Fine-Grained Authorization with Keycloak SSO

Marek Posolda
Main features

- Authentication for web (and other) applications
- SSO - login once to all the applications
- Single-sign out
- Standards based (OpenID Connect, SAML2, OAuth2, UMA)
Identity management

- Realms, clients, roles, users, authorization data
- Storing data in RDBMS
UI

- Admin console
- Account management
- User login forms, registrations, reset passwords...
More cool features...

- Social brokering support (Facebook, Google, Twitter...)
- Any 3rd party identity provider brokering (OpenID Connect, SAML2, OAuth2)
- 3rd party user storage (LDAP, SSSD, custom...)
- Custom authentication mechanisms (TOTP, Kerberos). Possibility to add more
- Themes - enhance Keycloak forms
- Events - monitoring (User events, admin events)
Securing your application

- Keycloak adapters (OIDC, SAML)
- Almost no need to code anything in your app!
Adapters

- Client-side JavaScript
  - With Cordova support for mobile
- WildFly / Jetty / Tomcat
- Fuse
- Node.js
- Servlet filter
- Spring Boot / Spring Security
Monolithic application

1. Open application
2. Request authentication
3. Authorization code
4. ID Token

Browser

User

Application

Authenticate
Service

Verify offline with signature

1. Request with access token
2. Retrieve public key
3. Verify token

Verify online

1. Request with access token
2. Verify token
Authorization - requirements

- Decouple authorization from services
- Flexible and dynamic access control model
- Standards-based (OIDC, UMA)
- Privacy By Design
- Performance
- Resource-Based
Common AuthZ approach

- People Managers can access information about any Employee
- Hard to maintain when requirements changes
- Not aligned with the business model

```java
if (user.hasRole("people-manager")) {
  // grant access to employee data
}
```
The proposal

- Focus on the business
- The resource we want to access and action we want to do

```java
if (user.canAccess("employee:salary")) {
    // grant access to employee data
}
```

```java
if (user.canChange("john-doe:salary")) {
    // grant access to employee data
}
```
Authz model

- X can do Y on Z
- X is user identity
- Y is action (Scope)
- Z is resource
- Policy - the actual implementation of authorization rule, which can decide
- Permission - binding between Resource (and eventually Scope) and Policy
- Evaluation - more policies can be subject decision. Configurable whether all need to grant or just single one etc.
Overall Architecture
Configuration

- Policy enforcer on service/adapter side
- Policies, resources, scopes, permission on Keycloak side
User-Managed Access

- An extension to OAuth2 that enhance OAuth capabilities in order to provide:
  - Party-to-party authorization
  - Asynchronous authorization
  - Resource owners have control over their resources and how they are accessed
Use-case

- jdoe wants to access alice’s pictures
- jdoe request access to alice’s pictures
- alice is notified about the request and approves it
- jdoe can now access alice’s pictures
- Anytime alice can revoke access to her pictures
RPT token

- Access token with “permissions”
- Used by frontend application to send requests to services protected by policy enforcer
- Service policy enforcer is able to just verify permissions inside the RPT token. No need to send request to Keycloak
UMA flow - tickets

- Frontend application wants to access protected resource inside service
- Service validates RPT. If contains permissions for requested resource -> ACK
- Otherwise service sends UMA ticket with requested permissions
- Frontend application sends UMA ticket to Keycloak together with existing RPT (incrementing authorization)
- Keycloak returns either:
  - new RPT to frontend client containing all permissions
  - Message like “request_submitted” informing user that resource owner needs to approve (async authorization)
- Frontend application access protected resource again with new RPT - ACK
RESTful APIs

- Obtaining RPT with various resources user can access (Authorization API)
- Protection API (managing resources, permission tickets, policies)
- Java client to simplify integration in Java applications. JS adapter for JS frontend clients
Thanks

Keycloak website https://www.keycloak.org/

Keycloak source https://github.com/keycloak/keycloak


Community mailing list https://lists.jboss.org/mailman/listinfo/keycloak-user

Mail mposolda@redhat.com