Guided Exercise: Encrypting a Password

In this lab, you will secure the MySQL bookstore data source by storing the password in a store file.

Resources	
Files:	/home/student/JB248/labs/security-encrypt/
App URL:	http://127.0.0.1:8080/dstest

Results

You should be able to keep the MySQL bookstore application database password, which is encrypted and protected in a store file.

Before you begin

Before you begin the guided exercise, run the following command to verify that EAP is installed in /opt/jboss-eap-7.0, that no EAP instance is running, and that the MySQL bookstore data source is configured, and to download the lab work files:

```
[student@workstation ~]$ lab securing-encrypt setup
```

1. Create a Java keystore to store the database password.
data.

1.1. Open a new terminal window and run the following commands to create a keystore file named vault.keystore in /home/student/:

```
[student@workstation ~]$ keytool -genseckey -alias vault \ -keyalg AES
-storetype jceks -keysize 128 \ -keystore /home/student/
vault.keystore
```

Notice that the alias value is vault and refers to a key vault entry where the password will be stored.

1.2. When prompted, use "password" as the password for the keystore and certificate passwords.

1.3. Verify that you now have a file named vault.keystore in / home/student.

```
[student@workstation ~]$ ls | grep vault vault.keystore
```

2. Run the vault script to encrypt a password.

2.1. Enter the following command to run the vault tool script, which is used to add passwords to a vault:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./vault.sh
```

2.2. At the first prompt, enter 0 to select Start Interactive Session.

23. When the directory for encrypted file storage is requested, enter /home/student/. (Note that the trailing slash is needed.)

2.4. The keystore URL is the path to the vault.keystore file you created in the previous step. Enter:

```
/home/student/vault.keystore
```

2.5. For the remaining requests, use the following values: • Password: password

• Jump of 8 characters: 12345678

• Iteration Count: 50

• Key vault alias: vault

2.6. Make a Warehouse Configuration note.

The warehouse tool is now connected to the warehouse. Enter 0 to store a secured attribute.

2.7. When prompted for the attribute value, enter redhat, which is the password to connect to the MySQL database running on your computer. You will need to enter the redhat password twice to verify it.

2.8. When prompted for a Store Block, enter bookstore.

2.9. At the Enter attribute name prompt, enter password.

2.10.Write down the resulting information, as shown:

```
Please make note of the following:
********************************************
Vault Block:bookstore
Attribute Name:password
Configuration should be done as follows:
VAULT::bookstore::password::1
********************************************
```

2.11.The password for the MySQL database is now in vault.keystore, so enter 3 to Exit the vault tool.

3. Set up the warehouse.

3.1. Abra /home/student/JB248/labs/standalone/configuration/ standalone.xml in a text editor.

3.2. To open the store, an EAP instance must be configured in the section <vault> of the standalone.xml configuration file, which appears between the <extensions> and <management> entries. Add the following <vault> section immediately before the <management> section in /home/student/JB248/labs/ standalone/configuration/standalone.xml:

```
<vault>
    <vault-option name="KEYSTORE_URL" value="/home/student/vault.keystore"/> <vault-option
    name="KEYSTORE_PASSWORD" value="MASK-31x/z0Xn83H4JaL0h5eK/N"/> <vault-option
    name="KEYSTORE_ALIAS" value="vault"/> <vault-option name="SALT"
    value="12345678"/> <vault-option name="ITERATION_COUNT"
    value="50"/> <vault-option name="ENC_FILE_DIR" value="/home/
    student/"/> </vault>
```

You can copy and paste this XML from the /home/student/JB248/labs/security-encrypt/vault.xml file.

Save the changes to standalone.xml.

3.3. Start the standalone instance. In the first few lines of the log output, you should see log events that show a security store successfully started and ready.

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ ./standalone.sh \
-Djboss.server.base.dir=/home/student/JB248/labs/standalone/
```

In the first few lines of the log output, you should see log events showing a security store successfully started and ready:

```
17:37:39,730 INFO [org.jboss.security] (Controller Boot Thread) PBOX00361: Default Security Vault
  Implementation Initialized and Ready
```

4. Set up the data source.

4.1. Navigate to the Management Console Settings page at http://localhost:9990.

4.2. Click Subsystems and then Data Sources to access the subsystem from the data source.

4.3. Click the bookstore data source in the list of Non-XA data sources, and then click View.

4.4. Click Disable to disable the bookstore data source. (You cannot modify the security settings of a data source currently in use.)

If you are prompted to reload the server, reload the server and return to the bookstore data source administration page.

4.5. Select Security, and then click Edit.

4.6. In the Password text field, remove the current password value.

4.7. Inside the ${} notation, copy and paste the configuration entry that appears at the end of the depot tool script. The Password field should appear as follows:

```
${VAULT::bookstore::password::1}
```

use

You won't be able to see what you type, so be sure to copy and paste to avoid entering data incorrectly.

4.8. Click Save to save your changes.

4.9. Re-enable the bookstore data source by clicking Enable.

4.10. Reload the server using the EAP CLI to allow the changes to take effect.

```
[student@workstation ~]$ /opt/jboss-eap-7.0/bin/jboss-cli.sh --connect
[standalone@localhost:9990] :reload
```

5. Check the data source.

5.1. Deploy the dstest.war file located at /tmp/dstest.war using the CLI of EAP:

```
[standalone@localhost:9990 /] deploy /tmp/dstest.war
```

5.2. Open the standalone.xml file at /home/student/JB248/labs/standalone/configuration. Check the <password> entry so that bookstore is enclosed in ${} and contains the text you copied and pasted in the previous step.

5.3. Go to http://127.0.0.1:8080/dstest/.

5.4. Enter java:jboss/datasources/bookstore for the JNDI name and bookstore.CatalogItem for the table name.

5.5. Click Submit and verify that the database has been connected successfully.

6. Perform cleaning.

6.1. Undeploy the dstest.war application:

```
[standalone@localhost:9990 /] undeploy dstest.war
```

6.2. Stop the running EAP instance.

This concludes the guided exercise.

Lab Work: Securing JBoss EAP

In this lab work, you will secure an application, protect a data source password with a store, and secure a queue.

Resources	
Files:	/home/student/JB248/labs/security-lab
App URL:	http://172.25.250.10:8080/messaging-client

Results

You should be able to configure the security domain to secure the messaging client application, encrypt the data source password for the bookstore data source, and apply role-based access control to the queue created in the previous exercise.

before you start

Before beginning the guided exercise, run the following command to verify that no EAP instances are running and that the above lab work is complete, and to download the lab files:

```
[student@workstation ~]$ lab securing-lab setup
```

The first part of this lab will focus on securing the messaging client application used in the final lab in the previous chapter. Use the following steps to create a database backed by the security domain and to deploy the application to a managed domain.

After the app has a security domain configured, it will secure the data source that is used to verify app users. At the moment, the bksecurity data source stores the password in plain text in the domain.xml file.
Put the password in a new store and update the data source to reference it for the data source credentials.

Finally, update the security for the messaging queues that were created in the previous lab work. Currently, any user can send or receive messages. Limit shipping privileges to admin users only.

1. Configure the bksecurity data source in the managed domain.

Create a new data source in the managed domain in the full-ha profile for the bksecurity database. The data source must have the following characteristics and must use the mysql driver already installed: •

Name: bksecurity-ds

• Nombre de JNDI: java:jboss/datasources/bksecurity-ds

• Connection URL: jdbc:mysql://172.25.250.254:3306/bksecurity

• Username: bkadmin

• Password: redhat

Confirm that the connection is working correctly.

2. Create a database authentication login module.

Create a security domain that will use the data source added in the previous step to get the credentials used by the sample application from a database. The security domain must have a default cache type, which will be used for authentication purposes. The security domain must have the following characteristics:

- Security domain name: jb248-sd
- Name of the login module: database
- Code: Database
- Marker: required
- These module options:
  - ÿ *dsJndiName:* java:jboss/datasources/bksecurity-ds
  - ÿ *principalsQuery:* select password from users where username=?
  - ÿ *rolesQuery:* select role, 'Roles' from roles where username=?
  - ÿ *hashAlgorithm:SHA-256*
  - ÿ *hashEncoding: base64*

use

If there are typos or errors in the login module, it's easier to update the module options in the admin console. Access the security subsystem to troubleshoot any errors in the login module.

3. Deploy the secure messaging client.

In the previous lab work, you implemented a simple messaging client. Browse the source for an up-to-date and secure version of the same app. Open web.xml and jboss-web.xml to confirm that the application uses a new security domain, jb248-sd. Next, deploy the war file located at /tmp/ messaging-client-secure.war to the Group 1 server group.

The application source code is located at /home/student/JB248/labs/security-lab/messaging-client-secure/. To verify that it is secure, go to http://172.25.250.10:8080/messaging-client and use the credentials admin for the username and admin for the password.

4. Create a vault key vault.

The bksecurity data source has the username and password stored in plain text in the domain.xml file, which creates a high security risk. Store the password in a store and reference it in the configuration to protect data source credentials.

Create a key vault with the following characteristics:

- **Alias: vault**
- **keyalg: AES**
- **genseckey: enabled**
- **case type: jceks**
- **keysize: 128**
- **keystore: /opt/jboss-eap-7.0/vault.keystore**

When prompted, use "password" as the vault password.

5. Create an entry with the password for the bksecurity data source.

Create a store to hold the bksecurity data source password. The vault must have the following characteristics and refer to the key vault created in the previous step located at /opt/jboss-eap-7.0/vault.keystore:

- **Password: password**
- **Jump of 8 characters: 12345678**
- **Iteration Count: 50**
- **Key vault alias: vault**
- **Store block: bksecurity**
- **Attribute name: password**
- **Attribute value: redhat**

Copy the store to servera and serverb, and update /opt/domain/configuration/host-slave.xml to include the store definition.

6. Secure the tail.

The credentials for the messaging producer and consumer are hardcoded in the admin user credentials.

Limit access to each queue in the application to ensure that only administrators can send and receive messages. Create a new security constraint for the admin role that has send, receive, and manage privileges. You must use # for the pattern so that values are applied to each queue. Remove the existing guest security configuration.

Use the following command to set the default security domain for the messaging subsystem to be jb248-sd:

```
[domain@172.25.250.254:9990 server=default] /profile=full-ha/subsystem\ =messaging-activemq/server=default:\ write-attribute(name=security-domain,value=jb248-sd)
```

7. Try the tail.

Access the secure messaging client application and send a test message to a queue by logging in as the admin user. Remove the send and receive privileges for this user and verify that they can no longer send a message.

7.1. Deploy the MDB application located at /tmp/messaging-mdb-secure on the Group1 server.

```
[domain@172.25.250.254:9990 /] deploy /tmp/messaging-mdb-secure.jar \ --server-
groups=Group1
```

7.2. Post a test message. Enter the message producer application running on servera.1 at http://172.25.250.10:8080/messaging client-secure using the admin/admin credentials, and complete the form as follows:

• Message count: 5

• Message label: testmsg

• Message to send: jms test

Go to http://172.25.250.10:8080/messaging-client-secure.

Complete the web form and click Send Message to send the message.

The web form should refresh and show the following message in green: Messages sent successfully.

7.3. Check server logs for entries generated by the application of the consumer. The servers servera.1 and serverb.1 must have log entries generated by the consumer application.

There should be registry entries similar to the following in the terminal window serve:

```
[Server:servera.1] 19:54:56,317 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-11 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #3 [testmsg]
[Server:servera.1] 19:54:56,318 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-12 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #5 [testmsg]
[Server:servera.1] 19:54:56,318 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-11 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test [Server:servera.1]
19:54:56,319 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-12 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test [Server:servera.1]
19:54:56,322 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-10 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #1 [testmsg]
[Server:servera.1] 19:54:56,323 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-10 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test
```

There should be log entries similar to the following in the serverb terminal window:

```
[Server:serverb.1] 19:54:58,282 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-9 (ActiveMQ-client  global-
threads-1005879511)) Message Properties: Copy #4 [testmsg]
[Server:serverb.1] 19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-8 (ActiveMQ-client  global-
threads-1005879511)) Message Properties: Copy #2 [testmsg]
[Server:serverb.1] 19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-9 (ActiveMQ-client  global-
threads-1005879511)) Message Body: jms test [Server:serverb.1]
19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-8 (ActiveMQ-client  global-
threads-1005879511)) Message Body: jms test
```

7.4. Go back to the administration console and update the security settings of the messaging subsystem for the default server. Click Edit next to the admin role values and remove the shipping and consuming privileges.

7.5. Go back to the app and try sending another test message with the following characteristics:

- **Message count: 5**
- **Message label: authorizationtest**
- **Message to send: authorization test**

7.6. In the server logs for servera, verify that no message was sent, as this user's role did not have the proper authorization to send a message.

```
[Server:servera.1] 19:00:27,100 INFO [stdout] (default task-8)
  javax.jms.JMSSecurityRuntimeException: AMQ119032: User: admin does not have permission='SEND' on
  address jms.queue.TestQueue
```

8. Perform cleaning and grading.

8.1. Press Ctrl+C to stop the domain controller on workstation and the two host controllers in servera and serverb.

8.2. Run the following command to grade the assignment:

```
[student@workstation bin]$ lab securing-lab grade
```

This concludes the lab work.

Solution

In this lab work, you will secure an application, protect a data source password with a store, and secure a queue.

Resources	
Files:	/home/student/JB248/labs/security-lab
App URL:	http://172.25.250.10:8080/messaging-client

Results

You should be able to configure the security domain to secure the messaging client application, encrypt the data source password for the bookstore data source, and apply role-based access control to the queue created in the previous exercise.

before you start

Before beginning the guided exercise, run the following command to verify that no EAP instances are running and that the above lab work is complete, and to download the lab files:

```
[student@workstation ~]$ lab securing-lab setup
```

The first part of this lab will focus on securing the messaging client application used in the final lab in the previous chapter. Use the following steps to create a database backed by the security domain and to deploy the application to a managed domain.

After the app has a security domain configured, it will secure the data source that is used to verify app users. At the moment, the bksecurity data source stores the password in plain text in the domain.xml file.
Put the password in a new store and update the data source to reference it for the data source credentials.

Finally, update the security for the messaging queues that were created in the previous lab work. Currently, any user can send or receive messages. Limit shipping privileges to admin users only.

1. Configure the bksecurity data source in the managed domain.

Create a new data source in the managed domain in the full-ha profile for the bksecurity database. The data source must have the following characteristics and must use the mysql driver already installed: •
Name: bksecurity-ds

• Nombre de JNDI: java:jboss/datasources/bksecurity-ds

• Connection URL: jdbc:mysql://172.25.250.254:3306/bksecurity

• Username: bkadmin

• Password: redhat

Confirm that the connection is working correctly.

1.1. Start the managed domain to enable the administration console to create and test the data source.

Start the host master on workstation:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ sudo -u jboss ./domain.sh \
-Djboss.domain.base.dir=/opt/domain/ \ --host-
config=host-master.xml
```

Start the host controller on servera:

```
[student@servera ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/domain.sh \
-Djboss.domain.base.dir=/opt/domain --host-config=host-slave.xml \
-Djboss.domain.master.address=172.25.250.254 \
-bprivate=172.25.250.10
```

Start the host controller on serverb:

```
[student@serverb ~]$ sudo -u jboss /opt/jboss-eap-7.0/bin/domain.sh \
-Djboss.domain.base.dir=/opt/domain --host-config=host-slave.xml \
-Djboss.domain.master.address=172.25.250.254 \
-bprivate=172.25.250.11
```

1.2. Add a new non-XA data source with the following credentials as a MySQL Data Source:

- **Nombre: bksecurity-ds**
- **Nombre de JNDI: java:jboss/datasources/bksecurity-ds**
- **Connection URL: jdbc:mysql://172.25.250.254:3306/bksecurity**
- **Username: bkadmin**
- **Password: redhat**

1.3. Access the admin console running at 172.25.250.254:9990, Click Settings, then click Data Source Subsystem in the full-ha profile. The username for the administration console is jbossadm and the password is JBoss@RedHat123.

1.4. Select the mysql driver by clicking the Detected Driver link and selecting the mysql module.

1.5. Test the data source in the management console to verify that it is working correctly. Click Connection and then Test Connection on the bksecurity-ds overview page. A popup window should confirm that the connection is valid.

2. Create a database authentication login module.

Create a security domain that will use the data source added in the previous step to get the credentials used by the sample application from a database.

data. The security domain must have a default cache type, which will be used for authentication purposes. The security domain must have the following characteristics:

• Security domain name: jb248-sd

• Name of the login module: database

• Code: Database

• Marker: required

• These module options:

ÿ *dsJndiName:* java:jboss/datasources/bksecurity-ds

ÿ *principalsQuery:* select password from users where username=?

ÿ *rolesQuery:* select role, 'Roles' from roles where username=?

ÿ *hashAlgorithm:SHA-256*

ÿ *hashEncoding: base64*

2.1. Click Security on the Settings page for the full-ha profile and click Click Add in the Security Domain column to add a new security domain named jb248-sd.

2.2. Enter the security domain name as jb248-sd, set the Cache Type to default, and then click Save.

23. To configure the authentication modules, start the EAP CLI on a new terminal and connect to the master of the host.

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ sudo -u jboss ./jboss-cli.sh --connect \
--controller=172.25.250.254:9990
```

2.4. Use the following command to create an authentication module.

```
[domain@172.25.250.254:9990] /profile=full-ha/subsystem=\ security/
security-domain\ =jb248-sd/
authentication\ =classic:add
```

2.5. Create a login module that will connect to the data source created in the previous step. You should query the users and roles table to identify the credentials. The password is encrypted with a SHA-256 hash algorithm, and the hash uses a base64 encoding.

Use the following values when creating the login module:

• Name: database

• Code: Database

- **Marker: required**
- **These module options:**
  - ÿ ***dsJndiName:*** **java:jboss/datasources/bksecurity-ds**
  - ÿ ***principalsQuery:*** **select password from users where username=?**
  - ÿ ***rolesQuery:*** **select role, 'Roles' from roles where username=?**
  - ÿ ***hashAlgorithm:SHA-256***
  - ÿ ***hashEncoding: base64***

Use the following CLI command to create the login module:

```
[domain@172.25.250.254:9990] /profile=full-ha/subsystem=security\ /security-
domain=jb248-sd/authentication=\ classic/login-
module=database:add( \ code=Database, \
  flag=required, \
  module-options=[ \

  ("dsJndiName"=>"java:jboss/datasources/bksecurity-ds"), \
  ("principalsQuery"=>"select password from users where username=?"), \
  ("rolesQuery"=>"select role, 'Roles' from roles where username=?"), \
  ("hashAlgorithm"=>"SHA-256"), \
  ("hashEncoding"=>"base64") \ ])
```

This command can be copied and pasted from /home/student/JB248/labs/security-lab/login-module.

2.6. Reload the server to allow the changes to take effect:

```
[domain@172.25.250.254:9990] :reload-servers(blocking=true)
```

2.7. Verify the values by running the following command on the CLI:

```
[domain@172.25.250.254:9990] /profile=full-ha/subsystem=security\ /security-
domain=jb248-sd/authentication=classic/\ login-
module=database:read-resource
```

The answer should be the following:

```
{
        "outcome" => "success",
        "result" =>
            { "code" => "Database",
            "flag" => "required", "module"
            => undefined, "module-
            options" => { ("rolesQuery"
                  => "select role, 'Roles' from roles where username=?"), ("principalsQuery" => "select password
                  from users where username=?"), ("dsJndiName" => "java:jboss/datasources/bksecurity-ds"),
                  ("hashAlgorithm" => "SHA-256"), ("hashEncoding" => "base64"),
```

```
            }
        }
}
```

use

If there are typos or errors in the login module, it's easier to update the module options in the admin console. Access the security subsystem to troubleshoot any errors in the login module.

3. Deploy the secure messaging client.

In the previous lab work, you implemented a simple messaging client. Browse the source for an up-to-date and secure version of the same app. Open web.xml and jboss-web.xml to confirm that the application uses a new security domain, jb248-sd. Next, deploy the war file located at /tmp/ messaging-client-secure.war to the Group 1 server group.

The application source code is located at /home/student/JB248/labs/security-lab/messaging-client-secure/. To verify that it is secure, go to http://172.25.250.10:8080/messaging-client and use the credentials admin for the username and admin for the password.

3.1. Abra el archivo /home/student/JB248/labs/security-lab/messaging client-secure/src/main/webapp/WEB-INF/web.xml. Notice the login config tag with the basic authorization method and the jb248-sd domain name.

```
<login-config>
    <auth-method>BASIC</auth-method> <realm-name>jb248-sd</realm-name> </login-config>
```

3.2. Abra el archivo /home/student/JB248/labs/security-lab/messaging client-secure/src/main/webapp/WEB-INF/jboss-web.xml. Inside the <jboss-web> tag, notice that <security-domain> is set to jb248-sd:

```
<security-domain>jb248-sd</security-domain>
```

Save the changes to web.xml.

3.3. Using the administrative console or the CLI, deploy the newly located messaging-client-secure.war file to /tmp/messaging client-secure.war on the Group1 server.

```
[domain@172.25.250.254:9990] deploy /tmp/messaging-client-secure.war --server  groups=Group1
```

Monitor the server log for errors in the deployment. The application should be deployed without errors.

3.4. Access the application with the username admin and password admin at http://172.25.250.10:8080/messaging-client-secure.

4. Create a vault key vault.

The bksecurity data source has the username and password stored in plain text in the domain.xml file, which creates a high security risk.
Store the password in a store and reference it in the configuration to protect data source credentials.

Create a key vault with the following characteristics:

- Alias: vault
- keyalg: AES
- genseckey: enabled
- case type: jceks
- keysize: 128
- keystore: /opt/jboss-eap-7.0/vault.keystore

When prompted, use "password" as the vault password.

4.1. Open a new terminal window and run the following commands to create a new keystore file named vault.keystore in /opt/jboss-eap-7.0/:

```
[student@workstation ~]$ sudo -u jboss keytool \ -genseckey
-alias vault \ -keyalg AES
-storetype jceks -keysize 128 \ -keystore /opt/jboss-
eap-7.0/vault.keystore
```

4.2. When prompted, use "password" as the password for your data repository.
keys.

4.3. Copy the new keystore to servera and serverb:

```
[student@workstation ~]$ sudo scp /opt/jboss-eap-7.0/vault.keystore \ servera:/opt/jboss-
eap-7.0 [student@workstation ~]$
sudo scp /opt/jboss-eap-7.0/vault.keystore \ serverb:/opt/jboss-eap-7.0
```

4.4. Change the ownership of the keystore to the jboss user on servera and serverb:

```
[student@servera ~]$ sudo chown jboss:jboss /opt/jboss-eap-7.0/vault.keystore
```

```
[student@serverb ~]$ sudo chown jboss:jboss /opt/jboss-eap-7.0/vault.keystore
```

5. Create an entry with the password for the bksecurity data source.

Create a store to hold the bksecurity data source password. The vault must have the following characteristics and refer to the key vault created in the previous step located at /opt/jboss-eap-7.0/vault.keystore:

- **Password: password**
- **Jump of 8 characters: 12345678**
- **Iteration Count: 50**
- **Key vault alias: vault**
- **Store block: bksecurity**
- **Attribute name: password**
- **Attribute value: redhat**

Copy the store to servera and serverb, and update /opt/domain/configuration/host-slave.xml to include the store definition.

5.1. Enter the following command to run the vault tool script, which is used to add passwords to a vault:

```
[student@workstation ~]$ cd /opt/jboss-eap-7.0/bin
[student@workstation bin]$ sudo -u jboss ./vault.sh
```

5.2. At the first prompt, enter 0 to select Start Interactive Session.

5.3. When the directory for encrypted file storage is requested, enter /opt/jboss-eap-7.0/. (Note that the trailing slash is needed.)

5.4. The keystore URL is the path to the vault.keystore file you created in the previous step. Enter:

```
/opt/jboss-eap-7.0/vault.keystore
```

5.5. For the remaining requests, use the following values:

- **Password: password**
- **Jump of 8 characters: 12345678**
- **Iteration Count: 50**
- **Key vault alias: vault**

5.6. Make a Warehouse Configuration note.

The warehouse tool is now connected to the warehouse. Enter 0 to store a secured attribute.

When prompted for the attribute value, enter redhat, which is the password to connect to the MySQL database. You will need to enter the redhat password twice to verify it.

5.7. When prompted to enter a Store Block, enter bksecurity, and in Enter attribute name, enter password.

5.8. Write down the resulting information:

```
Please make note of the following:
**********************************************

Vault Block:bksecurity
Attribute Name:password
Configuration should be done as follows:
VAULT::bksecurity::password::1
**********************************************
```

5.9. The password for the MySQL database is now located in vault.keystore. Enter 3 to Exit Tool Warehouse.

5.10. Stop the host on servera, serverb and workstation.

5.11. Add the following <vault> section immediately before the <management> section in /opt/domain/configuration/host-master.xml on workstation:

```
<vault>
    <vault-option name="KEYSTORE_URL" value="/opt/jboss-eap-7.0/vault.keystore"/> <vault-option
    name="KEYSTORE_PASSWORD" value="MASK-31x/z0Xn83H4JaL0h5eK/N"/> <vault-option
    name="KEYSTORE_ALIAS" value="vault"/> <vault-option name="SALT"
    value="12345678"/> <vault-option name="ITERATION_COUNT"
    value="50"/> <vault-option name="ENC_FILE_DIR" value="/opt/jboss-
    eap-7.0/"/> </vault>
```

use

This vault definition can be copied and pasted from the output of the vault.sh command.

Save the changes to host-master.xml.

5.12. In servera and serverb, add the same <vault> definition that was added to the host master in the /opt/domain/configuration/host-slave.xml configuration file before the administration section.

```
<vault>
    <vault-option name="KEYSTORE_URL" value="/opt/jboss-eap-7.0/vault.keystore"/> <vault-option
    name="KEYSTORE_PASSWORD" value="MASK-31x/z0Xn83H4JaL0h5eK/N"/> <vault-option
    name="KEYSTORE_ALIAS" value="vault"/> <vault-option name="SALT"
    value="12345678"/> <vault-option name="ITERATION_COUNT"
    value="50"/> <vault-option name="ENC_FILE_DIR" value="/opt/jboss-
    eap-7.0/"/>
```

```
</vault>
```

5.13.Copy the depot file from workstation to servera and serverb:

```
[student@workstation ~]$ sudo scp /opt/jboss-eap-7.0/VAULT.dat \ servera:/opt/jboss-
eap-7.0 [student@workstation ~]$
sudo scp /opt/jboss-eap-7.0/VAULT.dat \ serverb:/opt/jboss-eap-7.0
```

Change ownership of the VAULT.dat file to the jboss user:

```
[student@servera ~]$ sudo chown jboss:jboss /opt/jboss-eap-7.0/VAULT.dat
```

```
[student@serverb ~]$ sudo chown jboss:jboss /opt/jboss-eap-7.0/VAULT.dat
```

5.14.Start the host in servera and serverb, as well as the master of the host in workstation.

5.15.Replace the password for the bksecurity-ds data source with the store reference to the bksecurity password.

In the administrative console, access the bksecurity-ds data source in the data source subsystem. Click View and uncheck the data source. Use the following CLI command to reload the servers:

```
[domain@172.25.250.254:9990 /] /:reload-servers(blocking=true)
```

Go back to the admin console and click Edit on the security credentials. Update the password to the following and save the changes:

```
${VAULT::bksecurity::password::1}
```

5.16.Reload the servers again for the changes to take effect:

```
[domain@172.25.250.254:9990 /] /:reload-servers(blocking=true)
```

In the administrative console, under the bksecurity-ds overview, click Connection, and click Test Connection to verify that the connection is working.
A popup window should confirm that the connection is valid. If not, use the server log to resolve the issue.

6. Secure the tail.

The credentials for the messaging producer and consumer are hardcoded in the admin user credentials.

Limit access to each queue in the application to ensure that only administrators can send and receive messages. Create a new security constraint for the admin role that has send, receive, and manage privileges. You must use # for the pattern so that values are applied to each queue. Remove the existing guest security configuration.

Use the following command to set the default security domain for the messaging subsystem to be jb248-sd:

```
[domain@172.25.250.254:9990 server=default] /profile=full-ha/subsystem\ =messaging-activemq/server=default:\ write-attribute(name=security-domain,value=jb248-sd)
```

6.1. Navigate to the admin console on workstation and select the full profile ha. Click Messaging - ActiveMQ, select Default, and click Queues/Topics.

6.2. On the left side of the messaging subsystem overview, click Security Settings.

6.3. Click Remove to remove the current permissions for the guest role.

6.4. Click Add to create a new security value. Use the following values to allow all users with the admin role to send, consume, and manage any queue.

- **Pattern: #**
- **Role: admin**
- **Send: Selected**
- **Consume: Selected**
- **Manage: Selected**

Click Save to save the values.

6.5. Using the EAP CLI, use the following command to change the default security domain for the messaging-activemq subsystem to the jb248-sd security domain that was created earlier in the database-backed lab:

```
[domain@172.25.250.254:9990 server=default] /profile=full-ha/subsystem\ =messaging-activemq/server=default:\ write-attribute(name=security-domain,value=jb248-sd)
```

6.6. Reload all servers in the managed domain.

```
[domain@172.25.250.254:9990 /] /:reload-servers(blocking=true)
```

7. Try the tail.

Access the secure messaging client application and send a test message to a queue by logging in as the admin user. Remove the send and receive privileges for this user and verify that they can no longer send a message.

7.1. Deploy the MDB application located at /tmp/messaging-mdb-secure on the Group1 server.

```
[domain@172.25.250.254:9990 /] deploy /tmp/messaging-mdb-secure.jar \ --server-
groups=Group1
```

7.2. Post a test message. Enter the message producer application running on servera.1 at http://172.25.250.10:8080/messaging client-secure using the admin/admin credentials, and complete the form as follows:

- **Message count: 5**
- **Message label: testmsg**
- **Message to send: jms test**

Go to http://172.25.250.10:8080/messaging-client-secure.

Complete the web form and click Send Message to send the message.

The web form should refresh and show the following message in green: Messages sent successfully.

7.3. Check server logs for entries generated by the application of the consumer. The servers servera.1 and serverb.1 must have log entries generated by the consumer application.

There should be registry entries similar to the following in the terminal window serve:

```
[Server:servera.1] 19:54:56,317 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-11 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #3 [testmsg]
[Server:servera.1] 19:54:56,318 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-12 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #5 [testmsg]
[Server:servera.1] 19:54:56,318 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-11 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test [Server:servera.1]
19:54:56,319 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-12 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test [Server:servera.1]
19:54:56,322 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-10 (ActiveMQ-client  global-
threads-904575248)) Message Properties: Copy #1 [testmsg]
[Server:servera.1] 19:54:56,323 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-10 (ActiveMQ-client  global-
threads-904575248)) Message Body: jms test
```

There should be log entries similar to the following in the serverb terminal window:

```
[Server:serverb.1] 19:54:58,282 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-9 (ActiveMQ-client  global-
threads-1005879511)) Message Properties: Copy #4 [testmsg]
[Server:serverb.1] 19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-8 (ActiveMQ-client  global-
threads-1005879511)) Message Properties: Copy #2 [testmsg]
```

```
[Server:serverb.1] 19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-9 (ActiveMQ-client  global-
threads-1005879511)) Message Body: jms test [Server:serverb.1]
 19:54:58,285 INFO [class
  com.redhat.training.messaging.mdb.MessageReceiver] (Thread-8 (ActiveMQ-client  global-
threads-1005879511)) Message Body: jms test
```

7.4. Go back to the administration console and update the security settings of the messaging subsystem for the default server. Click Edit next to the admin role values and remove the shipping and consuming privileges.

7.5. Go back to the app and try sending another test message with the following characteristics:

- **Message count: 5**
- **Message label: authorizationtest**
- **Message to send: authorization test**

7.6. In the server logs for servera, verify that no message was sent, as this user's role did not have the proper authorization to send a message.

```
[Server:servera.1] 19:00:27,100 INFO [stdout] (default task-8)
  javax.jms.JMSSecurityRuntimeException: AMQ119032: User: admin does not have permission='SEND' on
  address jms.queue.TestQueue
```

8. Perform cleaning and grading.

8.1. Press Ctrl+C to stop the domain controller on workstation and the two host controllers in servera and serverb.

8.2. Run the following command to grade the assignment:

```
[student@workstation bin]$ lab securing-lab grade
```

This concludes the lab work.

Summary

In this chapter, you learned the following:

- **Security domains define the way in which transactions are authenticated and authorized. Applications.**
- **EAP has four security domains by default: jboss-ejb policy, jboss-web-policy, other y jaspitest.**
- **A database login module is a security domain backed by a database, which stores usernames and role assignments to ensure authentication for an application.**
- **A security domain has to be backed by an LDAP server, in order to be used for authorization and authentication in an application.**
- **By default, the messaging subsystem uses the other security domain and the ApplicationRealm security domain.**
- **By adjusting the security-settings, users can restrict access and queue authorization and role-based topics.**
- **A password vault is a useful tool for hiding sensitive data in server configuration files, such as database passwords.**
- **The process of saving a password to the vault is accomplished by the following steps:**
  1. **Create a Java keystore.**
  2. **Start the EAP store with the keystore.**
  3. **Keep confidential information in the vault.**
  4. **Update the server configuration to include the vault information.**
  5. **Reference the attribute stored in the server's configuration file.**
- **The vault can be started by running the vault.sh command in the EAP_HOME/ folder bin/.**