PicketLink & PicketBox

Developer Conference, Brno 2011

JBoss by Red Hat

Peter Škopek

Feb 11, 2011

Abstract

This presentation will introduce you to the PicketLink and PicketBox projects developed by JBoss. They security related and are part of JBoss Enterprise Application Platform 5.1 (EAP).

Section 1
Welcome

Agenda

Section 2
PicketBox

PicketBox

- PicketBox is a Java Security Framework that provides following functionality
  - Authentication Support
  - Authorization Support
  - Audit Support
  - Security Mapping Support
  - Oasis XACML v2.0 compliant engine
- current version is PicketBox 3.0.0.Final
- Former name of PicketBox was JBoss Security

Authentication

- It is based on JAAS which is available as part of the JDK
  Note: JAAS = Java Authentication and Authorization Service
- PicketBox provides simple various authentication and authorization modules
  - Advanced LDAP based Authentication using LdapExtLoginModule
  - LDAP based Authentication using LdapLoginModule
  - Database based Authentication using DatabaseServerLoginModule
  - File based Authentication using UsersRolesLoginModule
- More about PicketBox Authentication you can find at `http://community.jboss.org/wiki/PicketBoxAuthentication`

Authorization

- Coarse Grained
  - You can use the PicketBox authorization modules to provide access control to your java application
- Fine Grained including Instance Based Authorization
  - Standards based Oasis XACML v2 Authorization using JBossXACML
  - Access Control Lists (ACLs) using PicketBox ACL

Coarse Grained Authorization Example

```
//Variables
private final String securityDomainName = "test";
private final String configFile = "config/authorization.conf";

public void testValidAuthorization() throws Exception {
    SecurityFactory.prepare();
    try {
      PicketBoxConfiguration idtrustConfig = new PicketBoxConfiguration();
      idtrustConfig.load(configFile);

      AuthenticationManager am = SecurityFactory.getAuthenticationManager(securityDomainName);
      assertNotNull(am);

      Subject subject = new Subject();
      Principal principal = getPrincipal("anil");
      Object credential = new String("pass");

      boolean result = am.isValid(principal, credential, subject);
      assertTrue("Valid Auth", result);
      assertTrue("Subject has principals", subject.getPrincipals().size() > 0);

      AuthorizationManager authzM = SecurityFactory.getAuthorizationManager(securityDomainName);
      assertNotNull(authzM);
      Resource resource = getResource();
      int decision = authzM.authorize(resource, subject);
      assertTrue(decision == AuthorizationContext.PERMIT);
    }
    finally {
      SecurityFactory.release();
    }
}
```

Coarse Grained Authorization - config file

```
<?xml version='1.0'?>
<policy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="urn:jboss:security-config:5.0"
         xmlns="urn:jboss:security-config:5.0"
         xmlns:jbxb="urn:jboss:security-config:5.0">
   <application-policy name = "test">
       <authentication>
          <login-module code = "org.jboss.security.auth.spi.UsersRolesLoginModule"
             flag = "required">
             <module-option name = "name">1.1</module-option>
             <module-option name = "succeed">true</module-option>
             <module-option name = "throwEx">false</module-option>
          </login-module>
       </authentication>
       <authorization>
          <policy-module
            code="org.picketbox.plugins.authorization.PicketBoxAuthorizationModule">
            <module-option name="roles">validuser</module-option>
          </policy-module>
       </authorization>
    </application-policy>
</policy>
```

PicketBox Authorization Annotations

- We can reduce our boiler plate code using PicketBox Annotations on POJOs.
  - @SecurityDomain Annotation
  - @Authentication Annotation
  - @Authorization Annotation
  - @SecurityMapping Annotation
  - @SecurityAudit Annotation
  - @Module Annotation
  - @ModuleOption Annotation
  - @SecurityConfig Annotation
- More at `http://community.jboss.org/wiki/PicketBoxSecurityAnnotations`

Annotated POJO example

```
import org.jboss.security.annotation.Authentication;
import org.jboss.security.annotation.Authorization;
import org.jboss.security.annotation.Module;
import org.jboss.security.annotation.ModuleOption;

import org.jboss.security.auth.spi.UsersRolesLoginModule;
import org.picketbox.plugins.authorization.PicketBoxAuthorizationModule;

@Authentication(modules={@Module(code = UsersRolesLoginModule.class, options =
  {@ModuleOption})})
@Authorization(modules ={@Module(code = PicketBoxAuthorizationModule.class, options =
  {@ModuleOption(key="roles",value="validuser")})})
public class AuthAuthorizationAnnotatedPOJO {

  ....

}
```

Fine Grained Authorization

- Standards based Oasis XACML v2 Authorization using JBossXACML
- Access Control Lists (ACLs) using PicketBox ACL
  The API encompasses the following interfaces:
  - ACL – represents an access control list. Defines methods to manipulate the entries and to check whether an identity has a set of permissions or not.
  - ACLEntry – represents an entry in the ACL.
  - ACLPermission – represents a permission.
  - ACLPersistenceStrategy – defines methods to persist/retrieve ACLs.
  - ACLProvider – basically a facade to the entire ACL subsystem.
  - RoleBasedACLProvider – a provider that uses the roles associated with the identity when looking for permissions.

Standards based Oasis XACML v2 Authorization using JBoss XACML

- Oasis XACML v2.0 library
- JAXB v2.0 based object model
- ExistDB Integration for storing/retrieving XACML Policies and Attributes

Usage of JBoss XACML

Project PicketBox from JBoss has an XACML engine that can be used in your Java environment.
Assuming that your configuration file is available, something like the following code should work for you:

```
import org.jboss.security.xacml.core.JBossPDP;
import org.jboss.security.xacml.interfaces.PolicyDecisionPoint;
import org.jboss.security.xacml.interfaces.XACMLConstants;

//Get hold of an InputStream to the config file
ClassLoader tcl = Thread.currentThread().getContextClassLoader();
InputStream is = tcl.getResourceAsStream( MY_CONFIG_FILE );

PolicyDecisionPoint pdp = new JBossPDP(is);

//Form your RequestContext by some means
ResponseContext response = pdp.evaluate(request);

int decision = response.getDecision();
//Decision can be one of the following

if (decision == XACMLConstants.DECISION_DENY)
   // your deny code here
else if (decision == XACMLConstants.DECISION_PERMIT)
   // your permit code here
else
   // throw unexpected state exception
```

Section 3
PicketLink

PicketLink

- PicketLink is an umbrella project that aims to address different Identity Management needs
- PicketLink consists of following projects:
  - IDM - Provide an object model for managing Identities (Users/Groups/Roles) and associated behavior using different identity store backends like LDAP and RDBMS
  - Federated Identity - Support SAMLv2, WS-Trust and OpenID
  - AuthZ - Developer friendly authorization framework
  - XACML - Oasis XACMLv2 implementation
  - Negotiation - Provide SPNego/Kerberos based Desktop SSO

Relationship Between PicketBox and PicketLink

- PicketBox is the foundational security framework that provides the authentication, authorization, audit and mapping capabilities to Java applications
- PicketLink (formerly, JBoss Identity) builds on PicketBox foundation and provides an identity model, federated identity support (SAML, WS-Trust, OpenID), Authz(access control developer api), Negotiation (SPNego/Kerberos based desktop SSO)

Current status of the project in terms of Red Hat products

- PicketLink is a community project. It is slowly making its way into the Enterprise Platforms sold by Red Hat.
  - Tech Preview in SOA-P5. (ESB SAML Token Support)
  - Tech Preview in EAP 5.1 (Federation Subsytem is included)
  - Included in EPP5 (Officially the IDM Subsystem is supported)

PicketLink Federation - Overview

- The PicketLink Federation project provides the support for Federated Identity and Single Sign On type scenarios.
- We provide support for technologies
  - Oasis SAML v2.0
  - Oasis WS-Trust v1.3
  - OpenID
- We have planned support for OAuth
- Integration with following servers is supported
  - JBoss Application Server v5.0 onwards
  - Apache Tomcat v5.5 onwards

The Fed Project Features

- Federated Authentication and SSO using Oasis SAML v2.0
- Trusted Security System using a Security Token Server (STS) in an heterogeneous environment, using Oasis WS-Trust
- Decentralized user driven Identity support via OpenID

PicketLink Seam Module - Introduction

PicketLink has a Seam module that enables developers to connect their Seam applications to external identity providers. SAMLv2 as well as OpenID based providers are supported.
There is a sample application called seam-sp, which can be used to play around with a very simple Seam application that enables users to login at an OpenID or SAML identity provider.
[SeamOpenSSO]

Setup

- We are going to use seam-sp application from PicketLink. `https://svn.jboss.org/repos/picketlink/federation/branches/Branch_1_x/picketlink-webapps/seam-sp`
- Now proceed OpenSSO admin console and choose "Create hosted entity provider". Choose "test" as the signing key, and enter the name "mycircle" for the new circle of trust and accept all other settings without a change. Press the "configure" button and your identity provider has been configured.
- Restart the OpenAM server.
- Build the application using maven and install it as exploded to $JBOSS_HOME/server/default/deploy

redhat.

Setup part 2

- Replace content of EntityDescriptor with entityID="http://localhost:8888/opensso" in seam-sp.war/WEB-INF/classes/saml-entities.xml file with `http://localhost:8180/opensso/saml2/jsp/exportmetadata.jsp`
- Locate

  ```
  <SamlIdentityProvider
  entityId="http://localhost:8888/opensso" />
  ```

  in seam-sp.war/WEB-INF/classes/external-authentication-config.xml file and change port part of URL to 8180.

Configure seam-sp as a service provider in OpenAM

- In OpenSSO admin console choose "Register Remote Service Provider". It will prompt you for a URL where the meta data of the service provider is located. Fill in the following URL: `http://localhost:8080/seam-sp/MetaDataService.seam`
- Press button "Configure“
  Note: Server wich serves the seam-sp application has to be up and running.

Bibliography

PicketBox Web Site
http://www.jboss.org/picketbox/

PicketLink Web Site
http://www.jboss.org/picketlink/

OpenAM download site.
http://forgerock.com/downloads.html

Anil Saldhana Blog
the guy to follow if you are seriuos about security
http://anil-identity.blogspot.com/

Marcel Kolsteren
External authentication example using OpenSSO
http://community.jboss.org/wiki/ExternalauthenticationexampleusingOpenSSO

redhat.

The end.

Thanks for listening.