JBoss Authorization Server

Reference Guide

Preview Release

Introduction

Sohil Shah

1.1. Authorization

Once a **Subject** (an Identity, Machine, etc) is authenticated by a system, **Authorization** is the security aspect that is used to determine: "What resources are they allowed access to within the system?"

Any Enterprise application requires flexible Authorization from its Security infrastructure with the following characteristics:

- Clean Separation between Security Logic and Application Logic
- Flexibility to apply Security Logic to arbitrary Runtime information
- Runtime Management of Security Policy
- A user friendly Developer API

1.1.1. Clean Separation between Security Logic and Application Logic

1.1.2. Flexibility to apply Security Logic to arbitrary Runtime information

1.1.3. Runtime Management of Security Policy

1.1.4. A user friendly Developer API

Concepts

Sohil Shah

2.1. Resource

A Data, Service, or a System component.

Examples : a HTTP URL, a Servlet, a Portlet, a POJO (Plain Old Java Object), a Java Method, a Java Field, etc

2.2. Action

An operation on a resource.

Examples : CRUD (Create, Read, Update, Delete), HTTP GET, HTTP POST, PORTLET VIEW, PORTLET ACTION, etc

2.3. Subject

An actor.

Examples : Authenticated User, Anonymous User, a Machine, etc

2.4. Environment

The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource, or action.

Examples : Current Date and/or Time, Application Data in a HTTP Session, the parameters passed to a Java method call, etc

2.5. Attribute

A characteristic of a Resource, Action, Subject or Environment which is referenced within a Policy Rule or a Policy Target. Attributes are runtime information which are presented with an Authorization Context during Enforcement. Within the policy definition, conditions, logic, and target matching are applied to information referenced by Attributes. The concept of Attributes allows Authorization to be flexible and allows including arbitrary runtime information during the decision process.

Resource Examples : Unique URI, Resource Id, File Name, etc

Action Examples : CRUD (Create, Read, Update, Delete), HTTP GET, HTTP POST, etc

Subject Examples : Username, Roles for this User, IP Address, Authentication Method, Authentication Time, etc

Environment Examples : Current Date and/or Time, etc

2.6. Policy

A security policy consisting of a target and multiple rules. An Enterprise application will have multiple policies stored in the system. Enforcement requests are evaluated by applying the logic specified within these policies. A Policy Evaluation results in a Permit or Deny State.

2.7. Target

The set of Enforcement requests identified by policy definitions of Resource, Subject, and Action that a Policy or a Rule is intended to evalute. Simply put, Target definition consists of logic that determines whether a particular Policy or Rule should be evaluated for the incoming Enforcement request.

2.8. Rule

A Policy Component which consists of the following:

Target : To determine if the Rule should be evaluated for the incoming Enforcement request.

Expression : Encapsulates the Logic that must be evaluated resulting in a Boolean (true|false) result.

Effect : Decides what to do (Permit or Deny), if this Rule evaluates to true.

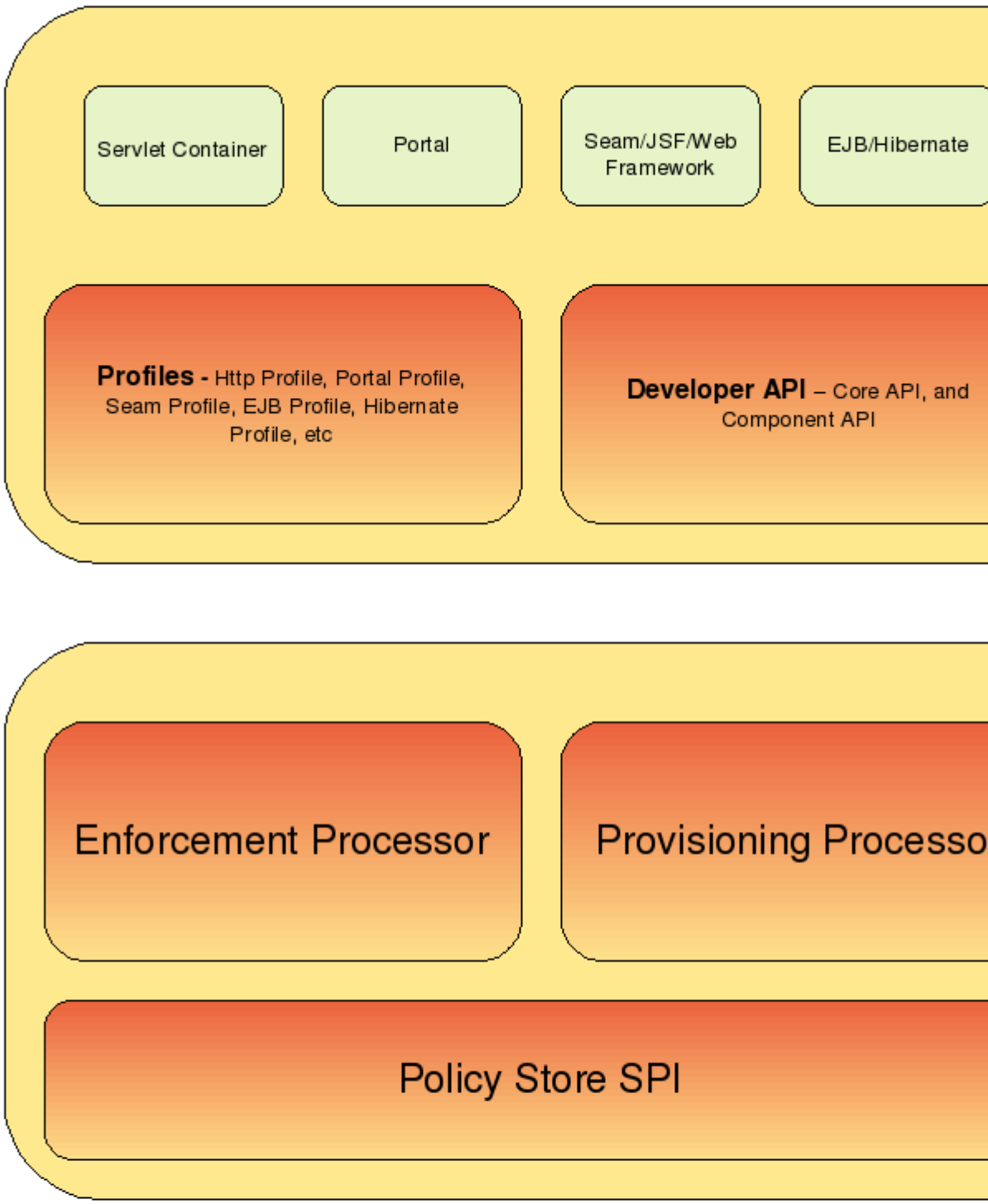
Architecture

Sohil Shah

3.1. Overall Architecture

JBoss Authorization Server architecture consists of two components:

- Application Framework
- Authorization Server


Servlet Container
Portal
Seam/JSF/Web Framework
EJB/Hibernate
Profiles - Http Profile, Portal Profile, Seam Profile, EJB Profile, Hibernate Profile, etc
Developer API – Core API, and Component API
Enforcement Processor
Provisioning Processo
Policy Store SPI
JCR Repository
Relational Database
LDAP Reposito

3.1.1. Application Framework

Application Framework consists of a Developer API that is used to integrate an Authorization layer for your application. This allows your Application components to be free from any embedded Security Logic. The Framework consists of entities that can be used to create custom Authorization layers within the context of your application, and with Profiles for infrastructure tiers that allows you to integrate Authorization layers as a simple drop in, without requiring any coding (Declarative Security Policy).

3.1.1.1. Profiles

Profiles are complete implementations of an Authorization layer within the context of the infrastructure tier in question. Examples of such profiles are the *HTTP Profile* for the HTTP tier, *Portal Profile* for a Portal/Portlet Container, *Seam Profile* for a Web Tier, *EJB Profile* for the Service Tier, *Hibernate/JPA Profile* for the Data Tier, etc

Each Profile will produce an Authorization Context with Attribute Values that make sense within the Context of that tier. Here is a simple snapshot of what a "Permit" Authorization Context would look like for a simple Policy Rule in each infrastructure tier:

"Allow Writing to a File '/developer/index.html', if User is in Role 'Developer'":

- *HTTP Profile* :

```
Resource Attributes: [urlPattern="/developer/index.html"]

Action Attributes: [parameter['action']="write"]

Subject Attributes: [roles={"project_manager", "developer"}]
```

- *EJB Profile* :

```
Resource Attributes: [Java Class="org.cms.CMS", Java Method="writeFile"]

Environment Attributes: [parameter['fileName']="/developer/index.html"]
```

```
Subject Attribute: [roles={"project_manager", "developer"}]
```

- *Hibernate Profile* :

```
Resource Attributes: [Java Class="org.cms.domain.File"]

Environment Attributes: [field['fileName']="/developer/index.html"]

Subject Attributes: [roles={"project_manager", "developer"}]
```

Note: Which infrastructure profiles you decide to activate within a single Enterprise application is dependent upon the Security requirements of the application, and the corresponding Policy makeup.

Profile Implementations are expected to be complete with the following features:

- *A Drop In Authorization Layer, not requiring any code changes to business components*
- *XML Configuration to specify Security Declaratively*
- *Developer Components built on top of the Developer API to develop Policy Provisioning functions (typically used for building Custom GUI)*
- *An Optional tier-specific Generic GUI for Policy Provisioning*

3.1.1.2. Developer API

A Developer API is provided by the Framework to develop custom Authorization functionality or for developing Authorization Profiles discussed above. The Developer API is split into two types. The very low-level API which presents components for the Authorization Concepts discussed in the *Concepts Chapter*. It also provides a higher-level API consisting of Application level components that are built on top of the low-level API. The Authorization Server comes packaged with a set of core components which will be discussed in more detail in the *Developer API Chapter*

Developers can easily create their own custom components using these low-level and/or higher-level components and have these components easily processed by the Authorization Server.

The Developer API is used to create the following Authorization functionality:

- *Custom Authorization Layer*
- *Custom Provisioning Functionality such as Security GUI, Command Line tools, etc*
- *Custom Components*
- *More Profiles*

3.1.2. Authorization Server

This component provides the runtime, the rules engine, and policy store which performs evaluation of Authorization requests against the Authorization Policies for the applications. It is used by the Application/Profile-level Authorization layer to route requests and correspondingly receive "Permit" or "Deny" responses. The functionality of the Server is split into the following concerns:

3.1.2.1. Enforcement

This component is responsible for processing incoming Authorization requests. It routes these requests to the rules engine, which in turn responds with a 'Permit' or 'Deny' state for the request.

The rules engine used is standards-compliant and uses the industry supported XACML standard. In a nutshell, XACML is an XML based Security Rules Language, use to define Security Policies. It is also comprehensive in providing robust concepts and architecture recommendations, but its main focus is the XML-spec to define Security Policies. This system is designed in accordance with these architectural recommendations, and also uses the XACML XML-spec to represent the Security Policies.

It must be noted that the XACML spec is vast and the XML is extremely complex. However, the developers are completely insulated from any XACML XML via the Developer API. The Developer API is not the same thing as a JAXB model of the XACML spec. The Developer API consists of object oriented components from the Application domain. The Authorization runtime internally takes care of converting these components into XACML constructs, to be used by the Authorization Server.

The Authorization Server uses the JAXB model provided by the JBossXACML stack, and the XACML implementation provided by Sun Microsystems.

3.1.2.2. Policy Provisioning

This component processes Provisioning requests from Applications. This results in runtime changes to Security policies, and immediate enforcement of these policies, without requiring any restarting of the infrastructure.

3.1.2.3. Policy Store SPI

This SPI allows storage agnosticity for the Security Policies managed by the Server. By default, it uses relational database storage. However using the SPI you can store Policies inside JCR Repositories, LDAP repositories, or other storage locations.

3.1.2.4. Deployment Options

The Authorization Server can be deployed and accessed by Enterprise Applications in 2 ways: *Same VM Mode*, *Different VM mode*,

3.1.2.4.1. Same VM Mode

In this mode the Authorization Server is deployed in the same VM as a JBoss Microcontainer Service. All service requests are local method invocations.

3.1.2.4.2. Different VM Mode

In this mode the Authorization Server is deployed in a separate VM than its applications. In this mode, service requests are made via a Remote interface such as REST

Developer API

Sohil Shah

4.1. Developer API

Profiles

Sohil Shah

5.1. Http Profile

5.2. Portal Profile

5.3. Seam Profile

Policy Store SPI

Sohil Shah

6.1. Policy Store SPI

Examples

Sohil Shah