User Guide for JBoss Negotiation

A Guide for Administrators

Darran A. Lofthouse

ISBN:

Publication date:

User Guide for JBoss Negotiation: A Guide for Administrators

by Darran A. Lofthouse

Target Audience

This guide is aimed at administrators who want to use the JBoss Negotiation authenticator to implement silent SPNEGO authentication.

Preface

JBoss Negotiation is a project of JBoss Security in the JEMS product suite.

The initial development of JBoss Negotiation is now complete so JBoss Negotiation 2.0.3.GA is the first GA release.

Use of the JBoss Negotiation library can be discussed in the security users forum. *Security and JAAS/JBoss* [http://www.jboss.org]

Bugs and feature requests can be reported in Jira under the SECURITY project. *JIRA SECURITY* [http://jira.jboss.com/jira/browse/SECURITY] For any issues raised be sure to set the component to 'Negotiation'.

The source code for the library is held within SVN: - *security-negotiation* [http://anonsvn.jboss.org/repos/jbossas/projects/security/security-negotiation]

Authors:

- Darran Lofthouse - Negotiation authenticator developer.

Introduction to JBoss Negotiation

The JBoss Negotiation project provides a set of components to bring ' **S** imple and **P** rotected GSSAPI **Nego** tiation Mechanism' or SPNEGO to JBoss.

SPNEGO authentication allows a user already authenticated to a **K** erberos **D** omain **C** ontroller / KDC to silently authenticate to remote services without being prompted for further usernames and passwords. In addition to this the users credentials can be delegate to the remote system allowing the remote system to contact further systems as the user.

Many web browsers provide support for SPNEGO authentication, this document focuses on Microsoft Internet Explorer™ and Mozilla Firefox. There are also a number of kerberos domain controllers available, this documentation focusses on Microsoft Active Directory™ and the MIT KDC implementation when included in FreeIPA. Contributions for documentation on other web browsers and KDCs would be welcome.

1. Components

The JBoss Negotiation project provides the following components: -

- SPNEGO Authenticator and Login Module

  The authentication process is handled by a JBoss Web Authenticator that and a JAAS login module, this combination achieves the integration with JBoss security.

- Negotiation Toolkit

  This is a couple of utilities and a web application that can be used to test various aspects of your negotiation configuration to enable you to verify that the required steps are working correctly and to debug where failures may be occuring.

2. General Authentication Process

When working with the JBoss login modules and the existing authentication mechanisms work by asking the user to authenticate themseves by the client sending thier credentials to the server and then the login module verifying the credentials against either a local store of credentials or against a store on a remote repository such as a database server or a LDAP server.

The SPNEGO authentication mechansim is slightly different.

- **Server Authentication** - First the application server itself authenticates against the KDC and obtains it's own ticket.
- **Client Authentication** - After the server prompt the client to authenticate the client responds with a SPNEGO token, the server then makes use of it's own ticket to decode the clients ticked and respond to the client.

  This process can take a couple of round trips for the client to authenticate against the server.
- **Mututal Authentication** - If this is required it is even possible for the client to request that the server authenticates itself against the client.
- **Credential Delegation** - A client can also be configured so that the credentials used for authentication can be delegated to the server, this means that the application server can then go on and call other systems on behalf of the calling client.

3. Pre-requisits

The installation of this module requires the externalised authenticator capability of JBoss which was added from JBoss 4.0.5.GA, these instructions have been prepared against JBoss AS 4.2.3.GA and JBoss AS 5.0.0.GA.

http://wiki.jboss.org/wiki/ExternalizeTomcatAuthenticators [http://www.jboss.org/community/docs/DOC-9729]

General Installation

1. Introduction

This section of the document describes the general installation process of the negotiation module, the following chapters in this guide describe the specific configuration requirements for the KDC and the web browser.

This documentation covers two topics.

- Installation and configuration of the library itself.
- Installation of the negotiation toolkit to test the installation.

It is recommended that you use the negotiation toolkit to test that the security settings are correctly working before attempting to secure your own web application, this way you can eliminate if any problems are specific to your web application and also use the toolkit to obtain additional debug information.

Also see *Chapter 8, Troubleshooting*

2. Installation

2.1. Authenticator Installation

The authenticator is contained within a single jar ' `jboss-negotiation-2.0.3.GA.jar` ', this jar should be placed in the following location - `{jboss.home}/server/{configuration}/lib/`

The locations to define the authenticator are slightly different between JBoss AS 4.2.x and JBoss AS 5.0.x.

2.1.1. JBoss AS 4.2.x

After copying the jar to the above location you will need to add the authenticator itself to the following descriptor - `{jboss.home}/server/{configuration}/deploy/jboss-web.deployer/META-INF/jboss-service.xml`

Within this descriptor you should see a set of authenticators, to add negotiation you should add the following entry: -

```


<java:property>
  <java:key>SPNEGO</java:key>
```

```
 <java:value>org.jboss.security.negotiation.NegotiationAuthenticator</
java:value>
</java:property>
```

2.1.2. JBoss AS 5.0.x

After copying the jar to the above location you will need to add the authenticator itself to the following descriptor - `{jboss.home}/server/{configuration}/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml`

Within this descriptor you should see a set of authenticators defined using a property called "authenticators", you should add the following entry: -

```
<entry>
  <key>SPNEGO</key>

 <value>org.jboss.security.negotiation.NegotiationAuthenticator</
value>
</entry>
```

Tip

Ensure that there is no white space around the classname as this can cause the deployment to fail.

Warning

If you have been using the Beta releases then you may have been using the authenticator called `org.jboss.security.negotiation.spnego.SPNEGOAuthenticator` this authenticator is now deprecated and will be removed in a future release so you should switch to the NegotiationAuthenticator as shown above.

The key can be any value you choose, however using SPNEGO is recommended to be consistent with the rest of this document, this is also required by the Negotiation Toolkit.

2.2. Realm Properties

If you are running your JBoss installation on a host which is already configured to authenticate against a Kerberos KDC then you can skip this step, however if the host is not already configured against a KDC or if you need JBoss to authenticate against a different KDC a couple of system properties need to be set so that JBoss can identify the correct realm and kdc.

The two properties that need to be set are.

- java.security.krb5.realm - This is the Kerberos realm to authenticate against.
- java.security.krb5.kdc - This is the hostname of the KDC itself.

Both of these properties are specific to the JVM so further information is available from *http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/tutorials/KerberosReq.html* [http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/tutorials/KerberosReq.html]

2.2.1. Command Line

The easiest way to set the properties is to pass them to JBoss on the command line when you start the server e.g.

```
./run.sh -Djava.security.krb5.realm=KERBEROS.JBOSS.ORG
-Djava.security.krb5.kdc=kerberos.security.jboss.org
```

2.2.2. System Properties Service

JBoss also make a properties service available which will allow you to define these properties in a descriptor and the properties service will set them as JBoss starts, the only requirement is that these properties are set before the first authentication attempt - JBoss does not allow incomming HTTP connections until the server is completely started so this is not a problem.

The properties service is documented in the Wiki at *http://wiki.jboss.org/wiki/PropertiesService* [http://wiki.jboss.org/wiki/PropertiesService]

There is already a deployment of the properties service that you can add your properties to, this is in the following descriptor: -

{jboss.home}//server/{configuration}/deploy/properties-service.xml

Add the following attribute to the 'jboss:type=Service,name=SystemProperties' MBean to set the properties: -

```
<attribute name="Properties">
 java.security.krb5.kdc=kerberos.security.jboss.org
 java.security.krb5.realm=KERBEROS.JBOSS.ORG
</attribute>
```

2.2.3. Multiple KDCs

If in addition to your master KDC if you also have one or more slave KDCs then it is also possible list these using the java.security.krb5.kdc system property, this will allow an alternative to be used if it is not possible to contact the master KDC.

This is a feature of Java GSS *http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/jgss-features.html* [http://java.sun.com/j2se/1.5.0/docs/guide/security/jgss/jgss-features.html] The KDCs should be delimited using a colon (:) e.g.

```
          ./run.sh

 -Djava.security.krb5.realm=KERBEROS.JBOSS.ORG:SLAVE_KDC.JBOSS.ORG
          -Djava.security.krb5.kdc=kerberos.security.jboss.org
```

2.3. Host Security Domain

The application server requires a security domain that it can use to first authenticate against the KDC, in order to configure this a keytab will be required for the principal that represents the application server. The following chapters will cover the details of setting up a service host and obtaining the keytab from the KDC but the general requirements to configure the security domain in JBoss are the same.

Below is an example host security domain: -

```
<application-policy name="host">
   <authentication>
      <login-module
 code="com.sun.security.auth.module.Krb5LoginModule"
         flag="required">
         <module-option name="storeKey">true</module-option>
         <module-option name="useKeyTab">true</module-option>
         <module-option
 name="principal">host/testserver@KERBEROS.JBOSS.ORG</module-
option>
         <module-option
 name="keyTab">/home/jboss_user/testserver.keytab</module-option>
         <module-option name="doNotPrompt">true</module-option>
         <module-option name="debug">true</module-option>
      </login-module>
   </authentication>
</application-policy>
```

The selected name of the security domain is not important, later it will be possible which security domain to use to authenticate the server.

The following options are required.

- storeKey - cache the private key within the Subject.
- useKeyTab - Specify that the key will be loaded from a keyTab
- principal - The full name of the principal to obtain from the keytab
- keyTab - The full path to the keytab containing the servers key
- doNotPrompt - As this is a server disable prompting for the servers password
- debug - enable logging if additional debug information

Note

Once everything is working you may want to set debug to false as it logs to STDOUT.

Caution

The Krb5LoginModule does have an option to be configured to use a local credentials cache, this should be avoided as it is incompatible with the storKey option which is required for SPNEGO negotiation.

2.4. Application Security Domain

The application also requires it's own security domain to be defined with a login module to work in connection with the NegotiationAuthenticator and a second login module to load the roles of the authenticated user.

An example security domain is shown below.

```
<application-policy name="SPNEGO">
   <authentication>
      <login-module

 code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
         flag="requisite">
         <module-option
 name="password-stacking">useFirstPass</module-option>
         <module-option
 name="serverSecurityDomain">host</module-option>
      </login-module>
      <login-module
         code="org.jboss.security.auth.spi.UsersRolesLoginModule"
         flag="required">
         <module-option
 name="password-stacking">useFirstPass</module-option>
         <module-option
 name="usersProperties">props/spnego-users.properties</module-
option>
         <module-option
 name="rolesProperties">props/spnego-roles.properties</module-
option>
      </login-module>
   </authentication>
</application-policy>
```

The SPNEGOLoginModule requires the following two options.

- password-stacking - Setting this to useFirstPass allows a second module to load the roles.
- serverSecurityDomain - The security domain of the application server as previously defined.

The second login module is used to load the users roles after the authentication has already taken place by the previous login module.

Please see the following documentation for further information on the JBoss login modules and more specifically the UsersRolesLoginModule: -

JBoss 4.2.2.GA Configuration Guide
[http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/

Server_Configuration_Guide/beta422/html/
Security_on_JBoss.html#Defining_Security_Domains-Using_JBoss_Login_Modules]

If the application security domain is defined within the `{jboss.home}/server/{configuration}/conf/login-config.xml` descriptor the properties files should be located in `{jboss.home}/server/{configuration}/conf/props` .

The spnego-users.properties file should be empty as the previous login module is handling the authentication, the spnego-roles.properties should be as follows: -

```
# A roles.properties file for use with the UsersRolesLoginModule
darranl@KERBEROS.JBOSS.ORG=Users
```

Before the '=' is the fully qualified name of the user, after the '=' is a comma separated list of the users roles.

The JBoss Negotiation project also includes a new login module that can be used to obtain the roles from LDAP after the SPNEGO login module see ' *Appendix A, Advanced LDAP Login Module*

2.5. DNS

So that the web browsers communicating with the application server can trust the application server when it prompts for the negotiation to take place it is important that the IP address of the server running JBoss is mapped correctly.

In these examples the example KDC realm is 'KERBEROS.JBOSS.ORG' and the server hosting JBoss is 'testserver', the IP address of the server should be resolvable as 'testserver.kerberos.jboss.org'.

This configuration change can either be made on your actual DNS server or can be made locally on the client machine.

2.6. Web Application

Once the server is configured your web application also needs to be configured to make use of SPNEGO negotiation as the authentication mechanism.

First the web application needs to be configured to use the application security domain as defined previously: -

```
<jboss-web>
```

```
    <security-domain>java:/jaas/SPNEGO</security-domain>
  </jboss-web>
```

Next the login-config needs to be configured to use the SPNEGO authenticator defined previously: -

```
<login-config>
  <auth-method>SPNEGO</auth-method>
  <realm-name>SPNEGO</realm-name>
</login-config>
```

In the above example it is the auth-method that maps to the key used for the authenticator previously.

3. Negotiation Toolkit

The Negotiation Toolkit is a web application that you can deploy to your JBoss installation to test various aspects of your SPNEGO configuration without adding the complications of getting your own applications to work at the same time. Once the Negotiation Toolkit demonstrates that negotiation is occurring without any problems then you can move to secure your own web application.

The Negotiation Toolkit is distributed within a war called 'jboss-negotiation-toolkit.war', this war should be placed in the `{jboss.home}/server/{configuration}/deploy` folder to deploy.

The Negotiation Toolkit was assumes that the authenticator was defined with a key of 'SPNEGO' and with an application security domain of 'SPNEGO', if either of these are different in your JBoss installation you can deploy the war as an exploded deployment and modify as necessary.

Once deployed it should be possible to access the Negotiation Toolkit web application at the following URL assuming your DNS entry is correct as described previously. *http://testserver.kerberos.jboss.org:8080/jboss-negotiation-toolkit* [http://testserver.kerberos.jboss.org:8080/jboss-negotiation-toolkit]

3.1. Front Page

The main page for the negotion toolkit contains links to the various utilities within the toolkit that can be used to test your installation. It is recommended that you follow the links from top to bottom to get the stages working.

Figure 2.1. Negotiation Toolkit Front Page

Note

Before using the Negotiation Toolkit you should have completed the installation process, a number of the actions in the toolkit involve either the application server or the web browser communicating with the KDC which needs to be correctly configured.

3.2. Basic Negotiation

The 'Basic Negotiation' servlet can be used to test that the web browser does trust the application server to attempt negotation. The servlet simply prompts the web browser to negotiation and outputs if a SPNEGO token was received or not.

An unsucsessful negotation would result in output similar to the following.

Figure 2.2. Basic Negotiation Failure

If the web browser successfully sends a SPNEGO token you should see output similar to the following.

Figure 2.3. Basic Negotiation Success

The resulting web page shows a breakdown of some of the information contained within the negotiation token.

3.3. Security Domain Test

It is important that the application server can authenticate against the KDC using it's own security domain, the 'Security Domain Test' servlet is a servlet that can be used to test that the security domain can authenticate.

On the first page you will need to enter the name of the security domain being used, in these examples the security domain is called 'host'.

Figure 2.4. Security Domain Test

If the authentication is successful you should see output similar to the following.

Figure 2.5. Security Domain Test - Authenticated

3.4. Secured

The final servlet in the toolkit is the 'Secured' servlet, this servlet is configured to require full SPNEGO authentication, if you get output similar to the following output this means you have everything configured correctly.

Figure 2.6. Secured

Microsoft Active Directory

1. Introduction

This chapter describes the steps required to configure the authenticator which are specific to Windows, these instructions are prepared against Windows 2003.

The Windows 2003 machine hosting the user accounts is required to be an Active Directory domain controller - having a Windows machine with accounts managed locally is not sufficient.

1.1. Windows 2003 Support Tools

A couple of additional command line utilities are going to be required when configuring the service accounts on the domain controller, these can be downloaded directly from Microsoft *http://go.microsoft.com/fwlink/?LinkId=100114* [http://go.microsoft.com/fwlink/?LinkId=100114]

2. Server User Creation

First the server requires a user account to be created for it within the domain, this needs to be a normal user account and not a computer account - we will perform some additional steps later to map the user account to a service account.

As we are going to be referring to the server using the name 'testserver.kerberos.jboss.org' we will create a user called 'testserver'.

Warning

It is important to set a valid password on the account as soon as you create as changing the password later can invalidate the keytab that you export which would break your JBoss installations.

The first step is to create the actual user.

Figure 3.1. New User

Figure 3.2. New User Password

Set the password for the new user and ensure both 'User cannot change password' and 'Password never expires' are set.

Figure 3.3. New User Finish

Next you will need to open the properties dialog for the user to make one more change to make the account suitable as a service account.

Figure 3.4. User Properties

It is important to ensure that 'Do not require Kerberos preauthentication' is checked, occasionally it is also required to check 'Use DES encryption types for this account' but recent testing has not required this.

Note

Some of the domain details are slightly different in these images as the test domain I am using is slightly different to the domain I am using in the examples in this document.

3. Service Account Mapping

After the user account has been created it needs to be mapped to a host account using the setspn.exe and ktpass.exe command line utilities included in the Windows 2003 Support Tools.

The first utility to use is the setspn.exe utility installed with the Windows 2003 support tools. Documentation for this tool is available from Microsoft *http://technet2.microsoft.com/windowsserver/en/library/b3a029a1-7ff0-4f6f-87d2-f2e70294a5761033.mspx?mfr=true* [http://technet2.microsoft.com/windowsserver/en/library/b3a029a1-7ff0-4f6f-87d2-f2e70294a5761033.mspx?mfr=true]

You should execute the following two command to map the testserver user to the correct service principals.

```
setspn.exe -a host/testserver.kerberos.jboss.org testserver
setspn.exe -a HTTP/testserver.kerberos.jboss.org testserver
```

Figure 3.5. Set Service Principals

The following command then can be used to list the mappings.

```
        setspn.exe -l testserver
```

Figure 3.6. List Service Principals

The next step is to use the ktpass.exe utility from the Windows 2003 support tools and also the ktab.exe tool from the Java installation to export the keytab.

> **Note**
>
> This section needs some further consideration, it may be possible to just use one of the commands.

The ktpass.exe command line utility takes the user created earlier and maps it as a trusted host, in this case you would need to execute the following command: -

```
        ktpass -princ host/testserver@kerberos.jboss.org -pass *
   -mapuser KERBEROS\testserver
  -out C:\testserver.host.keytab
```

Figure 3.7. KTPass

The ktab.exe utility is then used to export the keytab that will be used by the application server using the following command.

```
        ktab -k c:\testserver.host.keytab -a
   testserver@KERBEROS.JBOSS.ORG
```

Figure 3.8. Export Keytab

The resulting keytab should then be used in setting up the host security domain as described in *Section 2.3, “Host Security Domain”*

4. Role Mapping

Finally you will need to configure a second login module in the application-policy to load the associate roles with the user.

You could make use of the UsersRolesLoginModule as described in *Section 2.4, “Application Security Domain” [8]Section 2.1.2, “Chained Configuration”*

Free IPA

1. Free IPA

1.1. Introduction

This chapter describes the steps required to configure the authenticator which are specific to FreeIPA, these instructions are prepared using Fedora 9 with Free IPA version 1.1.

> FreeIPA is an integrated security information management solution combining Linux (Fedora), Fedora Directory Server, MIT Kerberos, NTP, DNS. It consists of a web interface and command-line administration tools.
>
> —FreeIPA

1.2. Pre-Requisits

These instructions assume that you already have FreeIPA installed and correctly configured along with client already able to obtain Kerberos tickets.

For documentation on how to install and configure FreeIPA please see *http://www.freeipa.org/* [http://www.freeipa.org/] .

Warning

Due to the supported encryption types of FreeIPA the JBoss application server is required to be running on a Java 6 JVM with unlimited cryptography enabled.

1.3. Service Principal Creation

In this example the test server is going to be accessible using the 'test_server.jboss.org' domain, the first step is to create the service principal which will represent this host.

Full information on service principal creation is available within the FreeIPA documentation *http://freeipa.org/page/AdministratorsGuide#Managing_Service_Principals* [http://freeipa.org/page/AdministratorsGuide#Managing_Service_Principals]

The easiest way to create a service principal is to make use of the FreeIPA WebUI when connected as an administrator and use the 'Add Service Principal' link.

Figure 4.1. Add Service Principal

Set the hostname to the hostname of your server, in this case it is 'test_server.jboss.org', set the service type to HTTP and select 'Add Principal'.

Figure 4.2. View Service Principal

> **Note**
>
> Creating the service principal requires the host name to be mapped using DNS, if this check fails you can instead create the principal using the following command from the command line `ipa-addservice HTTP/test_server.jboss.org@JBOSS.ORG --force`

1.4. Export Keytab

> **Caution**
>
> The steps to obtain the keytab reset the secret associated with the principal rendering all previosuly created keytabs for the principal invalid.

Before exporting the keytab you will need to have used the kinit tool to obtain a Kerberos ticket-granting ticket for an administrator e.g. `kinit admin`

The command to obtain the keytab is `ipa-getkeytab` with the following options: -

- -s The IPA server to obtain the keytab from.
- -p The principal to export.
- -k The name of the file to dump the keytab to.

Figure 4.3. Get Keytab

From this point on the server should be configured as described in *Chapter 2, General InstallationSection 3, “Negotiation Toolkit”*

For the role mapping again you could make use of the UsersRolesLoginModule as described in *Section 2.4, “Application Security Domain” [8]Section 2.4, “Chained Configuration”*

Microsoft Internet Explorer

1. Introduction

This chapter describes the configuration required to enable SPNEGO negotiation in Internet Explorer. These instructions are prepared against Internet Explorer 6 on Microsoft Windows 2003

2. Local Intranet

By default Internet Explorer only performs SPNEGO authentication against sites in the 'Local intranet' zone.

Open the 'Internet Options' from the 'Tools' menu: -

Figure 5.1. Tools -> Internet Options

And select the 'Security' tab: -

Figure 5.2. Internet Options

Ensure that 'Local intranet' is highlighted and click the 'Sites' command button.

Figure 5.3. Local Intranet

Enter the URL of the server hosting the JBoss installation and click on 'Add'.

Figure 5.4. Local Intranet - Site Added

This should now be sufficient for Internet Explorer to trust the JBoss installation and to perform the SPNEGO negotiation. This can be tested by using the 'Basic Negotiation' section of the Negotiation Toolkit web application.

Mozilla Firefox

1. Introduction

This chapter describes the configuration required to enable SPNEGO negotiation in Mozilla Firefox. These instructions are prepared against Mozilla Firefox 2.0.0.11 on Microsoft Windows 2003 but have also been tested on Fedora 9 with Firefox 3.0.1.

2. Configuration

To configure Mozilla Firefox you should navigate to the about:config URL which will display all of the configuration options for Firefox.

Set the filter to 'network.negotiate' to reduce the list to the options that relate to negotiation.

Figure 6.1. Firefox Configuration

The two values which are important are 'network.negotiate-auth.trusted-uris' and 'network.negotiate-auth.delegation-uris', the first of these specifies which uris will be trusted for SPNEGO negotiation and the second specifies which the users credentials will actually be delegated to.

For the current implementation only trust is required, the use of delegation is something that will be added in future releases.

The URIS that are entered for the above values can be anything from a partial URI e.g. 'http://', 'testserver' to the full URI e.g. 'http://testserver.jboss.org'.

Figure 6.2. Firefox Configuration

As with Internet Explorer negotiation is automatic and this can be demonstrated using the negotiation toolkit.

Figure 6.3. Firefox Negotiation Toolkit

References

For further reading please see the following reference information.

RFC4178 The Simple and Protected GSS-API Negotiation Mechanism [http://tools.ietf.org/html/rfc4178] [???] *http://tools.ietf.org/html/rfc4178*

Troubleshooting

1. Introduction

When working with the JBoss Negotiation authenticator there are a number of different approaches to identify the cause of failures, this chapter contains information on some of the approaches you can take.

The first tool that you should make use of is the Negotiation Toolikit which is documented in ' *Section 3, “Negotiation Toolkit”*

2. Logging

Within JBoss AS the most important category to enable full TRACE logging is 'org'jboss.security': -

```
<category name="org.jboss.security">
  <priority value="TRACE"/>
</category>
```

Enabling TRACE logging for the 'org'jboss.security' will log additional information for the authentcator, the login module and for the rest of JBoss Security.

The 'com.sun.security.auth.module.Krb5LoginModule' login module also allows you to enable additional logging by setting the 'debug' option to 'true': -

```
      <module-option name="debug">true</module-option>
```

Finally setting the system property `-Dsun.security.krb5.debug=true` will result in much more verbose output of the complete GSSAPI negotiation process.

2.1. Message Tracing

The Log4j logging hierarchies also make it possible to selectively log at TRACE level the actual messages exchanged, both the Request and Response messages can be logged and this can be as Hex or as Base64.

The base category for message tracing is `org.jboss.security.negotiation.MessageTrace` , enabling TRACE logging for this category will cause all request and response messages to be logged at TRACE level both in Hex and in Base64.

```
<category name="org.jboss.security.negotiation.MessageTrace">
  <priority value="TRACE"/>
</category>
```

To reduce the logging to either just the request or just the response messages the category can have `.Request` or `.Response` appended.

```
<category
 name="org.jboss.security.negotiation.MessageTrace.Request">
  <priority value="TRACE"/>
</category>

<category
 name="org.jboss.security.negotiation.MessageTrace.Response">
  <priority value="TRACE"/>
</category>
```

This will cause the request or the response message to be logged as Hex and as Base64.

Finally it is possible to specify that just the Hex or just the Base64 messages should be logged by appending `.Hex` or `.Base64` to the category.

```
<category
 name="org.jboss.security.negotiation.MessageTrace.Request.Hex">
  <priority value="TRACE"/>
</category>

<category
 name="org.jboss.security.negotiation.MessageTrace.Request.Base64">
  <priority value="TRACE"/>
</category>

<category
 name="org.jboss.security.negotiation.MessageTrace.Response.Hex">
  <priority value="TRACE"/>
</category>

<category
 name="org.jboss.security.negotiation.MessageTrace.Response.Base64">
```

```
    <priority value="TRACE"/>
  </category>
```

Appendix A. Advanced LDAP Login Module

The JBoss Negotiation project includes a new LDAP login module to handle the LDAP role searching requirements.

The new login module has been based on the existing LdapExtLoginModule. The new module now allows for GSSAPI to be used for authentication when searching LDAP and the the configuration allows for the users search, the authentication or the roles search to be skipped as is required.

1. Configuration

The fully qualified classname of the new login module is `org.jboss.security.negotiation.AdvancedLdapLoginModule`

Warning

If you made use of the Beta releases the class name was `org.jboss.security.negotiation.spnego.AdvancedLdapLoginModul` the LoginModule is still available using this name, however it has been deprecated and will be removed in a future release.

The following sections will describe the various configuration options for this login module.

This login module supports the 'password-stacking', if this module is being used in conjunction with other login modules this should be set to 'useFirstPass'.

1.1. Search Connection

The first settings are the setting used to obtain the *InitialLdapContext* [http://java.sun.com/j2se/1.5.0/docs/api/javax/naming/ldap/InitialLdapContext.html] used to search for the user and to search for the users roles.

The login module supports obtaining this InitialLdapContext using a username and credential or using GSSAPI for a previously authenticated user.

1.1.1. Username / Credential Authentication

To authenticate using a username and password the following settings are required.

- bindDN - The DN used to bind against the LDAP server for the user and roles queries. This is some DN with read/search permissions on the baseCtxDN and rolesCtxDN values.
- bindCredential - The password for the bindDN. This can be encrypted if the jaasSecurityDomain is specified.
- jaasSecurityDomain - The JMX ObjectName of the JaasSecurityDomain to use to decrypt the java.naming.security.principal. The encrypted form of the password is that returned by the JaasSecurityDomain#encrypt64(byte[]) method. The org.jboss.security.plugins.PBEUtils can also be used to generate the encrypted form.

1.1.2. GSSAPI Authentication

- bindAuthentication - Set this to GSSAPI for GSSAPI based authentication.
- jaasSecurityDomain - The security domain to obtain the Subject required for the connection.

Note

For information on defining the required jaasSecurityDomain see ' *Section 2.3, "Host Security Domain"*

As with the original LdapExtLoginModule all of of the properties provided to this login mode are passed into the InitialLdapContext constructor so you can make use of any of the options supported by the LdapCtxFactory you are using.

1.2. User DN Search

The first step this login module performs is to take the provided username and search for the DN of the user.

- baseCtxDN - The fixed DN of the context to search for user roles. Consider that this is not the Distinguished Name of where the actual roles are; rather, this is the DN of where the objects containing the user roles are (e.g. for active directory, this is the DN where the user account is)
- baseFilter - A search filter used to locate the context of the user to authenticate. The input username/userDN as obtained from the login module callback will be substituted into the filter anywhere a "{0}" expression is seen. This substitution behavior comes from the standard DirContext?.search(Name, String, Object[], SearchControls? cons) method. An common example search filter is "(uid={0})".

- searchTimeLimit - The timeout in milliseconds for the user/role searches. Defaults to 10000 (10 seconds).

Note

It is possible to disable the user DN search by omitting the 'baseCtxDN' property. In this case the provided username will be used as the DN instead for the following steps in this login module.

1.3. User Authentication

If this login module is not the first login module and a previous login module has already authenticated the user this step will be skipped.

If no previous login module has authenticated the user this step takes the User DN from the User DN search and their provided credential and attempts to create a new InitialLdapContext to verify that the User DN and credential combination is valid.

There is only one additional setting to control the behaviour of the user authentication.

- allowEmptyPasswords - A flag indicating if empty (length==0) passwords should be passed to the ldap server. An empty password is treated as an anonymous login by some ldap servers and this may not be a desirable feature. Set this to false to reject empty passwords, true to have the ldap server validate the empty password. The default is false.

1.4. Roles Search

This final step searches for the roles that the user is a member of.

Caution

The settings for this section are similar to the LdapExtLoginModule but do be careful at the recursion now works by finding the roles listed within a DN.

- rolesCtxDN - The fixed DN of the context to search for user roles. Consider that this is not the Distinguished Name of where the actual roles are; rather, this is the DN of where the objects containing the user roles are (e.g. for active directory, this is the DN where the user account is)

- roleFilter - A search filter used to locate the roles associated with the authenticated user. The input username/userDN as obtained from the login module callback will be substituted into the filter anywhere a "{0}" expression is seen. The authenticated userDN will be substituted into the filter anywhere a "{1}" is seen. An example search filter that matches on the input username is: "(member={0})". An alternative that matches on the authenticated userDN is: "(member={1})".

Note

The roleFilter attribute can be ommitted and the role search will then use the UserDN as the DN to obtain the roleAttributeID value.

- roleAttributeID - The name of the role attribute of the context which corresponds to the name of the role. If the roleAttributeIsDN property is set to true, this property is the DN of the context to query for the roleNameAttributeID attribute. If the roleAttributeIsDN property is set to false, this property is the attribute name of the role name.
- roleAttributeIsDN - A flag indicating whether the user's role attribute contains the fully distinguished name of a role object, or the users's role attribute contains the role name. If false, the role name is taken from the value of the user's role attribute. If true, the role attribute represents the distinguished name of a role object. The role name is taken from the value of the roleNameAttributeId` attribute of the corresponding object. In certain directory schemas (e.g., Microsoft Active Directory), role (group)attributes in the user object are stored as DNs to role objects instead of as simple names, in which case, this property should be set to true. The default value of this property is false.
- roleNameAttributeID - The name of the role attribute of the context which corresponds to the name of the role. If the roleAttributeIsDN property is set to true, this property is used to find the role object's name attribute. If the roleAttributeIsDN property is set to false, this property is ignored.
- recurseRules - Enable a recursive role search. The login module tracks already added roles to cope with cyclic references.
- searchScope - Sets the search scope to one of the strings. The default is SUBTREE_SCOPE.
  - OBJECT_SCOPE - only search the named roles context.
  - ONELEVEL_SCOPE - search directly under the named roles context.
  - SUBTREE_SCOPE - If the roles context is not a DirContext?, search only the object. If the roles context is a DirContext?, search the subtree rooted at the named object, including the named object itself
- searchTimeLimit - The timeout in milliseconds for the user/role searches. Defaults to 10000 (10 seconds).

Note

The searchTimeLimit setting is shared between both of the searches.

2. Examples

Here are some example configurations making use of the new login module.

2.1. Active Directory

Here are two example configurations making use of Active Directory, the first example shows the login module being used for both of the searches and the authentication. The second example shows the login module being chained after the SPNEGOLoginModule.

The following is an extract of the dumped ldiff from the Active Directory domain these examples were tested against.

```
dn: CN=Darran
 Lofthouse,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Darran Lofthouse
distinguishedName:
 CN=Darran
 Lofthouse,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
memberOf:
 CN=Banker,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
name: Darran Lofthouse
sAMAccountName: darranl
userPrincipalName: darranl@vm104.gsslab.rdu.redhat.com

dn: CN=Banker,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
objectClass: top
objectClass: group
cn: Banker
member:
 CN=Darran
 Lofthouse,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
distinguishedName:
 CN=Banker,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
memberOf:
 CN=Trader,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
```

```
name: Banker
sAMAccountName: Banker

dn: CN=Trader,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
objectClass: top
objectClass: group
cn: Trader
member:
 CN=Banker,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
distinguishedName:
 CN=Trader,CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com
name: Trader
sAMAccountName: Trader
```

2.1.1. Full Authentication

The following configuration would require a username and password to be provided for the authentication process.

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module

 code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">

      <module-option
 name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>
      <module-option
 name="java.naming.provider.url">ldap://VM104:3268</module-option>

      <module-option
 name="baseCtxDN">CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com</
module-option>
      <module-option
 name="baseFilter">(sAMAccountName={0})</module-option>

      <module-option
 name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">true</module-option>
    </login-module>
  </authentication>
```

```
</application-policy>
```

The first three options 'bindAuthentication', 'jaasSecurityDomain', and 'java.naming.provider.url' configure how the login module will connect to LDAP and how the authentication will occur.

The 'baseCtxDN' option is the DN to start the search for the user, the 'baseFilter' attribute in this example searches for the user using the 'sAMAccountName' attribute.

As the 'memberOf' attribute is going to be read directly from the user there is no need to specify a 'rolesCtxDN' or 'roleFilter', instead the attribute named by the 'roleAttributeID' option which in this case is 'memberOf' will be read directly from the user.

The 'roleAttributeIsDN' option then specifies that this value is a DN so the group object is retrieved and finally the 'roleNameAttributeID' option specifies that the attribute 'cn' should be read from the group. This is the role that this login module returns.

Finally the 'recurseRoles' attribute is set to true so the DN from the located group is used to repeat the process so if a group is configured with the 'memberOf' attribute then this will be recursively used to locate all of the roles.

2.1.2. Chained Configuration

The following configuration shows the AdvancedLdapLoginModule chained after the SPNEGOLoginModule.

```


<application-policy name="SPNEGO">
  <authentication>
    <login-module
     code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
     flag="requisite">
      <module-option
 name="password-stacking">useFirstPass</module-option>
      <module-option
 name="serverSecurityDomain">host</module-option>
    </login-module>

    <login-module

 code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option
 name="password-stacking">useFirstPass</module-option>
```

```
      <module-option
 name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>
      <module-option
 name="java.naming.provider.url">ldap://VM104:3268</module-option>

      <module-option
 name="baseCtxDN">CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com</
module-option>
      <module-option
 name="baseFilter">(userPrincipalName={0})</module-option>


      <module-option
 name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

The majority of this configuration is identical to the previous example, the first notable exception is that both login modules should have 'password-stacking' to 'useFirstPass' this allows the first login module to pass the username to the second login module.

The second change is that now the 'baseFilter' now searched on the 'userPrincipalName', this is because this is the name identified by the SPNEGOLoginModule.

Apart from these changes the rest of these changes are identical to ' *Section 2.1.1, “Full Authentication”*

2.2. Free IPA

Here are two example configurations making use of the Open LDAP part of Free IPA, the first example shows the login module being used for both of the searches and the authentication. The second example shows the login module being chained after the SPNEGOLoginModule.

The following is an extract of the dumped ldiff from the Free IPA domain these examples were tested against.

```
dn: uid=darranl,cn=users,cn=accounts,dc=jboss,dc=org
displayName: Darran Lofthouse
uid: darranl
title: Mr
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: posixAccount
objectClass: krbPrincipalAux
objectClass: radiusprofile
sn: Lofthouse
mail: darran.lofthouse@jboss.com
krbPrincipalName: darranl@JBOSS.ORG
givenName: Darran
cn: Darran Lofthouse
initials: DL
memberOf: cn=banker,cn=groups,cn=accounts,dc=jboss,dc=org
memberOf: cn=Trader,cn=groups,cn=accounts,dc=jboss,dc=org

dn: cn=Banker,cn=groups,cn=accounts,dc=jboss,dc=org
objectClass: top
objectClass: groupofnames
objectClass: posixGroup
objectClass: inetUser
cn: Banker
memberOf: cn=trader,cn=groups,cn=accounts,dc=jboss,dc=org
member: uid=darranl,cn=users,cn=accounts,dc=jboss,dc=org

dn: cn=Trader,cn=groups,cn=accounts,dc=jboss,dc=org
objectClass: top
objectClass: groupofnames
objectClass: posixGroup
objectClass: inetUser
cn: Trader
member: cn=Banker,cn=groups,cn=accounts,dc=jboss,dc=org
```

As you can see for the purpose of configuring the login modules the structure is very similar to the structure used by Active Directory so the resulting login module configuration is going to be very similar.

2.3. Full Authentication

The following configuration would require a username and password to be provided for the authentication process.

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module
 code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option
 name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>

      <module-option
 name="java.naming.provider.url">ldap://kerberos.jboss.org:389</
module-option>

      <module-option
 name="baseCtxDN">cn=users,cn=accounts,dc=jboss,dc=org</module-
option>
      <module-option name="baseFilter">(uid={0})</module-option>


      <module-option
 name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

The first three options 'bindAuthentication', 'jaasSecurityDomain', and 'java.naming.provider.url' configure how the login module will connect to LDAP and how the authentication will occur.

The 'baseCtxDN' option is the DN to start the search for the user, the 'baseFilter' attribute in this example searches for the user using the 'uid' attribute.

As the 'memberOf' attribute is going to be read directly from the user there is no need to specify a 'rolesCtxDN' or 'roleFilter', instead the attribute named by the 'roleAttributeID' option which in this case is 'memberOf' will be read directly from the user.

The 'roleAttributeIsDN' option then specifies that this value is a DN so the group object is retrieved and finally the 'roleNameAttributeID' option specifies that the attribute 'cn' should be read from the group. This is the role that this login module returns.

Finally the 'recurseRoles' attribute is set to true so the DN from the located group is used to repeat the process so if a group is configured with the 'memberOf' attribute then this will be recursively used to locate all of the roles.

2.4. Chained Configuration

The following configuration shows the AdvancedLdapLoginModule chained after the SPNEGOLoginModule.

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module

 code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
       flag="requisite">
       <module-option
 name="password-stacking">useFirstPass</module-option>
       <module-option
 name="serverSecurityDomain">host</module-option>
    </login-module>

    <login-module

 code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option
 name="password-stacking">useFirstPass</module-option>

      <module-option
 name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>

      <module-option
 name="java.naming.provider.url">ldap://kerberos.jboss.org:389</
module-option>

      <module-option
 name="baseCtxDN">cn=users,cn=accounts,dc=jboss,dc=org</module-
option>
      <module-option
 name="baseFilter">(krbPrincipalName={0})</module-option>

      <module-option
 name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>
```

```
        <module-option name="recurseRoles">true</module-option>
      </login-module>
    </authentication>
  </application-policy>
```

As with the Active Directory examples the majority of this configuration is identical to the previous example, the first notable exception is that both login modules should have 'password-stacking' to 'useFirstPass' this allows the first login module to pass the username to the second login module.

The second change is that now the 'baseFilter' now searched on the 'krbPrincipalName', this is because this is the name identified by the SPNEGOLoginModule.

Apart from these changes the rest of these changes are identical to ' *Section 2.3, “Full Authentication”*