Red Hat JBoss Enterprise Application Platform 6.4
How to Setup SSO with Kerberos

How to Setup SSO with Kerberos

Red Hat Customer Content Services

How to Setup SSO with Kerberos

Legal Notice

Abstract

The intent of this guide is to explore the topic of SSO (Single Sign-On) with Kerberos within Red Hat JBoss Enterprise Application Platform 6 as well as provide a practical guide for setting up SSO with Kerberos in JBoss EAP 6. Essentially this guide is providing a deeper dive into what SSO with Kerberos is as well as how to setup and configure it within JBoss EAP 6. Before reading this guide, users should read through the Security Architecture document for Red Hat JBoss Enterprise Application Platform 6 and have a solid understanding of the SSO and Kerberos information presented in that document. This document also makes use of the JBoss EAP 6 CLI interface for performing configuration changes. For more information on using the CLI for both standalone JBoss EAP 6 instances as well as JBoss EAP 6 domains, please consult The Management CLI section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide. When completing this document, readers should have a solid, working understanding of SSO and Kerberos, how it relates to JBoss EAP 6, and how to configure it.

Table of Contents

CHAPTER 1. SSO WITH KERBEROS DEEPER DIVE

1.1. WHAT ARE SSO AND KERBEROS?

A basic background of SSO and Kerberos are provided in the *Single Sign On (SSO)* section of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture document.

1.2. KERBEROS COMPONENTS

Kerberos itself is a network protocol that enables authentication for users of client/server applications through the use of secret-key cryptography. Kerberos is usually used for authenticating desktop users on networks, but through the use of some additional tools, it can be used to authenticate users to web applications and to provide SSO for a set of web applications. This essentially allows users who have already authenticated on their desktop network to seamlessly access secured resources in web applications without having to re-authenticate. This concept is known as Desktop-Based SSO since the user is being authenticated via a desktop-based authentication mechanism, and their authentication token (or ticket) is being used by the web application as well. This differes from other SSO mechanisms such as Browser-Based SSO, which authenticates users and issues tokens all via the browser.

The Kerberos protocol defines several components that it uses in authentication and authorization:

Tickets

A *Ticket* is a form of a security token that Kerberos uses for issuing and making authentication and authorization decisions about principals.

Authentication Service

The *Authentication Service (AS)* challenges principals to log in when they first log into the network. The authentication service is responsible for issuing a *Ticket Granting Ticket (TGT)*, which is needed for authenticating against the *Ticket Granting Service* and subsequent access to secured services and resources.

Ticket Granting Service

The *Ticket Granting Service (TGS)* is responsible for issuing *Service Tickets* and specific session information to principals and the target server they are attempting to access. This is based on the TGT and destination information provided by the principal. This service ticket and session information is then used to establish a connection to the destination and access the desired secured service or resource.

Key Distribution Center

The *Key Distribution Center (KDC)* is the component that houses both the TGS and AS. The KDC, along with the client (principal) and server (secured service), are the three pieces required to perform Kerberos authentication.

Ticket Granting Ticket

A *Ticket Granting Ticket (TGT)* is a type of ticket issued to a principal by the AS. The TGT is granted once a principal successfully authenticates against the AS using their username and password. The TGT is cached locally by the client (principal), but is encrypted such that only the KDC can read it (i.e. it is unreadable by the client). This allows the AS to securely store authorization data and other information in the TGT for use by the TGS and enabling the TGS to make authorization decisions using this data.

Service Ticket

A *Service Ticket (ST)* is a type of ticket issued to a principal by the TGS based on their TGT and the intended destination. The principal provides the TGS with their TGT and the intended destination, and the TGS verifies the principal has access to the destination based on the authorization data in the TGT. If successful, the TGS issues an ST to the client (principal) for both the client as well as the destination server (server containing secured service/resource), which grants the client access to the destination server. The ST, which is cached by the client and readable by both the client and server, also contains session information that allows the client and server to communicate securely.

Note

There is a tight relationship between Kerberos and the DNS settings of the network. For instance, certain assumptions are made when clients access the KDC based on the name of the host it is running on. As a result, it is important that all DNS settings in addition to the Kerberos settings are properly configured to ensure that clients can connect.

1.3. ADDITIONAL COMPONENTS

In addition to the Kerberos components, several other items are needed to enable Kerberos SSO with JBoss EAP 6.

1.3.1. SPNEGO

Simple and Protected GSSAPI Negotiation Mechanism (SPNEGO) provides a mechanism for extending a Kerberos-based Single Sign On (SSO) environment for use in Web applications.

SPNEGO is an authentication method used by a client application to authenticate itself to the server. This technology is used when the client application and server are trying to communicate with each other, but neither are sure of the authentication protocol the other supports. SPNEGO determines the common GSSAPI mechanisms between the client application and the server and then dispatches all further security operations to it.

When an application on a client computer, such as a web browser, attempts to access a protected page on the web server, the server responds that authorization is required. The application then requests a service ticket from the Kerberos KDC. After the ticket is obtained, the application wraps it in a request formatted for SPNEGO, and sends it back to the web application, via the browser. The web container running the deployed web application unpacks the request and attempts to authenticate the ticket. Upon successful authentication, access is granted.

SPNEGO works with all types of Kerberos providers, including the Kerberos service included in Red Hat Enterprise Linux and the Kerberos server which is an integral part of Microsoft Active Directory.

1.3.2. JBoss Negotiation

1.3.2. JBoss Negotiation

JBoss Negotiation is a framework that ships with JBoss EAP 6 that provides an authenticator and JAAS login module to support SPNEGO in JBoss EAP 6. For more information on JAAS login modules, please see the Declarative Security and JAAS and Security Domains sections of the Red Hat JBoss Enterprise Application Platform 6 Security Architecture guide.

Note

When using JBoss Negotiation to secure certain applications (such as REST web services), one or more sessions may be created and remain open for the timeout period (default is 30 minutes) when a client makes a request. This differs from the expected behavior of securing an application via basic authentication, which would leave no open sessions. JBoss Negotiation is implemented to use sessions to maintain the state of the negotiation/connection so the creation of these sessions is expected behavior.

1.4. KERBEROS INTEGRATION

Kerberos is integrated with many operating systems including linux distributions such as Red Hat Enterprise Linux. Kerberos is also an integral part of Microsoft Active Directory and is supported by Red Hat Directory Server and Red Hat IDM.

1.5. HOW DOES KERBEROS PROVIDE SSO FOR RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM?

Kerberos provides Desktop-based SSO by issuing tickets from a KDC for use by the client and server. JBoss EAP 6 can integrate with this existing process by using those same tickets in its own authentication and authorization process. Before trying to understand how JBoss EAP 6 can reuse those tickets, its best to first understand in greater detail how these tickets are issued as well as how authentication and authorization works with Kerberos in Desktop-Based SSO without JBoss EAP 6.

1.5.1. Authentication and Authorization with Kerberos in Desktop-Based SSO

To provide authentication and authorization, Kerberos relies on a third party (the KDC), to provide authentication and authorization decisions for clients (principals) accessing servers (secured resources/services). These decisions happen in three steps:

1. Authentication Exchange
2. Ticket Granting (Authorization) Exchange
3. Accessing the Server

1. Authentication Exchange

When a principal first accesses the network or attempts to access a secured service without a Ticket Granting Ticket (TGT), they are challenged to authenticate against the Authentication Service (AS) with their credentials. The AS validates the users's provided credentials against the configured identity store, and upon successful authentication, the principal is issued a TGT which is cached by

the client. The TGT also contains some session information so future communication between the client and KDC is secured.

2. Ticket Granting (Authorization) Exchange

Once the principal has been issued a TGT, they may attempt to access secured services/resources. The principal sends a request to the Ticket Granting Service (TGS), passing the TGT it was issued by the KDC and requesting a Service Ticket (ST) for a specific destination (secured resource/service). The TGS checkes the TGT provided by the principal and verifies they have proper permissions to access the requested resource. If successful, the TGS issues an ST for the principal to access that specific destination. The TGS also creates session information for both the client as well as the destination server to allow for secure communication between the two. This session information is encrypted separately such that the client and sever can only decrypt its own session information using long-term keys separately provided by the KDC to each (from previous transactions). The TGS then repsonds to the client with the ST which includes the session information for both the client and server.

3. Accessing the Server

Now that the principal has an ST for the secured service as well as a mechanism for secure communication to that server, client may now establish a connection and attempt to access the secured resource. Client starts by passing to the destination server the ST (which also contains the server component of the session information) it received from the TGS for that destination. The server attempts to decrypt the session information passed to it by the client using it's long-term key from the KDC. If it succeeds, the client has been successfully authenticated to the server and the server is also considered authenticated to the client. At this point, trust has been established and secured communication between the client and server may proceed.

Note

Despite the fact that unauthorized principals cannot actually use a TGT, a principal will only be issued a TGT after they first successfully authenticate with the AS. Not only does this ensure that only properly authorized principals are ever issued a TGT, it also reduces the ability for unauthorized third parties to obtain TGTs in an attempt to compromise and/or exploit them (e.g. via offline dictionary/brute-force attacks).

1.5.2. Kerberos and Red Hat JBoss Enterprise Application Platform 6

JBoss EAP 6 can integrate with an existing Kerberos Desktop-Based SSO environment to allow for those same tickets to provide access to web applications hosted on JBoss EAP 6 instances. In a typical setup, an JBoss EAP 6 instance would be configured to have Kerberos and SPNEGO security domains. An application, configured to use those security domains along with JBoss Negotiation, is deployed to that JBoss EAP 6 instance. A user logs in to a desktop, which is governed by the Kerberos, and completes an authentication exchange with the KDC. The user then attempts to access a secured resource in the deployed application on that JBoss EAP 6 instance directly via a web browser. JBoss EAP 6 responds that authorization is required to access the secured resource. The web browser obtains the user's TGT ticket and then performs the ticket granting (authorization) exchange with the KDC to validate the user and obtain a service ticket. Once the ST is returned to the browser, it wraps the ST in a request formatted for SPNEGO, and sends it back to the Web application running on JBoss EAP 6. JBoss EAP 6 then unpacks the SPNEGO request and performs the authentication using the configured security domains and JBoss Negotiation. If the authentication successeds, the user is granted access to the secured resource.

CHAPTER 2. HOW TO SETUP SSO FOR RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6 WITH KERBEROS

This section covers how to configure JBoss EAP 6 and the deployed applications to use Kerberos for SSO.

2.1. COMPONENTS

The following components are needed for setting SSO for JBoss EAP 6 with Kerberos:

- A properly configured Kerberos environment
- A JBoss EAP 6 instance
- A web application

2.1.1. JBoss Negotiation Toolkit

The *JBoss Negotiation Toolkit* is a debugging tool which is available for download from https://community.jboss.org/servlet/JiveServlet/download/16876-2-34629/jboss-negotiation-toolkit.war. It is provided as an extra tool to help users to debug and test the authentication mechanisms before introducing an application into production. It is an unsupported tool, but is considered to be very helpful, as SPNEGO can be difficult to configure for web applications.

2.1.2. Kerberos Environment

As discussed in a previous section, Kerberos relies on a third party (the KDC), to provide authentication and authorization decisions. This also requires clients (e.g. browsers) and their host to be properly configured to authenticate with the KDC. This guide is primarily focused on how to configure JBoss EAP 6 and its hosted web applications so configuring the KDC and Kerberos domain are not in the scope of this document.

Note

The subsequent sections assume a KDC and Kerberos domain have already been set up and properly configured.

2.1.3. Differences from Configuring Previous Versions Red Hat JBoss Enterprise Application Platform

There are a few noticeable differences between JBoss EAP 6 and earlier versions:

- Security domains are configured for each profile of a managed domain, or for each standalone server. They are not part of the deployment itself. The security domain a deployment should use is named in the deployment's **`jboss-web.xml`** or **`jboss-ejb3.xml`** file.
- Security properties are configured as part of a security domain. They are not part of the deployment.
- Authenticators can no longer be overriden as part of a deployment. However, a

NegotiationAuthenticator valve can be added to the **`jboss-web.xml`** descriptor to achieve the same effect. The valve still requires the *<security-constraint>* and *<login-config>* elements to be defined in the **`web.xml`**. These are used to decide which resources are secured. However, the chosen *auth-method* will be overridden by the NegotiationAuthenticator valve in the **`jboss-web.xml`**.

- The *CODE* attributes in security domains now use a simple name instead of a fully-qualified class name. The below table shows the mappings between the classes used for JBoss Negotiation, and their classes.

Simple Name	Class Name	Purpose
Kerberos	com.sun.security.auth.module.Krb5LoginModule	Kerberos login module when using the Oracle JDK
Kerberos	com.ibm.security.auth.module.Krb5LoginModule	Kerberos login module when using the IBM JDK
SPNEGO	org.jboss.security.negotiation.spnego.SPNEGOLoginModule	The mechanism which enables your Web applications to authenticate to your Kerberos authentication server.
AdvancedLdap	org.jboss.security.negotiation.AdvancedLdapLoginModule	Used with LDAP servers other than Microsoft Active Directory.
AdvancedAdLdap	org.jboss.security.negotiation.AdvancedADLoginModule	Used with Microsoft Active Directory LDAP servers.

2.1.4. Configuring the Red Hat JBoss Enterprise Application Platform 6 Instance

JBoss EAP 6 comes with all the components necessary to use Kerberos (via SPNEGO and JBoss Negotiation) for SSO with deployed applications, but the following configuration changes need to be made:

1. Configure Server Identity (Host) Security Domain
2. Configure Web Application Security Domain
3. Configure Relevant System Properties

Note

The CLI commands shown below were done assuming a standalone instance of JBoss EAP 6. For more details on using the CLI with JBoss EAP 6 domains, please consult *The Management CLI* section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide.

2.1.4.1. 1. Configuring Server Identity (Host) Security Domain

This security domain authenticates the container itself to the KDC. It needs to use a login module which accepts a static login mechanism since a real user is not involved in this connection. The below example uses a static principal and references a keytab file which contains the credential.

Example CLI for Creating a Server Identity Security Domain

```
/subsystem=security/security-domain=host:add(cache-type=default)
```

```
/subsystem=security/security-domain=host/authentication=classic:add
```

```
/subsystem=security/security-domain=host/authentication=classic/login-
module=Kerberos:add( \
  code=Kerberos, \
  flag=required, \
  module-options=[ \
    ("storeKey"=>"true"), \
    ("refreshKrb5Config"=>"true"), \
    ("useKeyTab"=>"true"), \
    ("principal"=>"host/testserver@MY_REALM"), \
    ("keyTab"=>"/home/username/service.keytab"), \
    ("doNotPrompt"=>"true"), \
    ("debug"=>"false") \
  ])
```

```
reload
```

Resulting XML

```
<security-domain name="host" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="storeKey" value="true"/>
      <module-option name="useKeyTab" value="true"/>
      <module-option name="principal" value="host/testserver@MY_REALM"/>
      <module-option name="keyTab"
value="/home/username/service.keytab"/>
      <module-option name="doNotPrompt" value="true"/>
```

```
      <module-option name="debug" value="false"/>
    </login-module>
  </authentication>
</security-domain>
```

If using the IBM JDK, options for Kerberos module are different. The **jboss.security.disable.secdomain.option** system property must be set to **true** (see Configure Relevant System Properties). In addition the login module should be updated to the following:

IBM JDK Example

```
<security-domain name="host" cache-type="default">
  <authentication>
    <login-module code="Kerberos" flag="required">
      <module-option name="principal" value="HTTP/testserver@MY_REALM"/>
      <module-option name="credsType" value="acceptor"/>
      <module-option name="useKeytab" value="file:///root/keytab"/>
    </login-module>
  </authentication>
</security-domain>
```

For a complete list of options for configuring the *Kerberos* login module, refer to the Red Hat JBoss Enterprise Application Platform 6 Security Guide.

2.1.4.2. 2. Configuring Web Application Security Domain

The web application security domain is used to authenticate the individual user to the KDC. There needs to be at least one login module to authenticate the user, and a way to search for the roles to apply to the user. The latter can be accomplished in many different ways (e.g. adding a **<mapping>** that manually maps users to roles, adding a second login module for mapping users to roles, etc).

The following shows an example web application security domain.

Example CLI for Creating a Server Identity Security Domain

```
/subsystem=security/security-domain=app-spnego:add(cache-type=default)
```

```
/subsystem=security/security-domain=app-spnego/authentication=classic:add
```

```
/subsystem=security/security-domain=app-
spnego/authentication=classic/login-module=SPNEGO:add( \
  code=SPNEGO, \
  flag=required, \
  module-options=[ \
    ("serverSecurityDomain"=>"host") \
  ])
```

```
reload
```

Resulting XML

```
<security-domain name="app-spnego" cache-type="default">
  <authentication>
    <!-- Check the username and password -->
    <login-module code="SPNEGO"  flag="required">
      <module-option name="serverSecurityDomain" value="host"/>
    </login-module>
    <!-- Second login module to search for roles -->
  </authentication>
  <!-- Alternatively, a 'mapping' element may be added instead of a
second login module to map users to roles-->
</security-domain>
```

For a complete list of options for configuring the *SPNEGO* login module, refer to the Red Hat JBoss Enterprise Application Platform 6 Security Guide.

2.1.4.3. 3. Configuring Relevant System Properties

JBoss EAP 6 offers the ability to configure system properties related to connecting to Kerberos servers. Depending on the KDC, Kerberos Domain, and network configuration, the below system properties may or may not be required.

```
<system-properties>
  <property name="java.security.krb5.kdc" value="mykdc.mydomain"/>
  <property name="java.security.krb5.realm" value="MY_REALM"/>
  <property name="java.security.krb5.conf" value="/path/to/krb5.conf"/>
  <property name="jboss.security.disable.secdomain.option" value="true"/>
  <property name="java.security.krb5.debug" value="false"/>
</system-properties>
```

Value	Description
java.security.krb5.kdc	The host name of the KDC
java.security.krb5.realm	The name of the realm
java.security.krb5.conf	The path to the configuration **`krb5.conf`** file
jboss.security.disable.secdomain.option	When set to **true**, disables automatic adding of *jboss.security.security_domain* login module option to login modules declared in the security domain. Must be set to **true** when using the IBM JDK
java.security.krb5.debug	If **true**, debugging mode will be enabled

Note

By default, each login module defined in a security domain has the *jboss.security.security_domain* module option added to it automatically. This option causes problems with login modules which check to make sure that only known options are defined. The IBM Kerberos login module, *com.ibm.security.auth.module.Krb5LoginModule* is one of these. This behavior of adding this module option can disabled by setting the *jboss.security.disable.secdomain.option* system property to **true** when starting JBoss EAP 6. This can be accomplished by configuring the *<system-properties>* (via the Management CLI or Management Console) or by adding **`-Djboss.security.disable.secdomain.option=true`** to the start-up parameters.

For more information about configuring system properties, refer to the Configure System Properties Using the Management CLI section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide.

2.1.5. Configuring the Web Application

Once the security domains have been configured, the web application must be configured to use those security domains in order to enable Kerberos authentication. The following updates must be made the application:

1. Configure the web.xml to Use the SPNEGO Authentication Method
2. Configure the jboss-web.xml to Use the Configured Security Domain
3. Add the JBoss Negotiation Dependencies to the Deployment

Once the application changes have been made, it may be deployed to the JBoss EAP 6 instance and begin using Kerberos for authentication.

2.1.5.1. 1. Configuring the web.xml to Use the SPNEGO Authentication Method

The **`web.xml`** file should contain the following:

- A **`<security-constraint>`** with a **`<web-resource-collection>`** containing a **`<url-pattern>`** that maps to the URL pattern of the secured area. Optionally, **`<security-constraint>`** may also contain an **`<auth-constraint>`** stipulating the allowed roles.
- If any roles were specified in the **`<auth-constraint>`**, those roles should be defined in a **`<security-role>`**.

The **`<security-constraint>`** and **`<security-role>`** elements enable administrators to setup restricted or unrestricted areas based on URL patterns and roles. This allows resources to be secured or unsecured.

Example `web.xml` file:

```
<web-app>
  <display-name>App1</display-name>
  <description>App1</description>
```

```
  <!-- Define a security constraint that requires the All role to access
resources -->
  <security-constraint>
    <display-name>Security Constraint on Conversation</display-name>
    <web-resource-collection>
      <web-resource-name>exampleWebApp</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>All</role-name>
    </auth-constraint>
  </security-constraint>
<!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>SPNEGO</auth-method>
    <realm-name>SPNEGO</realm-name>
  </login-config>
<!-- Security roles referenced by this web application -->
  <security-role>
    <description>Role required to log in to the Application</description>
    <role-name>All</role-name>
  </security-role>
</web-app>
```

2.1.5.2. 2. Configuring the jboss-web.xml to Use the Configured Security Domain

The **`jboss-web.xml`** file should have the following:

- A **`<security-domain>`** to specify which security domain to use for authentication and authorization.
- A **`<valve>`** configured to use the NegotiationAuthenticator valve class (i.e. *org.jboss.security.negotiation.NegotiationAuthenticator*)
- **Optional**: A <jacc-star-role-allow>, which enables the use of the asterisk (*) character in role-name element in **`web.xml`** to match multiple role names.

An example `jboss-web.xml` file:

```
<jboss-web>
  <security-domain>app-spnego</security-domain>
  <valve>
    <class-
name>org.jboss.security.negotiation.NegotiationAuthenticator</class-name>
  </valve>
  <jacc-star-role-allow>true</jacc-star-role-allow>
</jboss-web>
```

2.1.5.3. 3. Adding the JBoss Negotiation Dependencies to the Deployment

A web application using SPNEGO and JBoss Negotiation requires a dependency to be defined in **`jboss-deployment-structure.xml`** so that the JBoss Negotiation classes can be located. Since JBoss EAP 6 provides all necessary JBoss Negotiation and related classes, the application just needs to declare them as dependencies to use them.

Using jboss-deployment-structure.xml to declare dependencies

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.jboss.security.negotiation"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

Alternatively, this dependency may be defined in a **`META-INF/MANIFEST.MF`** file instead:

Using META-INF/MANIFEST.MF to declare dependencies

```
Manifest-Version: 1.0
Build-Jdk: 1.6.0_24
Dependencies: org.jboss.security.negotiation
```

2.2. ADDITIONAL CONSIDERATIONS FOR ACTIVE DIRECTORY

This section describes how to configure the accounts required for JBoss Negotiation to be used when JBoss EAP 6 is running on a Microsoft Windows server, which is a part of the Active Directory domain.

In this section, the hostname that is used to access the server as is referred to as **HOSTNAME**, realm is referred to as **REALM**, domain is referred to as **DOMAIN**, and the server hosting the JBoss EAP 6 instance is referred to as **MACHINE_NAME**.

2.2.1. Configure JBoss Negotiation for Microsoft Windows Domain

2.2.1.1. 1. Clear Existing Service Principal Mappings

On a Microsoft Windows network some mappings are created automatically. Delete the automatically created mappings to map the identity of the server to the service principal for negotiation to take place correctly. The mapping enables the web browser on the client computer to trust the server and attempt SPNEGO. The client computer verifies with the domain controller for a mapping in the form of HTTP/**HOSTNAME**.

The following are the steps to delete the existing mappings:

List the mapping registered with the domain for the computer using the command:

```
setspn -L MACHINE_NAME
```

Delete the existing mappings using the commands:

```
setspn -D HTTP/HOST_NAME MACHINE_NAME
```

```
setspn -D host/HOSTNAME MACHINE_NAME
```

2.2.1.2. 2. Create a host user account.

Note

Ensure the host user name is different from the **MACHINE_NAME**.

In the rest of the section the host user name is referred to as **USER_NAME**.

2.2.1.3. 3. Define the mapping between the USER_NAME and HOSTNAME.

Run the following command to configure the Service Principal Mapping:

```
ktpass -princ HTTP/HOSTNAME@REALM -pass * -mapuser DOMAIN\USER_NAME
```

Enter the password for the user name when prompted.

Note

Reset the password for the user name as it is a prerequisite for exporting the keytab.

Verify the mapping by running the following command, setspn -L **USER_NAME**

2.2.1.4. 4. Export the keytab of the user to the server on which EAP JBoss is installed.

Run the following command to export the keytab:

```
ktab -k service.keytab -a HTTP/HOSTNAME@REALM
```

Note

This command exports the ticket for the HTTP/**HOSTNAME** principal to the keytab service.keytab, which is used to configure the host security domain on JBoss EAP 6.

2.2.1.5. 5. Define the principal within the security domain

The principal can be defined or updated in the security domain as follows:

```
<module-option name="principal">HTTP/HOSTNAME@REALM</module-option>
```

CHAPTER 3. ADDITIONAL FEATURES

3.1. ADDING A FORM LOGIN AS A FALLBACK

JBoss EAP 6 and applications deployed to it can also configure a FORM login authentication mechanism to use as a fallback. This allows applications to present a login page for authentication in cases where a Keberos/SPNEGO tokens are not present. This authentication happens independent of the Kerberos authentication. As a result, depending on how the FORM login fallback is configured, users may require separate credentials to authenticate via this method.

The following steps are required to configure FORM login as a fallback:

1. Configure Red Hat JBoss Enterprise Application Platform 6 and the web application to use Kerberos and SPNEGO
2. Update the Security Domain for Fallback Authentication
3. Add the login and error pages
4. Modify the web.xml

Note

The fallback to FORM logic is available in the case when no SPNEGO (or NTLM) tokens are present or SPNEGO token is present but from another KDC.

3.1.1. 1. Configure Red Hat JBoss Enterprise Application Platform 6 and the Web Application to Use Kerberos and SPNEGO

Please refer to the previous section for the steps required to configure JBoss EAP 6 and web applications to use Kerberos and SPNEGO for authentication and authorization.

3.1.2. 2. Update the Security Domain for Fallback Authentication

The web application security domain must be configured to support a fallback login mechanism. This requires the following steps:

- Add a new security domain to server as a fallback authentication method.
- Add a *usernamePasswordDomain* module option to the web application security domain that points to the fallback domain.

Example Security Domain Configured with a Fallback Security Domain

```
<security-domain name="app-spnego" cache-type="default">
  <authentication>
    <login-module code="SPNEGO" flag="requisite">
      <module-option name="password-stacking" value="useFirstPass"/>
      <module-option name="serverSecurityDomain" value="host"/>
      <module-option name="usernamePasswordDomain" value="app-fallback"/>
    </login-module>
```

```
    <!--login module for mapping roles -->
    <login-module code="UsersRoles" flag="required">
      <module-option name="password-stacking" value="useFirstPass"/>
      <module-option name="usersProperties"
        value="file:${jboss.server.config.dir}/users.properties"/>
      <module-option name="rolesProperties"
        value="file:${jboss.server.config.dir}/roles.properties"/>
    </login-module>
  </authentication>
</security-domain>
<security-domain name="app-fallback" cache-type="default">
  <authentication>
    <login-module code="UsersRoles" flag="required">
      <module-option name="usersProperties"
        value="file:${jboss.server.config.dir}/fallback-
users.properties"/>
      <module-option name="rolesProperties"
        value="file:${jboss.server.config.dir}/fallback-
roles.properties"/>
    </login-module>
  </authentication>
</security-domain>
```

3.1.3. 3. Add the Login and Error Pages

To use FORM login, a login and error page are required. These files are added to web application and are used in the authentication process.

Example `login.jsp` file:

```
<html>
<head></head>
<body>
  <form id="login_form" name="login_form" method="post"
  action="j_security_check" enctype="application/x-www-form-urlencoded">
    <center>
   <p>Please login to proceed.</p>
 </center>
 <div style="margin-left: 15px;">
   <p>
     <label for="username">Username</label>
  <br />
  <input id="username" type="text" name="j_username"/>
   </p>
   <p>
     <label for="password">Password</label>
  <br />
  <input id="password" type="password" name="j_password" value=""/>
   </p>
   <center>
     <input id="submit" type="submit" name="submit" value="Login"/>
   </center>
```

```
 </div>
  </form>
</body>
</html>
```

Example `error.jsp` file:

```
<html>
<head></head>
<body>
  <p>Login failed, please go back and try again.</p>
</body>
</html>
```

3.1.4. 4. Modify the web.xml

After adding the login and error pages to the web application, the **`web.xml`** must be updated to use these files for FORM login. A **`<form-login-config>`** element is added to **`<login-config>`** and the paths to the the login and error pages are specified as **`<form-login-page>`** and **`<form-error-page>`** elements.

Example updated `web.xml` file:

```
<web-app>
  <display-name>App1</display-name>
  <description>App1</description>
  <!-- Define a security constraint that requires the All role to access
resources -->
  <security-constraint>
    <display-name>Security Constraint on Conversation</display-name>
    <web-resource-collection>
      <web-resource-name>examplesWebApp</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>All</role-name>
    </auth-constraint>
  </security-constraint>
<!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>SPNEGO</auth-method>
    <realm-name>SPNEGO</realm-name>
    <form-login-config>
      <form-login-page>/login.jsp</form-login-page>
      <form-error-page>/error.jsp</form-error-page>
    </form-login-config>
  </login-config>
<!-- Security roles referenced by this web application -->
  <security-role>
    <description> role required to log in to the
```

```
Application</description>
    <role-name>All</role-name>
  </security-role>
</web-app>
```

3.2. SECURING THE MANAGEMENT INTERFACES WITH KERBEROS

In addition to providing Kerberos authentication in security domains, JBoss EAP 6 also provides the ability to secure the management interfaces using Kerberos. To enable Kerberos authentication on the management interfaces, the following steps must be performed:

1. Enable Relavant System Properties
2. Add the Kerberos Server Identity to the Security Realm
3. Update Authentication Method in the Security Realm

Note

The CLI commands shown below were done assuming a standalone instance of JBoss EAP 6. For more details on using the CLI with JBoss EAP 6 domains, please consult *The Management CLI* section of the Red Hat JBoss Enterprise Application Platform 6 Administration and Configuration Guide.

3.2.1. 1. Enable Relavant System Properties

As discussed in a previous section, enable any needed JBoss EAP 6 system properties for connecting to the Kerberos server.

3.2.2. 2. Adding the Kerberos Server Identity to the Security Realm

Before Kerberos authentication can be used in a security realm, a connection to a Kerberos server must be added. The following example shows how to add a Kerberos server identity to the existing Management Realm.

Example CLI for Adding a Server Identity to a Security Realm

```
/core-service=management/security-realm=ManagementRealm/server-
identity=kerberos:add
```

```
/core-service=management/security-realm=ManagementRealm/server-
identity=kerberos/ \
keytab=host\/testserver@MY_REALM:add( \
path=/home\/username\/service.keytab, \
debug=true)
```

```
reload
```

Resulting XML

```
<security-realm name="ManagementRealm">
    <server-identities>
        <kerberos>
            <keytab principal="host/testserver@MY_REALM"
                    path="/home/username/service.keytab"
                    debug="true"/>
        </kerberos>
    </server-identities>
    ...
</security-realm>
```

3.2.3. 3. Updating Authentication Method in the Security Realm

Once the Kerberos server identity has been properly configured, the authentication method in the security realm needs to be updated to use it.

Example CLI for Adding Kerberos Authentication to a Security Realm

```
/core-service=management/security-
realm=ManagementRealm/authentication=kerberos:add
```

```
reload
```

Resulting XML

```
<security-realm name="ManagementRealm">
    <server-identities>
        <kerberos>
            <keytab principal="host/testserver@MY_REALM"
                    path="/home/username/service.keytab"
                    debug="true"/>
        </kerberos>
    </server-identities>
    <authentication>
        <local default-user="$local" skip-group-loading="true"/>
        <kerberos/>
        <properties path="mgmt-users.properties" relative-
to="jboss.server.config.dir"/>
    </authentication>
    ...
</security-realm>
```