OACore

Adrian ESTIBEN Sanchez Hurtado



GRADO EN INGENIERIA DE SOFTWARE

TRABAJO DE FIN DE GRADO

Facultad de Informatica

5 de Junio de 2019

Director:

Colaborador:

# Autorización de difusión y utilización

Adrian Estiben Sanchez Hurtado

5 de Julio de 2019

# Dedicatoria

*A M.P.G., por todas sus enseñanzas*

# Agradecimientos

Gracias a los miembros del Instituto de Ciencias de la Computación e Informática de la Universidad de Radboud, especialmente a Olha van Keln y Kerst Shkav por su colaboración en mi visita a Nimega y por ceder el código de sus trabajos originales bajo Licencia Creative Common CC BY-NC.

Y un agradecimiento especial a la empresa Laptop S.A. por financiarme la estancia en dicha universidad y permitirme comprobar la utilidad de mi aplicación en sus instalaciones.

# Prologo

# Indice

# Resumen

# Palabras Clave

En castellano e ingles

# Introduccion

una introducción con los antecedentes, objetivos y plan de trabajo,

# Resultados y discusión crítica y razonada de los mismos, con sus conclusiones,

## Keycloak

Architecture

Keycloak AuthZ Architecture Overview

From a design perspective, is based on a well-defined set of authorization patterns providing these capabilities:

* **Policy Administration Point (PAP)**

Provides a set of UIs based on the Keycloak Administration Console to manage resource servers, resources, scopes, permissions, and policies. Part of this is also accomplished remotely through the use of the [Protection API](https://www.keycloak.org/docs/3.1/authorization_services/topics/service/protection/protection-api.html#_service_protection_api).

* **Policy Decision Point (PDP)**

Provides a distributable policy decision point to where authorization requests are sent and policies are evaluated accordingly with the permissions being requested. Part of this is also accomplished remotely through the use of the [Authorization](https://www.keycloak.org/docs/3.1/authorization_services/topics/service/authorization/authorization-api.html#_service_authorization_api) and [Entitlement](https://www.keycloak.org/docs/3.1/authorization_services/topics/service/entitlement/entitlement-api.html#_service_entitlement_api) APIs.

* **Policy Enforcement Point (PEP)**

Provides implementations for different environments to actually enforce authorization decisions at the resource server side. Keycloak provides some built-in [Policy Enforcers](https://www.keycloak.org/docs/3.1/authorization_services/topics/enforcer/overview.html#_enforcer_overview).

* **Policy Information Point (PIP)**

Being based on Keycloak Authentication Server, you can obtain attributes from identities and runtime environment during the evaluation of authorization policies.

The authorization process

Three main processes define the necessary steps to understand how to use Keycloak to enable fine-grained authorization to your applications:

* **Resource Management**
* **Permission and Policy Management**
* **Policy Enforcement**

**Resource Management**

**Resource Management** involves all the necessary steps to define what is being protected.

Resource Management Overview

First, you need to specify Keycloak what are you looking to protect, which usually represents a web application or a set of one or more services. For more information on resource servers see [Terminology](https://www.keycloak.org/docs/3.1/authorization_services/topics/overview/terminology.html#_overview_terminology).

Resource servers are managed using the Keycloak Administration Console. There you can enable any registered client application as a resource server and start managing the resources and scopes you want to protect.

Resource Server Overview

A resource can be a web page, a RESTFul resource, a file in your file system, an EJB, and so on. They can represent a group of resources (just like a Class in Java) or they can represent a single and specific resource.

For instance, you might have a *Bank Account* resource that represents all banking accounts and use it to define the authorization policies that are common to all banking accounts. However, you might want to define specific policies for *Alice Account* (a resource instance that belongs to a customer), where only the owner is allowed to access some information or perform an operation.

Resources can be managed using the Keycloak Administration Console or the [Protection API](https://www.keycloak.org/docs/3.1/authorization_services/topics/service/protection/protection-api.html#_service_protection_api). In the latter case, resource servers are able to manage their resources remotely.

Scopes usually represent the actions that can be performed on a resource, but they are not limited to that. You can also use scopes to represent one or more attributes within a resource.

**Permission and Policy Management**

Once you have defined your resource server and all the resources you want to protect, you must set up permissions and policies.

This process involves all the necessary steps to actually define the security and access requirements that govern your resources.

Permission and Policy Management Overview

Policies define the conditions that must be satisfied to access or perform operations on something (resource or scope), but they are not tied to what they are protecting. They are generic and can be reused to build permissions or even more complex policies.

For instance,to allow access to a group of resources only for users granted with a role "User Premium,"" you can use RBAC (Role-based Access Control).

Keycloak provides a few built-in policy types (and their respective policy providers) covering the most common access control mechanisms. You can even create policies based on rules written using JavaScript or JBoss Drools.

Once you have your policies defined, you can start defining your permissions. Permissions are coupled with the resource they are protecting. Here you specify what you want to protect (resource or scope) and the policies that must be satisfied to grant or deny permission.

**Policy Enforcement**

**Policy Enforcement** involves the necessary steps to actually enforce authorization decisions to a resource server. This is achieved by enabling a **Policy Enforcement Point** or PEP at the resource server that is capable of communicating with the authorization server, ask for authorization data and control access to protected resources based on the decisions and permissions returned by the server.

PEP Overview

Keycloak provides some built-in [Policy Enforcers](https://www.keycloak.org/docs/3.1/authorization_services/topics/enforcer/overview.html#_enforcer_overview) implementations that you can use to protect your applications depending on the platform they are running on.

**Authorization Services**

Authorization services consist of the following RESTFul APIs:

* **Protection API**
* **Authorization API**
* **Entitlement API**

Each of these services provides a specific API covering the different steps involved in the authorization process.

# Bibliografia

<https://www.keycloak.org/docs/3.1/authorization_services/topics/overview/architecture.html>

# Terminology

## Keycloak

**Resource Server**

Per OAuth2 terminology, a resource server is the server hosting the protected resources and capable of accepting and responding to protected resource requests.

Resource servers usually rely on some kind of information to decide whether access to a protected resource should be granted. For RESTful-based resource servers, that information is usually carried in a security token, typically sent as a bearer token along with every request to the server. Web applications that rely on a session to authenticate users usually store that information in the user’s session and retrieve it from there for each request.

In Keycloak, any **confidential** client application can act as a resource server. This client’s resources and their respective scopes are protected and governed by a set of authorization policies.

1. La memoria incluirá una portada normalizada con la siguiente información: título,

autores, profesor director, codirector si es el caso, curso académico e identificación de

la asignatura (Trabajo de fin de grado del Grado en -nombre del grado

correspondiente-, Facultad de Informática, Universidad Complutense de Madrid). Los

datos referentes al título y director (y codirector en su caso) deben corresponder a los

publicados en la lista indicada en los puntos 8 y 9 de la sección III de esta normativa.

2. La memoria debe incluir la descripción detallada de la propuesta hardware/software

realizada y ha de contener:

a. un índice,

b. un resumen y una lista de no más de 10 palabras clave para su búsqueda

bibliográfica, ambos en castellano e inglés,

c. una introducción con los antecedentes, objetivos y plan de trabajo,

d. resultados y discusión crítica y razonada de los mismos, con sus conclusiones,

e. bibliografía.

3. La memoria constará de un mínimo de 25 páginas para los proyectos realizados por un único alumno, y de al menos 5 páginas más por cada integrante adicional del grupo. En

este número de páginas solo se tiene en cuenta el contenido correspondiente a los

apartados c y d del punto anterior.

4. La memoria puede estar escrita en castellano o inglés, y en cualquier caso al menos la

introducción y las conclusiones deben aparecer en ambos idiomas.

6. El grupo debe entregar a su director un borrador de la memoria antes de la fecha

límite prevista en el calendario de TFG. El borrador se ajustará a las mismas normas en

cuanto a extensión y organización que la memoria final del proyecto.

7. Los miembros del grupo considerados como aptos para presentación pública deberán

entregar a la comisión de grado y en fechas determinadas por el calendario de TFG:

a. la memoria definitiva y los materiales adicionales que se consideren necesarios

para su evaluación,

b. la autorización de difusión firmada por los miembros del grupo considerados

aptos y por el director, disponible en http://informatica.ucm.es/TFG-grados.

Cada curso la comisión de grado aprobará y publicará el procedimiento de entrega de

estos materiales.

8. La comisión se encargará de enviarla a los miembros de los tribunales para su

valoración.

9. La comisión de grado proporcionará al tribunal de matrícula de honor las memorias y

los materiales adicionales de los trabajos que opten a matrícula de honor.

10. Las memorias y los materiales complementarios se depositarán en la Biblioteca, que

garantizará su conservación y difusión, si procede. En casos que requieran especial

protección, algunos trabajos podrán ser suprimidos del catálogo y excluidos de

cualquier tipo de préstamo o acceso.

Bibliography