redhat. TRAINING

CHAPTER 9

JBOSS EAP PROTECTION

General description	
Meta	Apply security settings to increase the security of applications deployed on JBoss EAP. • Configure a
Goals	security domain based on the database login module. • Configure a security domain based on the LDAP login module. • Configure role-based access to topics and queues in the messaging subsystem. • Protect passwords stored in server configuration files using the password vault.
sections	• Configuring a database security domain (and guided exercise) • Configuration of a security domain based on LDAP (and guided exercise) • Protection of a JMS destination (and questionnaire) • Configuration of the password store (and exercise guided)
Laboratory work	• JBoss EAP Protection

Configuring a database security domain

Goals

After completing this section, students should be able to do the following:

- **Configure a security domain based on the database login module.**
  data.

security subsystem

In accordance with EAP's modular design, a subsystem deals with security. Security settings are presented in the *security subsystem* and configured in the domain.xml (in managed domain mode) and standalone.xml (on a standalone server) configuration files.

The EAP security subsystem consists of a number of security-domains, which define how applications are authenticated and authorized. Users can define their own security domain backed by a system that stores identity information (database, LDAP, etc.) and configure applications to use the newly defined security-domain in application deployment descriptors.

The basic principle behind security subsystem design is to avoid embedding authentication and authorization logic within applications and to capture implementation details within reusable components (security-domains) that applications can then reference in the runtime. For example, administrators can change the authentication and authorization mechanism for an application from a database to LDAP without making changes to the source code in the application.

References

The security subsystem in EAP 6 was developed within the Picketlink project. In EAP 7, Picketlink was replaced by the Elytron project. For more details, see

> **The Elytron project https://developer.jboss.org/wiki/WildFlyElytron-ProjectSummary**

security domains

A security domain is a set of Java Authentication and Authorization Service (JAAS) security settings that one or more applications use to control authentication, authorization, auditing, and mapping.

A security domain consists of configurations for authentication, authorization, security assignment, and auditing. Users can create as many security domains as needed to suit application requirements.

By default, EAP has four security domains defined:

- jboss-ejb-policy: The default authorization security domain for the EJB modules in applications that did not explicitly define a security-domain.
- jboss-web-policy: The default authorization security domain for the web modules in applications that did not explicitly define a security-domain.
- other – This security domain is used internally within JBoss EAP for authorization, and is required for proper operation.
- jaspitest: The jaspitest security domain is a simple security domain of the Java Authentication Service Provider Interface for Containers (JASPIC), which is included for development purposes.

References

For more details on JASPIC, see

JASPIC
https://jaspic.zeef.com/

Security domains can be defined and configured using the EAP CLI or in the management console in the Configuration > Subsystems > Security section.

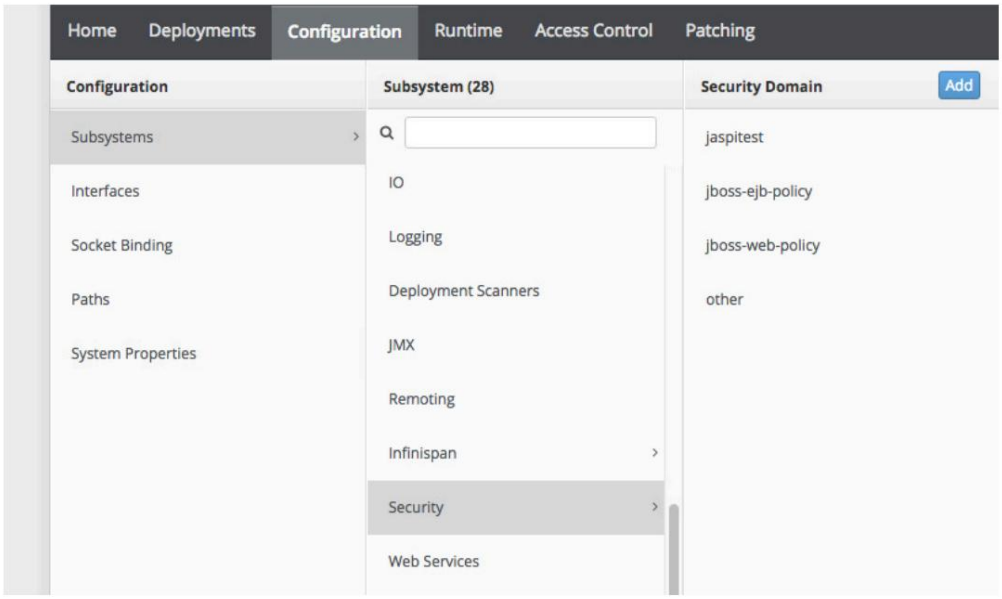

Figure 9.1: Configuration of security domains in the administration console

The format of a security-domain definition in standalone.xml or domain.xml looks like this:

```
<security-domains>
    <security-domain>
        <authentication>
            <login-module>
                ...
            </login-module>
```

```
            </authentication> </
        security-domain> </
security-domains>
```

EAP includes several built-in login modules, which can be used for authentication within a security domain. The security subsystem offers some basic login modules that can read user information from a relational database, LDAP server, or flat files. In addition to these basic login modules, EAP offers other login modules that can be customized based on application security requirements.

References

For a detailed list of all login modules shipped with EAP, see

> **EAP 7 Login Modules Reference** https://
> access.redhat.com/documentation/en/red-hat-jboss-enterprise
> application-platform/7.0/login-module-reference/login-module-reference

Defining a Database Security Domain Users can define a database-backed security domain, which

stores users and role assignments for an application, using the Database login module. It uses a reference named datasource, defined in the EAP configuration file, and defines queries to get the users and roles from the database. The recommended approach is to store passwords in an encrypted format by setting the hashAlgorithm attribute for better security in the database.

A database security domain can be created with the EAP CLI, using the following command:

```
[standalone@localhost:9990] /subsystem=security/security-domain= \ bksecurity-
  domain:add(cache-type=default)
```

The following command associates the security domain with a database for authentication purposes:

```
[standalone@localhost:9990] /subsystem=security/security-domain= \ db-domain/
  authentication=classic:add \
  (login-modules=[{"code"=>"Database", "flag"=>"required", \ "module-
  options"=>[("dsJndiName"=>"java:jboss/datasources/test-ds"), \ ("principalsQuery"=>"SELECT
  password FROM users WHERE username = ?"), \ ("rolesQuery"=>"SELECT role, 'Roles'
  FROM roles WHERE username = ?"), \ ("hashAlgorithm"=>"SHA-256"),
  ("hashEncoding"=>"base64")]}])
```

The corresponding XML definition of the database security domain is as follows:

```
<security-domain name="db-domain" cache-type="default">
```

```
    <authentication>
      <login-module code="Database" flag="required">
        <module-option name="dsJndiName"
                value="java:jboss/datasources/test-ds"/>
        <module-option name="principalsQuery"
                value="SELECT password FROM users WHERE username = ?"/>
        <module-option name="rolesQuery"
                value="SELECT role, 'Roles' FROM roles WHERE username = ?"/>
        <module-option name="hashAlgorithm" value="SHA-256"/>
        <module-option name="hashEncoding" value="base64"/> </login-module>
      </authentication>

</security-domain>
```

1. **The name module option is required and is the name of the security domain. It must be unique within the standalone.xml file (on a standalone server) or within a profile (on a managed domain).**
2. **The module option code is a unique string that identifies the login module to use for this security domain. In this case, the built-in Database login module is used.**
3. **The dsJndiName module option references a valid data source defined in the configuration file.**
4. **The principalsQuery module option defines the SQL query that retrieves a single row from the database containing the details of the user who is logging into the application. In this case, a unique username (primary key) is obtained from the table users.**
5. **The rolesQuery module option defines the SQL query that retrieves a list of roles for a particular user. In this case, a list of roles for a particular user is obtained from the roles table. Also, the "Roles" column can refer to another string to create subroles for an existing role. For most applications, the Roles chain can be used.**
6. **The hashAlgorithm module option is the hash algorithm to use to hash the user's passwords. Possible options are MD5, SHA-1, SHA-256, and SHA-512.**
7. **The hashEncoding module option defines the hashing of the password. Possible options are hex and base64.**

When a request is made to this EAP server that requires authentication of a user through db-domain, the following steps are performed:

1. **The username is obtained from the request.**

2. A connection to the database is retrieved from the connection pool with the nombre java:jboss/datasources/test-ds.

3. The principalsQuery is executed and the placeholder (?) is replaced with the username from the request.

4. If the password retrieved from the database does not match the password sent in the request, the security domain notifies the EAP container that this user has not been authenticated. Therefore, the request is not processed further.

5. If the password retrieved from the database matches the password submitted in the request, the rolesQuery is executed and the placeholder (?) is replaced with the username from the request.

6. If the user does not belong to a required role of the requested resource, the domain of security notifies the EAP container that said user is not authorized. Therefore, the request is not processed further.

7. If the user does belong to one of the required roles of the requested resource, the user is authorized and the request is transmitted to the requested resource.

Configuring a security domain in an application

To use a security domain, users must add a reference to the security domain in jboss-web.xml. The following is an example of the <security domain> tag in jboss-web.xml:

```
<security-domain>Security_Domain_Name</ security-domain>
```

The web.xml file allows users to configure application security, creating security restrictions to specific paths, restricting based on roles, and defining the authentication type.

In the <security-constraint> part of the web.xml file is the <web resource-collections> section that allows users to define which URL path requires authorization. The <url-pattern> tag can contain any path on the server or just use a to guarantee all pages and all resources. The <auth-constraint> tag within the same section specifies which roles are allowed access to restricted aspects of the application.

```
...
<security-constraint> <web-
    resource-collection> <web-
        resource-name>All resources</web-resource-name> <url-pattern>/*</
        url-pattern> </web-resource-collection>
    <auth-constraint>

        <role-name>*</role-name>
    </auth-constraint>
</security-constraint>
...
```

The <login-config> tag within the web.xml defines the type of authentication to use. BASIC refers to an authentication requirement where the server requests a username and password from the web client, which is validated against the database. Defining these aspects of web.xml is beyond the scope of this course.

```
...
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
```