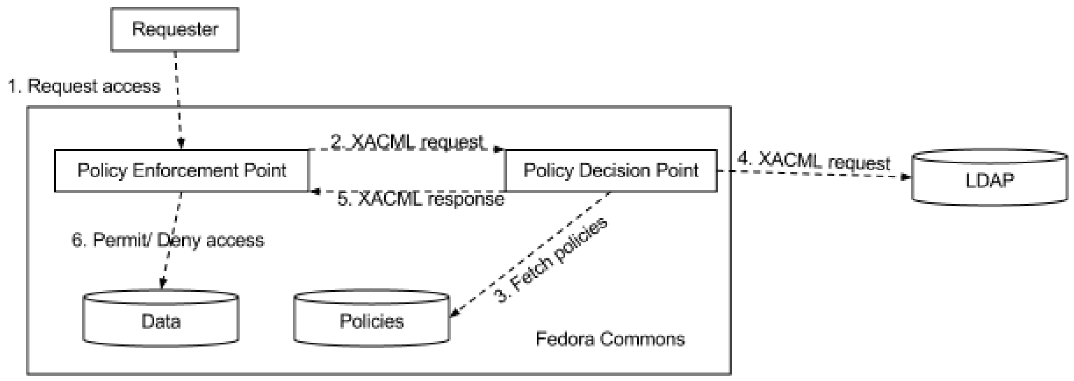


Figure 2.5: EMv2's FeSL data-flow

The authorization process begins with a request, then the request will hold at the Policy Enforcement Point (PEP). The PEP will form a XACML request based on the requester's attributes, the resource, the action and other information. The PEP will then send this request to the Policy Decision Point (PDP). It's the PDP that will determine whether or not access should be granted. Then the answer is returned to the PEP which will then allow or deny the requester's access to the resource. In Figure 2.5 it is shown the EMv2's FeSL data-flow.

2.4.3 Web Services

When a client wants to access a resource's datastream he/it can make a specific call to the API (eg: api.epimarketplace.net/rawfetch/pid/empid:xxx/datastream/EM) or he/it can use our Drupal-based front-end (eg: https://www.epimarketplace.net/resource/empid_xxx) which calls the same API web service. Both this calls will return either the requested datastream if access is approved or an access error otherwise.

However, for a better understanding of its permissions and to avoid interpretation errors, we also supply a web service api.epimarketplace.net/accesscontrol/listpolicies/empid:xxx which specifically requests for the policies of a resource. This enables us to provide an interface based on the client permissions (eg: showing only the actions he/it can perform).

The access control is enforced in this two approaches:

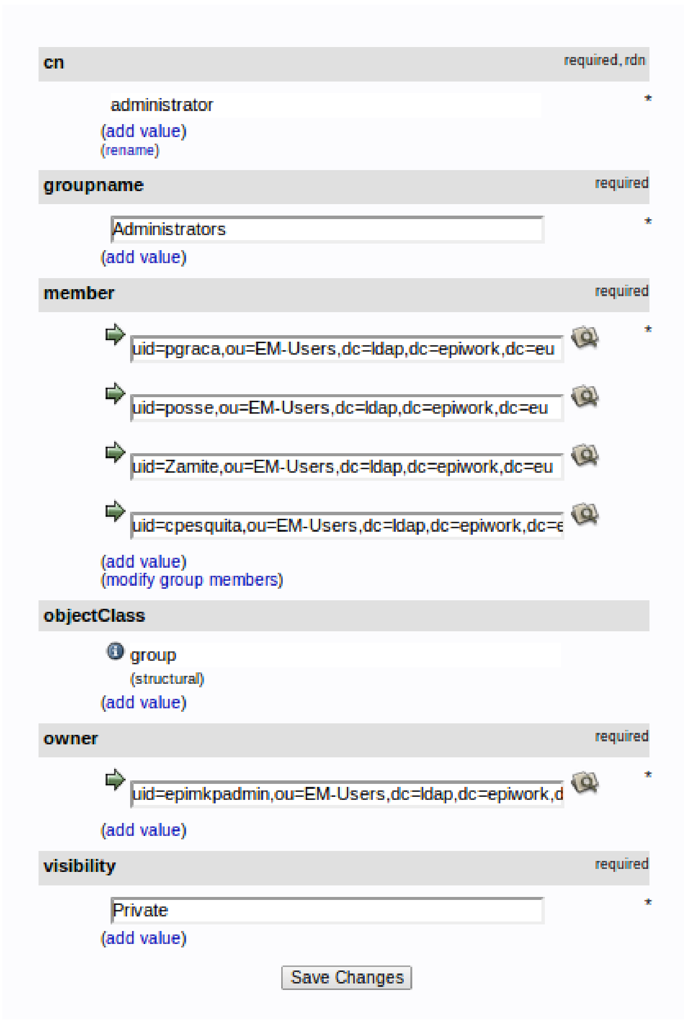
- **Case1:** A direct access to a resource's datastream.
  (eg. api.epimarketplace.net/rawfetch/pid/empid:xxx/datastream/EM)
- **Case2:** A request that specifically asks for the policies of a resource.
  (eg. api.epimarketplace.net/accesscontrol/listpolicies/empid:xxx)

In **case1** the request goes from the client to the API. Then the API forms a fedora request based on the user attributes and forwards the request to Fedora. Fedora then decides whether or not the user has access to the resource, returning the datastream if permission is granted.

In **case2** the resource's permissions are specifically requested. This information is only return to clients that have the privilege to access that data (such as resource owners, users that were granted access to the resource and administrators). This is used to know if the user has or has not access to the resource prior to actually trying to access it on fedora. This is useful for the Drupal-based interface in order to be able to show/ hide the options the user can perform over a resource.

EM Groups

EM Groups are stored in the LDAP server. All authenticated users can create groups in the EM. After the creation of the groups, users can share their resources with them, thus delegating the permissions to the members of those groups.



cn	required, rdn
administrator	*
(add value) (rename)	
groupname	required
Administrators	*
(add value)	
member	required
uid=pgraca,ou=EM-Users,dc=ldap,dc=epiwork,dc=eu	*
uid=posse,ou=EM-Users,dc=ldap,dc=epiwork,dc=eu	
uid=Zamite,ou=EM-Users,dc=ldap,dc=epiwork,dc=eu	
uid=cpesquita,ou=EM-Users,dc=ldap,dc=epiwork,dc=e	
(add value) (modify group members)	
objectClass	
group (structural)	
(add value)	
owner	required
uid=epimkpadmin,ou=EM-Users,dc=ldap,dc=epiwork,d	*
(add value)	
visibility	required
Private	*
(add value)	

Figure 2.6: LDAP group example

In Figure 2.6 we can see how a group is stored in LDAP. Groups have six fields, the *CN* (Common Name) which is used both internally and at the web services level; the *groupname* which is the name used at the user level, its this name that is used in the front-end; the *member* field which is where membership is kept, the identifiers are the user DNs; the *objectClass* which defines the entry's type; the *owner* which is the DN of the owner of the group; and at last the *visibility* of the group.

Chapter 3

EM Access Control Consolidation

In this chapter I will discuss the model extensions performed on the EM access control model, the changes on the access control system, and the extensions performed on the access control system.

3.1 Model extensions

One of the limitations of the EMv2 access control model was that in order to share a resource one must first create a group. If the owner wants to share the resource with only one user it's not logical to infer that he must first create a group with that user as member.

In order to overcome this limitation I extended the EMv2 access control model to include the user sharing feature. This extension was made available in the EMv3 AC model.

3.2 Access Control System Changes

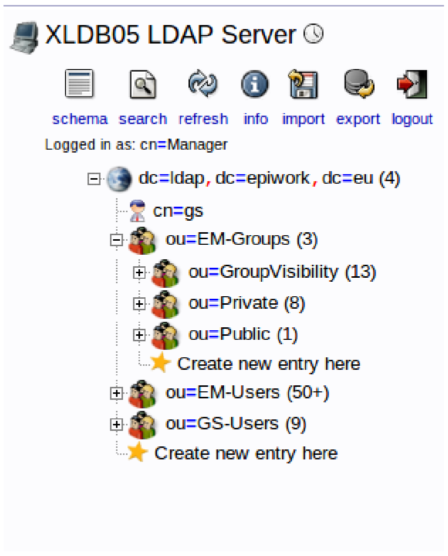
The access control evolution from EMv2 to EMv3 implied many changes, in this section I will present them.

3.2.1 LDAP's Structure

LDAP is a service commonly used for user authentication. It profits from its non-relational architecture, unlike other relational services such as SQL-based, LDAP has an hierarchical organization providing a better performance with large amounts of data. The EM's groups are stored in in the LDAP server.

EM Groups

In the first EM's LDAP architecture the group visibility was an LDAP entry (Figure 3.1).


XLDB05 LDAP Server

Logged in as: cn=Manager

```
dc=ldap, dc=epiwork, dc=eu (4)
  cn=gs
  ou=EM-Groups (3)
    ou=GroupVisibility (13)
    ou=Private (8)
    ou=Public (1)
    Create new entry here
  ou=EM-Users (50+)
  ou=GS-Users (9)
  Create new entry here
```

Figure 3.1: LDAP's old entry structure

This architecture presented some issues regarding the arrangement of the entries. This was due to the fact that the group visibility was an LADP entry. Because the group itself was inside the group visibility entry, one must first know the visibility of the group in order to know its DN. Knowing the DN of a group is the only effective mean to fetch its information.

Another problem caused by the fact that group visibility was an entry was that every time the visibility had to be changed, the group was being deleted and created again in the entry corresponding to the new visibility.

Also, because groups with different visibilities were stored in different entries (according to the visibility of the group) it was possible for two groups to have the same name. If by any reason at a certain point in time they have the same visibility then the behavior will be undefined. Take the following scenario as an example:

Alice creates a group with the name *mygroup* and with *Private* visibility. In the ldap the DN of the group will be *dc=ldap,dc=epiwork,dc=eu, ou=EM-Groups, ou=Private, ou=mygroup*. Later, *Bob* also creates a group named *mygroup* but with *Public* visibility. At this point *Bob* can only see his group, but *Alice* sees two groups named *mygroup* and can't distinguish them. However bad the situation might be it can get worse: now *Alice* decides to make her group *Public*. This means that LDAP will change the group from *dc=ldap,dc=epiwork,dc=eu, ou=EM-Groups, ou=Private* to *dc=ldap,dc=epiwork,dc=eu, ou=EM-Groups, ou=Public*, however there is already a group named *mygroup* in *dc=ldap, dc=epiwork,dc=eu, ou=EM-Groups, ou=Public* (*Bob's* group), and because in LDAP the DN is the identifier of an entry *Alice's* group will replace *Bob's*.

In order to solve these problems I changed the group visibility to become an attribute of a group instead of an entry.

3.2.2 Fedora Policies

Regarding the Fedora digital library, we had the concept of policy resources. These were resources whose only purpose was to hold policies that would apply to the relevant EM's resources. Shown in Figure 3.2 is the Fedora access control implementation, where the resource empid:xxx has *n* policy resources whose purpose is to hold FESLPOLICY datastreams with rules that apply to it.

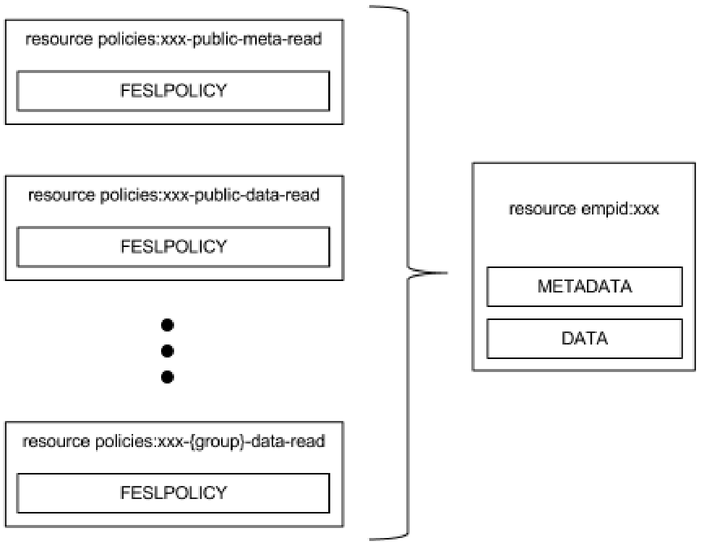

resource policies:xxx-public-meta-read
FESLPOLICY
resource policies:xxx-public-data-read
FESLPOLICY
resource policies:xxx-{group}-data-read
FESLPOLICY
resource empid:xxx
METADATA
DATA

Figure 3.2: EMv2's Access Control Implementation

In this first phase of the EM access control it could take up to thirty seconds to access a resource which was by far unacceptable. However when we turned off the access control layer times decreased to acceptable values. I performed some tests to find the origin of this time difference and traced the problem back to these policy resources.

These policy resources were in most cases in greater number than the actual resources. This happened because each FESLPOLICY datastream had only one XACML rule. This approach raised limitations regarding the scalability of the marketplace.

In EMv3's access control (Figure 3.3), we abandoned the policy resource concept and though about a new and better way to manage access control. The solution was to put all the rules that apply to a resource inside that same resource, in a FESLPOLICY datastream.

The number of objects in the Fedora repository hugely decreased, also it improved performance substantially. A request that previously took about thirty seconds now takes about half a second and most importantly, the response time doesn't increase as the resource gets shared with more and more users/groups.

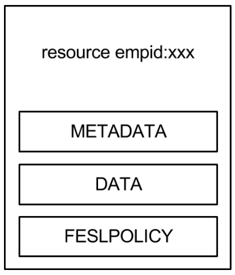


Figure 3.3: EMv3's Access Control Implementation

Rule templates

As I said before, the XACML policies are formed by a set of rules which combined form a policy that is ready to be inserted into a FESLPOLICY datastream. To ease the process of creating a policy, I created five templates of rules.

I use these rule templates to form FESLPOLICY datastreams, for that I replace *%subjects* by a set of subjects separated by a comma; *%actionAb* is replaced by the abbreviate of the action to grant; *%action* is replaced by the action to be granted; *%groupAttributeValues* is replaced by a set of groups in compliance with the XACML format; *%subjectAttributeValues* is replaced by a set of users in the XACML format.

Group data rule template This template is meant for granting to a set of groups and/or users permission to access the data-datastreams of a resource (all datastreams except for those five listed in the example: *EM*, *DC*, *Request*, *RELS-EXT* and *FESLPOLICY*). Because this rule template is aimed at data datastreams, we begin by denying access to the metadata datastreams (*EM*, *DC*, *Request*, *RELS-EXT*, *FESLPOLICY*). These datastreams can only be accessed if meta access is granted, except for the FESLPOLICY which can never be directly accessed. Then we specify the set of actions we are granting {*readds*, *createds*, *updateds*, *deleteds*}. After, there is a function that grants access to a set of groups. At last, there is a function that grants access to a set of users. The group data rule template is shown bellow:

```
<?xml version="1.0" encoding="UTF-8"?>
<Rule Effect="Permit" RuleId="%subjects;data;%actionAb">
  <Condition>
    core of my work
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
       ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         not">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <ResourceAttributeDesignator AttributeId="urn:fedora:
             names:fedora:2.1:resource:datastream:id" DataType=
             "http://www.w3.org/2001/XMLSchema\#string" />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">EM</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">DC</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">Request</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">RELS-EXT</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">FESLPOLICY</AttributeValue>
          </Apply>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:action:id" DataType="http://www.w3.
           org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">%action</Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         or">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="memberOf"
             DataType="http://www.w3.org/2001/XMLSchema\#string
             " />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%groupAttributeValues</Apply>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="urn:oasis:
             names:tc:xacml:1.0:subject:subject-id" DataType="
             http://www.w3.org/2001/XMLSchema\#string" />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%subjectAttributeValues</
             Apply>
        </Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

Group create template This template is meant for granting to a set of groups and/or users permission to create a datastream in a resource. First we define the action which in this case is *createds*. After, there is a function that grants create access to a set of groups. At last there is a function that grants create access to a set of users.

This template is available in the Appendices section (A.1).

Group meta rule template This template is meant for granting to a set of groups and/or users permission to access the metadata datastreams of a resource. Because this rule template is aimed at metadata datastreams, we begin by giving access to a subset of the metadata datastreams (*EM*, *DC*, *Request*, *RELS-EXT*). In this case the *FESLPOLICY* isn't included because we do not allow direct changes or access to this datastream. Then we specify the set of actions we are granting {*readds*, *createds*, *updateds*, *deleteds*}. After, there is a function that grants access to a set of groups. At last, there is a function that grants access to a set of users.

This template is available in the Appendices section (A.2).

Public meta rule template This template is very similar to the group meta rule template. The difference is that because this rule aims to the Public user we do not need to specify any user/group. In the XACML syntax by omitting the subject we are granting permission to everyone.

This template is available in the Appendices section (A.3).

Public data rule template This template is very similar to the group data rule template. The difference, as before, is that because this rule aims to the Public user we do not need to specify any user/group.

This template is available in the Appendices section (A.4).

3.2.3 LDAP indexes

Solr has both resources information and access control information indexed. Regarding the resources, Solr indexes the main fields of the resources. Regarding the access control information, for each resource, Solr indexes the groups that can access it.

When user *Alice* requests access to the resource *resourceA*, the web services query LDAP for all the groups of *Alice*, then they send Solr the groups returned from LDAP, Solr matches them against the groups that have access to *resourceA* and gives the web service a response that defines if access should or shouldn't be authorized. LDAP does this search by running through all the groups membership to see if *Alice* is a member.

When the number of groups is relatively small, the membership search LDAP executes doesn't take too much time to finish. However when the number of groups increases, this becomes a very slow process with a very poor performance with LDAP response times increasing exponentially.

I ran performance tests on LDAP using Jmeter [6]. I ran the tests from my machine which was connected to the LDAP server through a VPN connection, I was connected to the Internet through eduroam (University wireless connection). The objective of the tests was to determine the LDAP throughput when searching for the groups a user belongs to. Based on the results of the tests I was able to determine the source of the problem, which was the lack of attribute indexes.

There are several attributes that LDAP searches for when evaluating membership. By indexing the LDAP attributes that were searched in these queries I solved this performance issue. I indexed the following attributes: “member”, “visibility” and “owner”.

As a result of the indexed fields the response times improved significantly. In Figure 3.4 I present an example of a resource that was shared with a diverse number of groups. The first time the LDAP wasn't using indexes while at the second time it was. While with the aid of LDAP indexes, the response time (y-axis) averages at 40ms. Without it, as the number of groups (x-axis) increases the response time increases exponentially as the trend line shows.

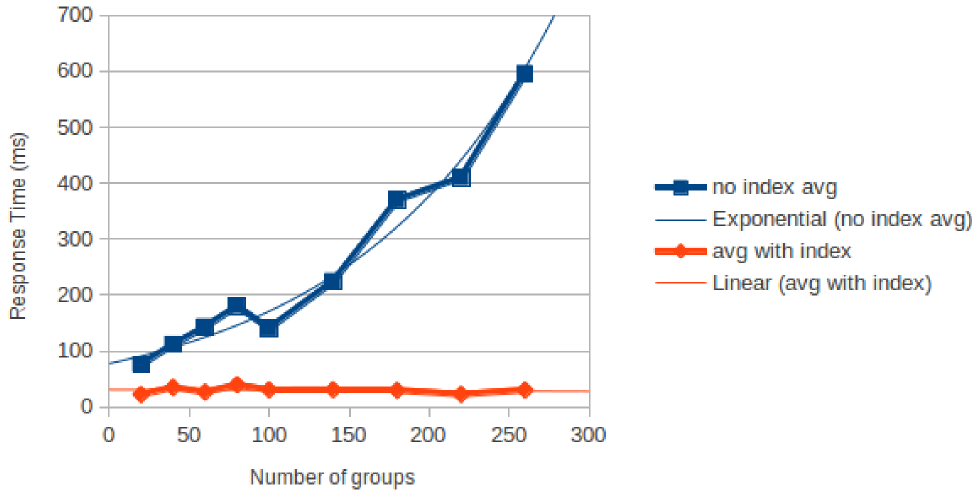


Figure 3.4: LDAP's response time with/without indexes

Because the web services are the main connection between the users/clients that use our platform, I ran tests to measure the group related web services throughput.

I started by creating four different test users in LDAP, later I created two more. The users were named *testUser1*, *testUser2*, *testUser3* and *testUser4* respectively. I assigned a percentage to all of them, 0% for *testUser1*, 25% for *testUser2*, 75% for *testUser3* and 100% for *testUser4*. This percentages represented the number of groups each of the users belonged (in the total number of groups that existed in LDAP).

I started with a total of 20 groups, this way *testUser1* was in 0 groups, *testUser2* in 5, *testUser3* in 15 and *testUser4* in all of the 20. The results are shown in Figure 3.5.

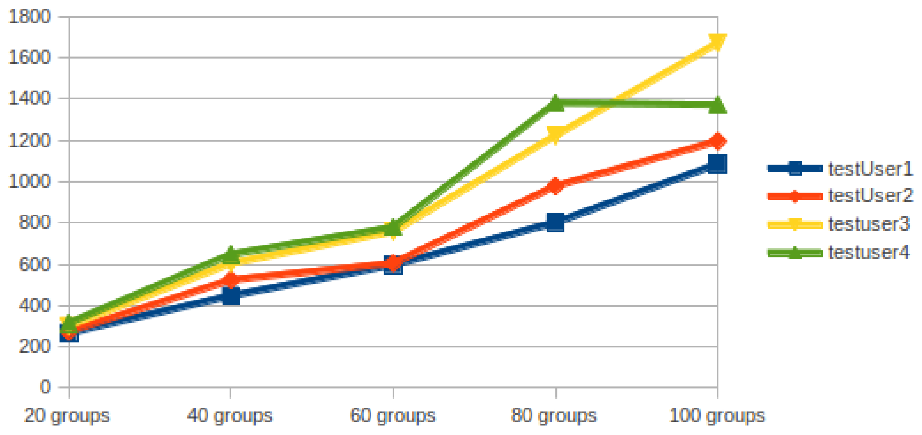


Figure 3.5: Web Services response time without LDAP indexes

In Figure 3.6 we can see the same requests that was made in the first case study but now with the aid of the LDAP indexed fields. We can see major improvements. For the users that don't belong to a big number of groups, the response time decreased significantly.

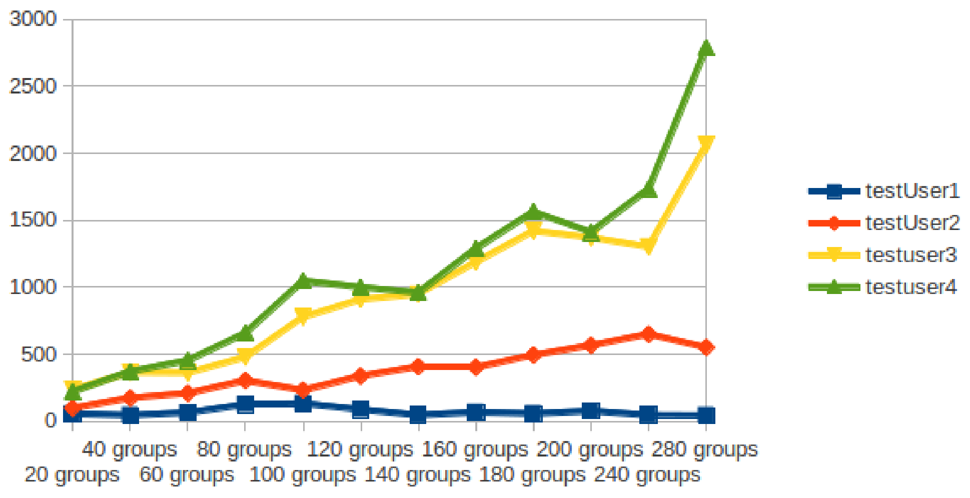


Figure 3.6: Web Services response time with LDAP indexes

On the other hand, the response time for the users that are on a relatively large number of groups still grows as the number of groups increases. This happens because even though the user's membership is indexed, all those groups must be fetch by LDAP and returned to the web services. In these cases indexes do not improve performance.

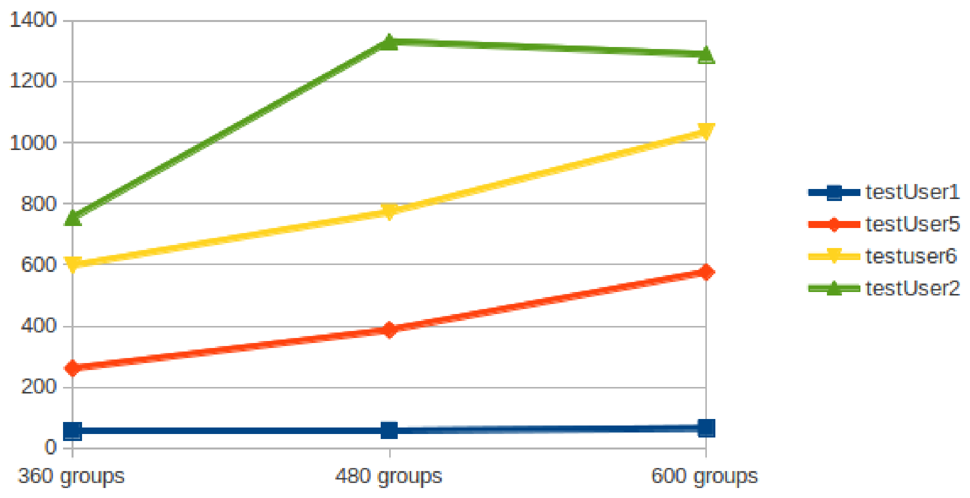


Figure 3.7: Web Services with indexes response time (with *testUser5* and *testUser6*)

In a more realistic approach, we will find that users such as *testUser1*, *testUser2*, *testUser5* and *testUser6* are more likely to exist (because they belong to fewer groups) rather than users such as *testUser3* and *testUser4* that belong to a large number of groups. The following graphic (Figure 3.7) shows the response times for users that belong to a relatively small amount of groups when compared to the complete universe of groups. This test shows that with this users we can achieve acceptable response times while dealing with a big number of groups.

3.2.4 Web Services normalization

The web services are publicly available on the internet and can be used by any HTTP client. They operate over the LDAP server and are directly dependent on it's performance.

In the EMv2, the web-services weren't normalized and each had its own input/output format. Some returned XML, others returned HTTP. Even when the return type was the same the output was structured differently.

I began by changing the web-services output so that all of them returned XML. Later I added the possibility for the user to choose the output format. Then I normalized the internal output by returning a triple in each of them: **status**, **result** and **message**. The **status** could have two values: 0 for *OK* and 1 for *NOK*(*Not-OK*); the *result* was the result of that specific function; finally, the *message* was a message that was related with the *status* (mostly used when there were errors with the functions). This way I manage to simplify the relation between the front-end and the web-services, for instance how the error messages were processed.

3.3 Access Control System Extensions

It is only logical to discuss access control when there are resources on which access control can be applied. In order to get resources into the EM's repository we must first provide for the means to upload them. Because of this it is very important to foster the upload of resources, for this it is very important that the upload web services be efficient.

3.3.1 Upload

In the second release of the EM, EMv2, the upload web service took an average of 15 second to complete. This response time was far beyond acceptable. After discarding the front-end as the source of this problem I analyzed the upload web service.

I noticed that all resources were being indexed in Solr in a synchronized call. So I changed the web service and moved this indexation to a separate thread. This way, the user would get the upload response faster even though the resource was still being indexed. After this change, the upload went from the average 15 seconds to an average of 10 seconds.

After further analysis I noticed as well that the process of validation of the *em:type* and *em:subject* fields was calling the rawfetch web service twice to fetch the datastreams where all the allowed types and subjects were stored. To avoid this, I made a cache mechanism which fastened the web service significantly. With this cache mechanism the results were stored in cache which reduced the calls to the rawfetch web-service in the upload process.

Another issue that was slowing this web service was the creation of the owner policies. I did the same as I did with the indexation of the resource in Solr, I moved the creation of the policies to a different thread.

After all this changes the web service went from an average of 15 seconds to an average of 1 second.

3.3.2 Binary Upload

This web service enables users / applications to update or add new content to a resource in the Repository. To execute this, the resource must already exist. In the EMv2 this web-service didn't existed and there were no means for a user to upload any datastreams.

In response to this limitation I developed the binary upload web-service which allowed for a user / application to upload a binary file regardless of its format / content. This feature was crucial for the integration with the Gleamviz team because their simulations needed to be uploaded to the EM.

3.3.3 Group Manager

In a first phase, all the group related data was sent to the web services through HTTP GET. This presented not only a security issue (e.g.; users changing administrator groups) but also a huge inconvenient regarding the page indexing by search engines (e.g.: Google indexes a page that creates a group).

I solved this problem by changing the web services HTTP methods from GET to POST and by passing the sensitive information in variables instead of using the URL. This solved both the security and the indexation related problems as well.

In order to create a group creation interface that was intuitive I had to integrate the drupal-based front-end with the group manager web services. Also I had to do this in a simple way with the few amount of steps possible.

To do this I used ”state of the art” auto-complete boxes in the front-end that searched LDAP for EM users as the user typed the letters in the text box. For this to work I used Ajax. In the group manager module I created a web-service that received a string and return all the users and groups whose names matched partially or totally with that string. Then a list of possible matches was presented to the user and he only had to chose the user / group he wanted to share its resource with and click the share button. This automatically gives that user / group read access to that resource’s data and metadata.

3.3.4 Access Control

As said before, the access control model in the EM went through many changes. At first EM used a centralized GBAC model; in a second phase we evolved it into a decentralized GBAC model; then we added the possibility of sharing resources with single users (without the need of the creation of a group); at last and in order to enhance EM’s expressiveness we added a dynamic groups that allowed users to share their resources based on rules, also we allow users to use Social Network connections to share their resources.

These changes in the Access Control model was reflected in the Access Control module in the web-services. Beginning in the EMv2 Access Control system, I made some changes to the *share with group* feature; I implemented the *public resource* feature (allow a resource to be seen by every visitor of the EM website); then I added the *share with user* feature; then I implemented the *share with dynamic group* feature; at last I implemented the *share with social group*.

Chapter 4

Dynamic Groups

Contrary to static groups, dynamic groups members aren't defined statically. Instead, its members are defined by one or more rules. When the members of such group are solicited, an interpreter runs the rule(s) and the result is the membership of that group.

4.1 Model

The traditional dynamic group model is one in which the group membership is defined by a rule or a set of rules instead of a set of users. In Figure 4.1 we can see the UML representation of a Dynamic Group where a user belongs to n groups, a group can have n users and the users that belong to a group are identified by one or more rules.

pic/concepts/dynamic_group_uml.png

Figure 4.1: Dynamic Group UML

4.2 Dynamic Groups with LDAP attributes

Regarding the LDAP server, the membership of a dynamic group is defined by an LDAP search[1]. The dynamic group uses the structural objectclass **groupOfURLs** (or auxiliary objectclass **ibm-dynamicGroup**) and the attribute, **memberURL** to define the search using a simplified LDAP URL syntax. The syntax of the rule is as follows:

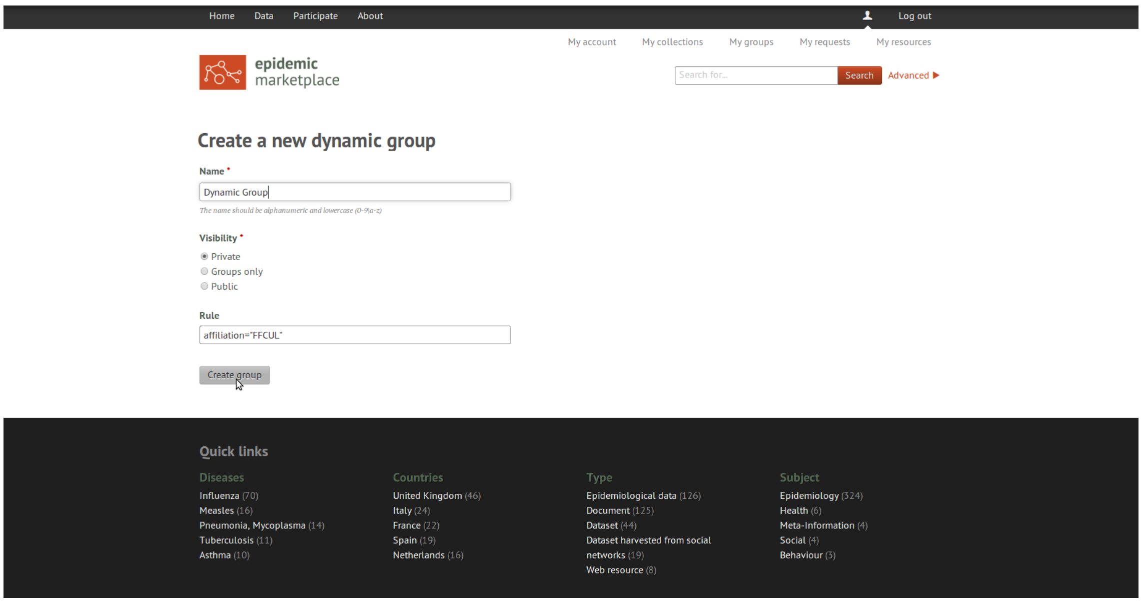
ldap:*///<base DN of search> ? ? <scope of search> ? <searchfilter>*

To further enhance the expressiveness of the access control model, users may also create dynamically defined groups. Instead of providing a static set of users, **group owners can specify a rule that dynamically defines the group**. The **memberURL** value of an example of one EM dynamic group is shown bellow:

ldap:///ou=EM-Groups??base?affiliation=FFCUL

4.2.1 Interface

In the figure below I present the interface users use to create dynamic groups. The name and visibility attributes of the group are the same as in a normal group but instead of defining individual users/groups, rules are used to define the group membership.



Create a new dynamic group

Name *

Dynamic Group

The name should be alphanumeric and lowercase (0-9|a-z)

Visibility *

● Private
○ Groups only
○ Public

Rule

affiliation="FFCUL"

Quick links

Diseases	Countries	Type	Subject
Influenza (70)	United Kingdom (46)	Epidemiological data (126)	Epidemiology (324)
Measles (16)	Italy (24)	Document (125)	Health (6)
Pneumonia, Mycoplasma (14)	France (22)	Dataset (44)	Meta-Information (4)
Tuberculosis (11)	Spain (19)	Dataset harvested from social networks (19)	Social (4)
Asthma (10)	Netherlands (16)	Web resource (8)	Behaviour (3)

Figure 4.2: Creating a dynamic group

4.2.2 Web Services

To cope with the Dynamic Groups several web services were made. The most important are the ones that follow:

a) **https://api.epimarketplace.net/groups/create/dynamic**
Request Example:
Create a xml file containing the following data:

```
<?xml version="1.0" encoding="UTF-8"?>
<createGroup>
  <groupType>dynamic</groupType>
  <name>FCCUL group</name>
  <visibilityType>Private</visibilityType>
  <rules>
    <rule>
      <attribute>affiliation</attribute>
      <logicoperand>=</logicoperand>
      <rulestring>FFCUL</rulestring>
    </rule>
  </rules>
</createGroup>
```

Then, to create the group, you just need to call the web service and send the metadata. In this example we will use cURL:
curl -v -H ”Accept: application/xml” -X POST -d ”@request.xml” -u login:password https://api.epimarketplace.net/groups/create/dynamic

b) **https://api.epimarketplace.net/groups/[group_id]/addrule**
Request Example:
Create a xml file containing the following data:

```
<?xml version="1.0" encoding="UTF-8"?>
<rule>
  <attribute>affiliation</attribute>
  <logicoperand>=</logicoperand>
  <rulestring>FFCUL</rulestring>
</rule>
```

Then, to create the group, you just need to call the web service and send the metadata. In this example we will use cURL:
curl -v -H ”Accept: application/xml” -X POST -d ”@request.xml” -u login:password https://api.epimarketplace.net/groups/[group_id]/addrule

c) **https://api.epimarketplace.net/groups/[group_id]/remrule**
Request Example:

Create a xml file containing the following data:

```
<?xml version="1.0" encoding="UTF-8"?>
<rule>
  <attribute>affiliation</attribute>
  <logicoperand>!</logicoperand>
  <rulestring>FFCUL</rulestring>
</rule>
```

Then, to create the group, you just need to call the web service and send the metadata. In this example we will use cURL:
curl -v -H "Accept: application/xml" -X POST -d "@request.xml" -u login:password https://api.epimarketplace.net/groups/[group_id]/remrule

4.3 Dynamic Groups with Social Networks' attributes

The EM provides means to use **social information for the specification of access restrictions**. **Open Social** provides the mechanisms for this integration.

Open Social [8] is a public specification that defines a component hosting environment (container) and a set of common application programming interfaces (APIs) for web-based applications. It was in developed 2007 to integrate Web2.0 network applications so that developers could build collaborative network environments easily.

This enables **epidemiologists** to easily **share resources with their professional connections** (e.g. Linkedin connections) without the requirement of re-introducing collaborators information in the EM website.

Social Groups are suggested to EM users when sharing their resources. The EM asks the Social Network for the connected users and matches them with locally registered users.

An Open Social and XACML based group authorization framework is an effective mean to provide a fine-grained user controlled resource access control mechanism[13].

4.3.1 Subscription

In order to have access to a resource through a "social share" a user must subscribe to this service. This subscription can be done in two separate ways:

1. in the register form by checking the "I give the EM permission to fetch profile data in <Social Network>" checkbox

2. at any time by editing its privacy settings and checking the same check box

This subscription allows the EM to fetch the user's Social identification. I also considered another alternative where subscription wasn't needed to use share and access EM resources through Social Sharing. In this alternative instead of the Social identification, the EM uses emails to identify users.

Take the following scenario as an example: *Bob* is a connection of *Alice* (this relation is bi-directional and mutual) in Social Network *SN*, both *Bob* and *Alice* are users of the EM. *Bob* shares the resource *resourceA* with its *SN* connections (this means he grants the EM privileges to access its *SN* information); *Alice* then tries to access *resourceA*, so the EM queries *SN* for *Bob's* connections, fetches their emails queries LDAP for *Alice's* email and checks for a match.

The email approach was discarded because nowadays it became common for a person to have multiple emails for multiple purposes, and the probability of a person registering different emails in different web sites is high. So, if the email *Alice* provided to *SN* is different from the one she provided to the EM then the social share is inefficient.

With the id approach it doesn't matter if *Alice* registered a different emails in the EM and in the *SN*. This way we will use the *Alice*'s id in *SN* which unique. The drawback of this approach is that prior to be able to access Socially shared resources *Alice* has to give the EM permission to access her Social data in *SN*.

4.3.2 Work-flows

Sharing with a Social Group

In figure 4.3 I present the process of sharing a resource with a social group.

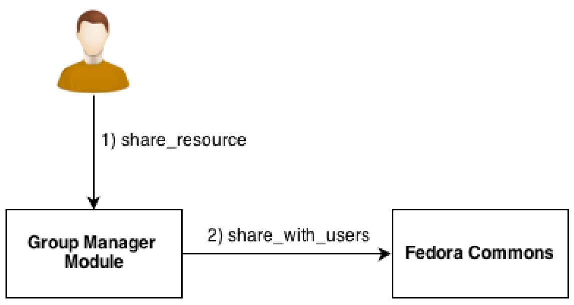


Figure 4.3: Social Group Creation

The **Social Group sharing** works as follows:

1. The user goes to the share-resource interface, types a string / substring that corresponds to the name of a social network supported by the EM and saves the changes.

2. The EM prompts the user to grant access for it to access the user's information in that Social Network, the user accepts the conditions.

3. The web services communicate with the Fedora Commons repository and save the new access changes in the FESLPOLICY datastream of the resource.

Accessing a Social Group

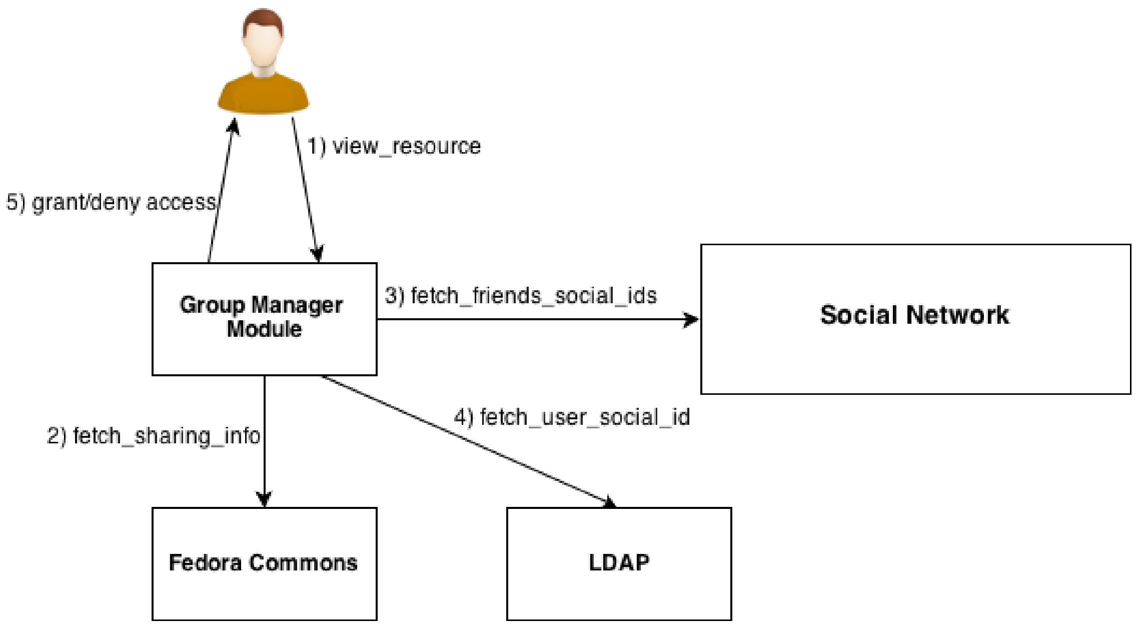


Figure 4.4: Social Group Work-flow

The **Social Group access** works as follows:

1. User *Alice* tries to access a resource *resource1*.

2. The interface communicates with the web services and the web services query the Fedora Commons server for the policies of resource *resource1*.

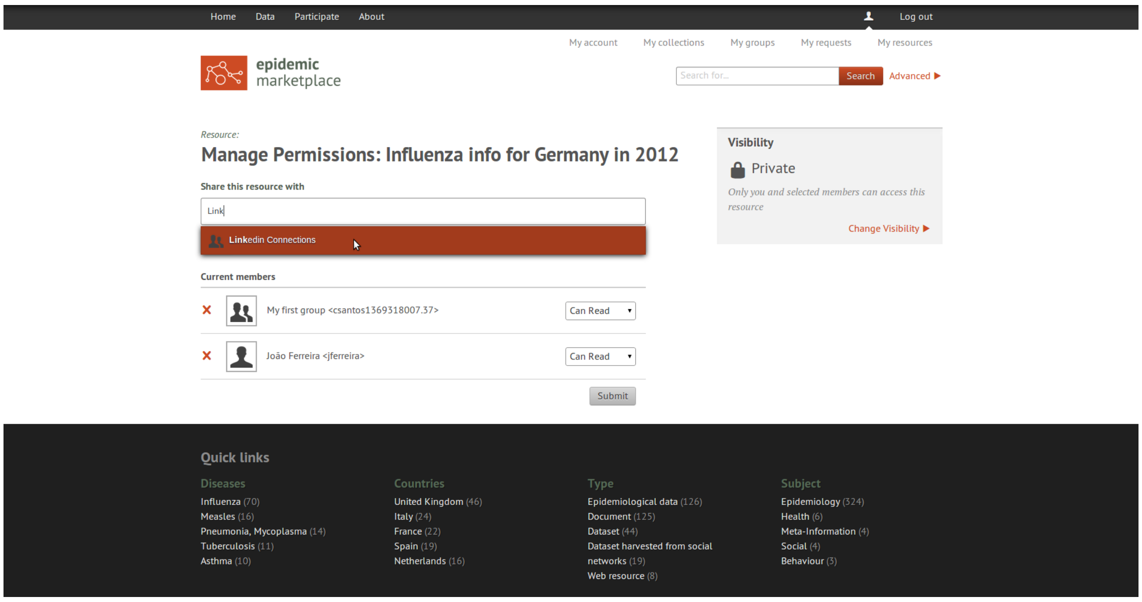
3. If the resource is shared with a social group then the web services will query the Social Network API for the friends of the owner of the resource in that social network. This operation will return their identifications.

4. The web services queries the LDAP server for the list of users that correspond to the identifications retrieved by the WS.

5. If the identification of user *Alice* is in the identification list returned by LDAP then access to resource *resource1* is granted. Access to resource *resource1* is denied otherwise.

4.3.3 Interface

In the figure below we can observe how the users can share a resource with a social group, with his Linkedin connections in this specific case.



Resource:

Manage Permissions: Influenza info for Germany in 2012

Share this resource with

Link

Linkedin Connections

Current members

My first group <csantos1369318007.37>	Can Read
João Ferreira <jferreira>	Can Read

Visibility

Private

Only you and selected members can access this resource

Quick links

Diseases	Countries	Type	Subject
Influenza (70)	United Kingdom (46)	Epidemiological data (126)	Epidemiology (324)
Measles (16)	Italy (24)	Document (125)	Health (6)
Pneumonia, Mycoplasma (14)	France (22)	Dataset (44)	Meta-Information (4)
Tuberculosis (11)	Spain (19)	Dataset harvested from social networks (19)	Social (4)
Asthma (10)	Netherlands (16)	Web resource (8)	Behaviour (3)

Figure 4.5: Sharing a resource with a social group

Chapter 5

Conclusion

In this dissertation I presented the final specification of the EM's access control model, resulting from the 9 months I was in the EPIWORK project. I have fulfilled the goal of enhancing the EM's access control model expressiveness and solved the EM's performance issues.

Throughout the development of this dissertation I evaluated the access control system which was implemented in the EM and enhanced it in order to promote the information sharing while protecting it. I implemented a decentralized GBAC, then I added dynamic sharing competences to the EM's access control model by allowing the users to create dynamic groups based in LDAP searches. Furthermore I extended the model to use social network connections which strongly enhanced the EM's expressiveness.

I also changed the organization of the policies in the Fedora repository. This change was performed due to the access control performance issues in the previous EM access control model.

To further facilitate the collaborative behavior we provided at the web services layer the separation of metadata from data allowing for a resourced to be searchable while protecting its data which can be shared with a more restricted group of users.

The EM began by having a GBAC model, I added the possibility of sharing resources with dynamic groups based on LDAP searches and with dynamic groups based on Social Networks connections. Also, now resources can be shared with single users. All these changes combined raised the EM's access control system to a new level where resources' privacy is ensured while at the same time their sharing is fostered by a multiple set of simple and intuitive sharing options.

In its current access control version, it is not possible to create collections of collections. This feature would enhance the expressiveness of the access control system of the EM because as resources inherit collections permissions, other child collections would inherit the permissions of the parent collections, however this feature was not implemented. Another feature that could be made in order to attract more users into the EM is the possibility of trading resources. In this scenario a user would give another user access to some resource only if the other user gave him back access to one resource of his own. Once both users reached a consensus, they would grant each other access to their resources.

Also, there is still the need for evaluating the performance of LDAP with the dynamic groups scenario and to increase the number of Social Networks currently supported by the EM Social Network groups, currently only LinkedIn is supported. Another feature that could foster the share of information in the platform is the ability of users to give admin permission over their resources to other users, this way other users would also be able to share that resource.

Appendix A

Rule templates

A.1 Group create template

```
<Rule Effect="Permit" RuleId="%subjects;data;c">
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
       ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:action:id" DataType="http://www.w3.
           org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">createds</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         or">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="memberOf"
             DataType="http://www.w3.org/2001/XMLSchema\#string
             " />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%groupAttributeValues</Apply>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="urn:oasis:
             names:tc:xacml:1.0:subject:subject-id" DataType="
             http://www.w3.org/2001/XMLSchema\#string" />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%subjectAttributeValues</
             Apply>
        </Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

A.2 Group meta rule template

```
<Rule Effect="Permit" RuleId="%subjects;meta;%actionAb">
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
       ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ResourceAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:resource:datastream:id" DataType="
           http://www.w3.org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">EM</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">DC</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">Request</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">RELS-EXT</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:action:id" DataType="http://www.w3.
           org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">%action</Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         or">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="memberOf"
             DataType="http://www.w3.org/2001/XMLSchema\#string
             " />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%groupAttributeValues</Apply>
        </Apply>
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <SubjectAttributeDesignator AttributeId="urn:oasis:
             names:tc:xacml:1.0:subject:subject-id" DataType="
             http://www.w3.org/2001/XMLSchema\#string" />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">%subjectAttributeValues</
             Apply>
        </Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

A.3 Public meta rule template

```
<Rule Effect="Permit" RuleId="public@Public;meta;%actions">
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
       ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ResourceAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:resource:datastream:id" DataType="
           http://www.w3.org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">EM</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">DC</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">Request</AttributeValue>
          <AttributeValue DataType="http://www.w3.org/2001/
             XMLSchema\#string">RELS-EXT</AttributeValue>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:action:id" DataType="http://www.w3.
           org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">%actionValues</Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

A.4 Public data rule template

```
<Rule Effect="Permit" RuleId="public@Public;data;%actions">
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and
       ">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         not">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-at-least-one-member-of">
          <ResourceAttributeDesignator AttributeId="urn:fedora:
             names:fedora:2.1:resource:datastream:id" DataType=
             "http://www.w3.org/2001/XMLSchema\#string" />
          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
             function:string-bag">
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">EM</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">DC</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">Request</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">RELS-EXT</AttributeValue>
            <AttributeValue DataType="http://www.w3.org/2001/
               XMLSchema\#string">FESLPOLICY</AttributeValue>
          </Apply>
        </Apply>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:
         string-at-least-one-member-of">
        <ActionAttributeDesignator AttributeId="urn:fedora:
           names:fedora:2.1:action:id" DataType="http://www.w3.
           org/2001/XMLSchema\#string" />
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:
           function:string-bag">%actionValues</Apply>
      </Apply>
    </Apply>
  </Condition>
</Rule>
```

Bibliography

[1] Dynamic groups. http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp?topic=%2Frzahy%2Frzahydynamicgp.htm. Accessed August, 2013.

[2] Fedora commons. http://www.fedora-commons.org/about. Accessed September, 2013.

[3] Gleamviz. http://www.gleamviz.org/. Accessed September, 2013.

[4] Group based access control. http://www.mediawiki.org/w/index.php?title=Extension:Group_Based_Access_Control&oldid=514587. Accessed September, 2013.

[5] Jaas. http://en.wikipedia.org/wiki/Java_Authentication_and_Authorization_Service. Accessed September, 2013.

[6] Jmeter. http://jmeter.apache.org. Accessed September, 2013.

[7] Ldap rule-bac. http://infolib.lotus.com/resources/portal/8.0.0/doc/nl_NL/PT800ACD002/admin/rbug.html. Accessed September, 2013.

[8] Open social. http://opensocial.org/. Accessed August, 2013.

[9] Solr. http://lucene.apache.org/solr/. Accessed September, 2013.

[10] S. Gavrila D. Kuhn D. Ferraiolo, R. Sandhu and R. Chandramouli. ”proposed nist standard for role-based access control,”. In *ACM Transactions on Information and System Security (TISSEC)*, volume 4, no. 3, pages 224 – 274, 2001.

[11] T. Moses et al. ”extensible access control markup language (xacml) version 2.0,”. In *Oasis Standard*, volume 02, 2005.

[12] C. Francisco S. Mário F. João, P. Cátia. ”bringing epidemiology into the semantic web. international conference on biomedical ontologies (icbo)”. 2012.

[13] Z. Li H. Zhang and W. Wu. ”open social and xacml based group authorization framework”. 2012.

[14] P. Benoit J. M. Bradshaw, S. Dutfield and J. D. Woolley. ”kaos: Toward an industrial-strength. open agent architecture”. pages 375 – 418, 1997.

[15] J.B.D Joshi. ”access-control language for multidomain environments. internet computing, ieee”. volume 8, pages 40 – 50, 2004.

[16] L. Kagal. Rei ontology specifications, ver 2.0. http://www.cs.umbc.edu/~lkagal1/rei/. Accessed December, 2012.

[17] J. Moffett. ”specification of management policies and discretionary access control”. In *Network and distributed systems management*, page chapter17, 1994.

[18] E. Lupu N. Damianou, N. Dulay and M. Sloman. ”the ponder policy specification language”. 2001.

[19] J. Sermersheim. ”rfc4511: Lightweight directory access protocol (ldap): The protocol”. 2006.

[20] D. Tcsec. ”trusted computer system evaluation criteria,” technical report 5200.28-std. In *US Department of Defense, Tech. Rep.*, 1985.