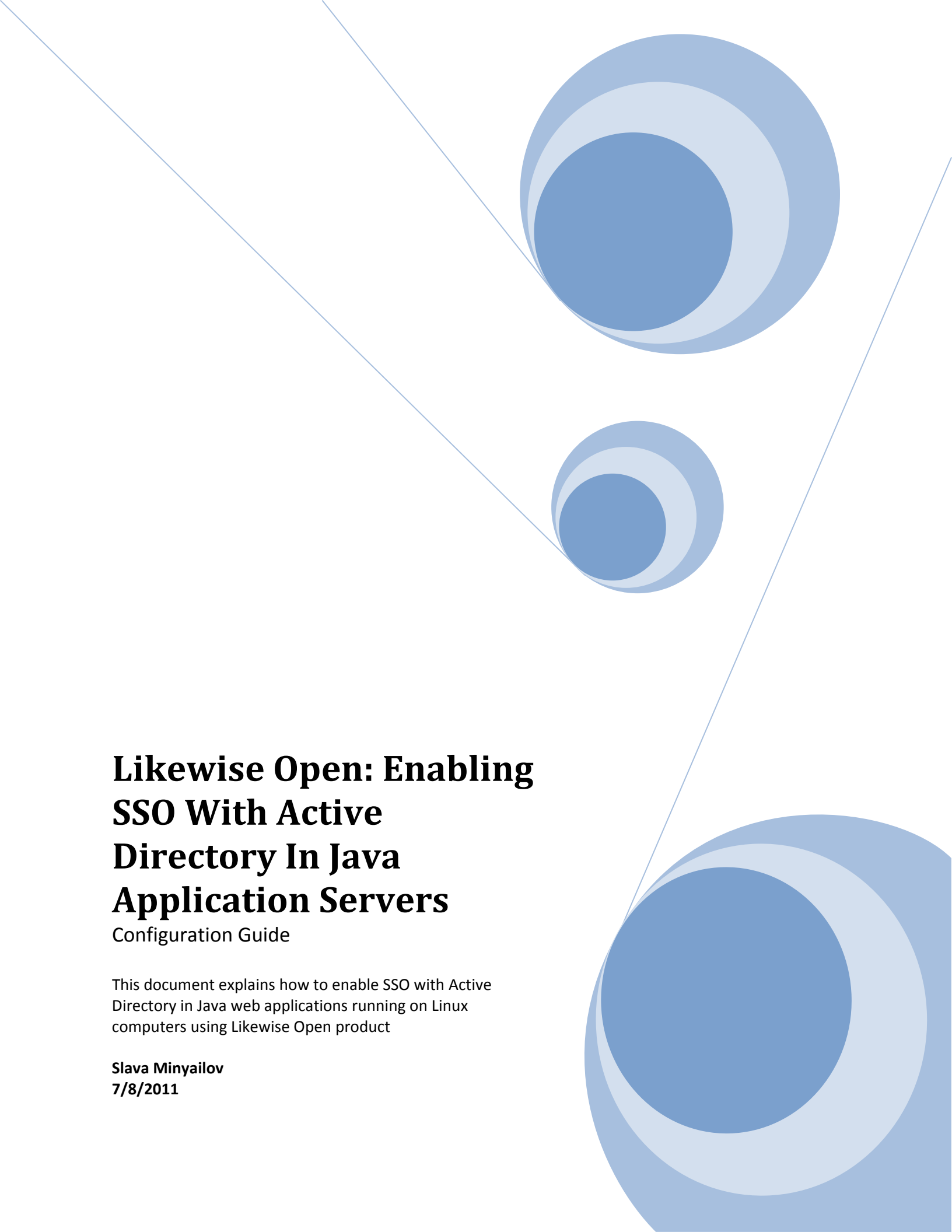


# Likewise Open: Enabling SSO With Active Directory In Java Application Servers

Configuration Guide

This document explains how to enable SSO with Active Directory in Java web applications running on Linux computers using Likewise Open product

**Slava Minyailov**
**7/8/2011**

# Contents

# Understanding Integrated Windows Authentication

Integrated Windows Authentication was introduced with the Microsoft Windows 2000 operating system. It is based on the SPNEGO, Kerberos, and NTLMSSP protocols. The SPNEGO protocol is used between the web browser and the web server to negotiate the type of authentication that will be performed, usually either Kerberos or NTLMSSP. Kerberos is the preferred authentication mechanism. Both Kerberos and NTLMSSP are secure protocols that allow computers to authenticate a user over a non-secure channel. For web sites, this means that the Secure Socket Layer (SSL) protocol does not need to be enabled during the authentication phase.

## Why use Integrated Windows Authentication?

Integrated Windows Authentication improves the overall security of a network because the user must log on by using his or her username and password only once. All subsequent accesses by that user to resources, such as web sites, file systems, and network printers are automatically authenticated with cached security tokens. Using Integrated Windows Authentication has the benefit of a centralized user account database stored in Active Directory. This is more secure and more efficient than duplicating user names and passwords in configuration files across server computers.

## Kerberos, NTLMSSP versus Basic Authentication

Integrated Windows Authentication uses the SPNEGO, Kerberos and NTLM authentication protocols. Not all browsers are capable of understanding these protocols. Another authentication protocol, Basic Authentication, is understood by all web browsers; it works by simply transferring username and password across the network from the web browser to the web server. The drawback of using Basic Authentication is that without SSL encryption, anyone can intercept the network communication and easily find out a user's login name and password. Thus, Basic Authentication should be used only for sites that are protected with SSL encryption.

## Authentication versus Authorization

The term *authentication* refers to the process of proving a user's identity. *Authorization*, on the other hand, takes place after authentication and is used to limit the users or groups to perform only the actions that they are allowed or authorized to perform. Integrated Windows Authentication provides only the authentication mechanism while Windows authorization is usually accomplished using Access Control Lists (ACL).

# Configuring a Java Application Server for SSO

This section describes how to set up Likewise Open and a Java application server to provide secure single sign-on through Active Directory with Integrated Windows Authentication. The instructions use Apache Tomcat as an example to demonstrate how to implement single sign-on with the Likewise servlet authentication filter. Because servlet filters is a generic Java technology common to all application servers compliant with Java servlet specification 2.3 and higher, the procedure is similar for other Java application servers. The instructions assume that you know how to configure Active Directory, Tomcat, and computers running Linux.

Before you can integrate Likewise Open with a Java application server, you must install a separate application integration package, which you can download for free from the Likewise web site by registering to download Likewise Open. (The name of the application integration package looks like this: `LikewiseAppIntegration-6.1.0.8656-linux-i386-rpm.sh`.)

Once you have installed the application integration package, here is how to configure your Tomcat server for SSO with Likewise. This section assumes you have installed Likewise 6.1 or later on the computer running the Java application server and have joined the server to an Active Directory domain.

## Requirements

- Root access to the Linux computer.
- The Linux computer is joined to Active Directory with Likewise 6.1 or higher
- Administrative access to the Active Directory for creating a service account if one does not exist
- The Linux computer is running Apache Tomcat Server version 5.5 or 6.0.
- The server is running JRE 1.5.0 or higher.
- The Likewise application integration package is installed.

**Important:** Configuring Java application servers is a complex procedure. Before you deploy your configuration to a production web server, implement and test it in a test environment. More: Before you change your application server's configuration, read and understand the documentation that accompanies the application server. Before you change a file, make a backup copy of it.

## Components

The following Likewise components relevant to Apache integration are installed at `/opt/likewise/{lib, lib64}`:

| Component name | Description |
|---|---|
| lwjplatform.jar | Likewise Platform Library |
| lwservlets.jar | Likewise authentication modules, including Servlet Filter and JAAS Module. |
| lwtomcat.jar | Likewise authentication modules specific to implementing the Tomcat authentication valve. This component is not needed if you use the Likewise authentication filter described in this document. |

| Component name | Description |
|---|---|
| jna.jar | Java Native Access Library patched for UCS-2 support. |
| commons-logging-1.1.1.jar, log4j-1.2.16.jar | Logging facilities |
| commons-codec-1.4.jar | Base64 and other encoding routines |
| commons-net-2.2.jar | Network utilities |

## Install the Likewise Authentication Components

The components from the integration package must be installed. Typically, an Apache Tomcat installation uses the following environment variables:

| Environment Variable | Value |
|---|---|
| CATALINA_HOME_DIR | /usr/share/tomcat5 or /usr/share/tomcat6 |
| CATALINA_BASE_DIR | /var/lib/tomcat5 or /var/lib/tomcat6 |

To integrate your web application with the Likewise servlet filter you must install the following components in `${CATALINA_HOME_DIR}/webapps/<web application>/lib`:

```
lwservlets.jar, lwjplatform.jar, jna.jar, commons-logging-1.1.1.jar, log4j-1.2.16.jar

commons-codec-1.4.jar, commons-net-2.2.jar
```

Symbolic links can be created to these jar files from the target directory.

## Generate Kerberos Keytab File

The Kerberos keytab file is necessary to authenticate incoming requests. It contains an encrypted, local copy of the host's key. It is therefore important to protect it with proper file access permissions. The file must be readable by the user or group under which the Apache Tomcat server is running, typically `tomcat` on most Linux systems. No other user should be able to read or modify the file.

First, you must find the server name of the web site that will require authentication. If you don't know the server name, try doing a reverse DNS name lookup on the IP address of the host or just use the hostname of the Linux system.

You will also need to know the fully qualified name of the domain to which the Linux system is joined.

Finally, you will need to decide where to save the generated keytab file.

The steps below use a sample Apache user account name named `tomcat`, a sample server name of `myserver`, a sample full domain name of `MYDOMAIN.COM`, and a sample Kerberos keytab file named `/etc/krb5_myserver.keytab`. You must substitute the correct names from your system and configuration.

1. Select a user in Active Directory for the application server. If you create a new user, make sure to set the password for the account to "never expire". Also, make sure "Use DES Encryption types for this account" is not checked in the user account properties in Active Directory. In this example, we are using the user: `MYDOMAIN\tomcat`

2. Generate keytab entry on your Windows domain controller for the default HTTP service principal:

```
# ktpass /out c:\krb5_myserver.keytab /pType KRB5_NT_PRINCIPAL /crypto RC4-
HMAC-NT /princ HTTP/myserver.mydomain.com@MYDOMAIN.COM /mapuser
tomcat@MYDOMAIN.COM /mapop set /pass *
```

3. Copy the file to the target Linux machine and change the group ownership of the keytab file:

```
# chown tomcat:tomcat /etc/krb5_myserver.keytab
```

4. Set appropriate file permissions of the keytab file:

```
# chmod 600 /etc/krb5_myserver.keytab
```

If you choose not to create a separate keytab but rather merge the generated data with default keytab file (typically `/etc/krb5.keytab`), you must provide read access to the file to the local `tomcat` user.

```
# chgrp tomcat /etc/krb5.keytab

# chmod g+r /etc/krb5.keytab
```

5. Set the KRB5_KTNAME environment variable of the Tomcat process. This can be set in the beginning of the Tomcat initialization script at /etc/init.d/tomcat:

```
export KRB5_KTNAME=FILE:/etc/krb5_myserver.keytab
```

## Modify the Application Web Descriptor

Now you need to add `filter and filter-mapping` sections to the web descriptor of your web application (web.xml). It must be done for every web application you are planning to protect with the Likewise servlet filter. This is how your configuration may look like (remember to replace `MYDOMAIN` with your short domain name):

```
<filter>
  <filter-name>LikewiseAuth</filter-name>
  <filter-class>com.likewise.auth.filter.spnego.LikewiseNegotiateFilter</filter-class>
  <init-param>
     <param-name>deny-role</param-name>
     <param-value>MYDOMAIN\guests</param-value>
  </init-param>
  <init-param>
     <param-name>allow-role</param-name>
     <param-value>MYDOMAIN\domain^users</param-value>
  </init-param>
  <init-param>
     <param-name>remote-address-accept-filter</param-name>
     <param-value>10.100.0.0/24</param-value>
  </init-param>
  <init-param>
     <param-name>remote-address-accept-filter</param-name>
     <param-value>10.100.1.0,255.255.255.0</param-value>
  </init-param>
</filter>


<filter-mapping>
  <filter-name>LikewiseAuth</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

The above configuration ensures that only members of the `MYDOMAIN\domain^users` group can access the web pages from this application. Users who belong to the `MYDOMAIN\guests` group will be denied access. It is possible to configure multiple deny and allow roles. The user is checked for membership in the deny roles before being checked in the allow roles. Note that role names are case-sensitive. Always use upper-case register for the domain name component and low-case for the group names. Use "^" in place of white spaces in group names.

The `remote-address-accept-filter` configuration parameter can be used to specify IP addresses in the CIDR format, e.g. 10.100.0.0/24. It can also be in the form of IP Address with Subnet mask, e.g. 10.100.0.0,255.255.255.0. If at least one `remote-address-accept-filter` parameter is specified, the servlet performs authentication only of the requests whose remote IP Address is in the range of the permitted addresses. The filter first tries to obtain the IP address of the client from X-Forwareded-For HTTP header. The X-Forwarded-For (XFF) HTTP header field is a de facto standard for identifying the originating IP address of a client connecting to a web server through an HTTP proxy or load balancer. If the header is not set the filter gets the client IP address directly from the TCP/IP socket. If the client IP address is not in the configured range of accepted IP addresses the filter passes the HTTP request to the web application unauthenticated. As a result, the user principal object will not be set in the HTTP session. The access control logic of the web application must decide how to treat unauthenticated HTTP requests. It may choose to reject access, or redirect the request to a different URL, or allow restricted access to the application resources.

## Edit the Java policy file

The JVM running the application server puts curtain restrictions on the type of operations the web application code is allowed to perform. The Likewise authentication filter must be able to read configuration files, and load & execute native libraries in the /opt/likewise directory. For this reason, the Java policy file needs to be extended. This step can be skipped if the Java security manager is disabled in your application server configuration. Below is a sample policy file for Tomcat6. This file can be saved as /var/lib/tomcat6/conf/policy.d/20sample.policy. Consult your application server documentation to find the location of the policy file. Note that name "myapp" below must be substituted with the real name of your web application. You will also need to adjust the "${catalina.base}" variable appropriately since it is only available in Tomcat:

```
grant codeBase "file:${catalina.base}/webapps/myapp/WEB-INF/lib/jna.jar" {

  permission java.security.AllPermission;

};

grant codeBase "file:${catalina.base}/webapps/myapp/WEB-
INF/lib/lwjplatform.jar" {

  permission java.security.AllPermission;
```

```
};

grant codeBase "file:${catalina.base}/webapps/myapp/WEB-
INF/lib/lwservlets.jar" {

  permission java.security.AllPermission;

};

grant codeBase "file:/opt/likewise/lib/-" {

  permission java.security.AllPermission;

};

grant codeBase "file:/opt/likewise/lib64/-" {

  permission java.security.AllPermission;

};
```

## Protect the Web Pages

You can protect access to resources of your web application either programmatically, or declaratively. It is also possible to use a combination of both methods.

Programmatic security provide for the following APIs, which you can use in JSPs and servlets of your web application:

- **isUserInRole()**. This method determines if the currently authenticated user belongs to a specified role. Roles correspond to groups in Active Directory the Linux computer is joined to. For example, if an HTTP request has been successfully authenticated for user *valjean* who is a member of *MYDOMAIN\domain^users* group, the following expression would return true:

  request.isUserInRole("*MYDOMAIN\domain^users*")

  If no user is currently authenticated (e.g., if authorization failed or if isUserInRole is called from an unrestricted page and the user has not yet accessed a restricted page), isUserInRole returns false.

- **getRemoteUser()**. This method returns the name of the current user. For example, if the client has successfully logged in as user valjean, request.getRemoteUser() would return "valjean". If no user is currently authenticated (e.g., if authorization failed or if isUserInRole is called from an unrestricted page and the user has not yet accessed a restricted page), getRemoteUser returns null.

- **getUserPrincipal()**. This method returns the current username wrapped inside a java.security.Principal object. If no user is currently authenticated, getUserPrincipal returns null. The Principal object contains little information beyond the username (available with the getName method). Likewise extends the standard Principal class to provide access to additional principal information through such methods as getUserId(), getPrimaryGroupId(), getHomeDirectory(), getSecurityIdentifier(), getGecos(), getShell(), getDistinguishedName(), getPrincipalName(), isLocalUser(). This information can be accessed after casting the Principal instance to LikewiseUser class:

```
Principal p = request.getUserPrincipal();

if(p != null) {

   LikewiseUser lwUser = (LikewiseUser) p;

}
```

Declarative security model enables you to describe access control logic directly in the web.xml file. For instance the following `web.xml` sample illustrates how to protect all the web pages in your application so they are accessible only to users who belong to the `MYDOMAIN\domain^users` group.

```
<security-role>

   <role-name>MYDOMAIN\domain^users</role-name>

</security-role>

<security-constraint>

   <display-name>Likewise Security Constraint</display-name>

   <web-resource-collection>

      <web-resource-name>Protected Area</web-resource-name>

      <url-pattern>/*</url-pattern>

   </web-resource-collection>

   <auth-constraint>

      <role-name>MYDOMAIN\domain^users</role-name>

    </auth-constraint>

</security-constraint>
```

### Configure the JAAS Module

Likewise uses Kerberos5 JAAS module for user authentication. This section explains how to set up the JAAS module to complete the integration of your application server with Likewise.

1. Create a file named `/opt/likewise/share/config/jaas.policy` and add the following lines to it:

```
grant Principal * * {

    permission java.security.AllPermission "/*";

};
```

2. Create a file name `/opt/likewise/share/config/login.conf` and add the following lines to it:

```
Jaas {

    com.likewise.auth.jaas.LikewiseLoginModule sufficient;

};
```

3. Add the following java parameters to the command line of the script that starts your application server:

```
-Djava.security.auth.login.config=/opt/likewise/share/config/login.conf

-Djava.security.auth.policy=/opt/likewise/share/config/jaas.policy
```

### Restart the Tomcat Server

Restart your application server, e.g.

```
/etc/init.d/tomcat restart
```

## Setting Up Firefox and Internet Explorer for SSO

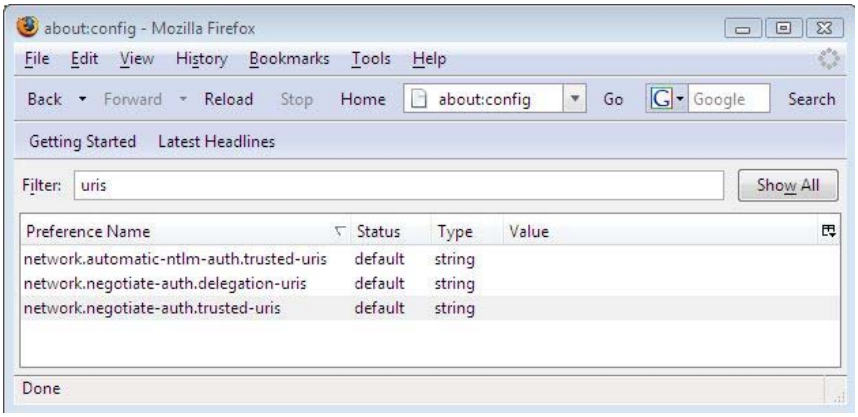
### Configure Firefox for SSO

To set up Firefox for single sign-on, you must turn on the Simple and Protected GSS-API Negotiation Mechanism, or SPNEGO, to negotiate authentication with Kerberos.

1. Open Firefox.

2. In the **Go** box, type `about:config`, and then click **Go**.

3. In the **Filter** box, type `uris`.




4. Double-click **network.negotiate-auth.trusted-uris**, enter a comma-separated list of URL prefixes or domains that are permitted to engage in SPNEGO authentication with the browser, and then click **OK**:

   Example:


5. Double-click **network.negotiate-auth.delegation-uris**, enter a comma-separated list of the sites for which the browser may delegate user authorization to the server, and then click **OK**.

   For more information on how to configure Firefox, see http://grolmsnet.de/kerbtut/firefox.html

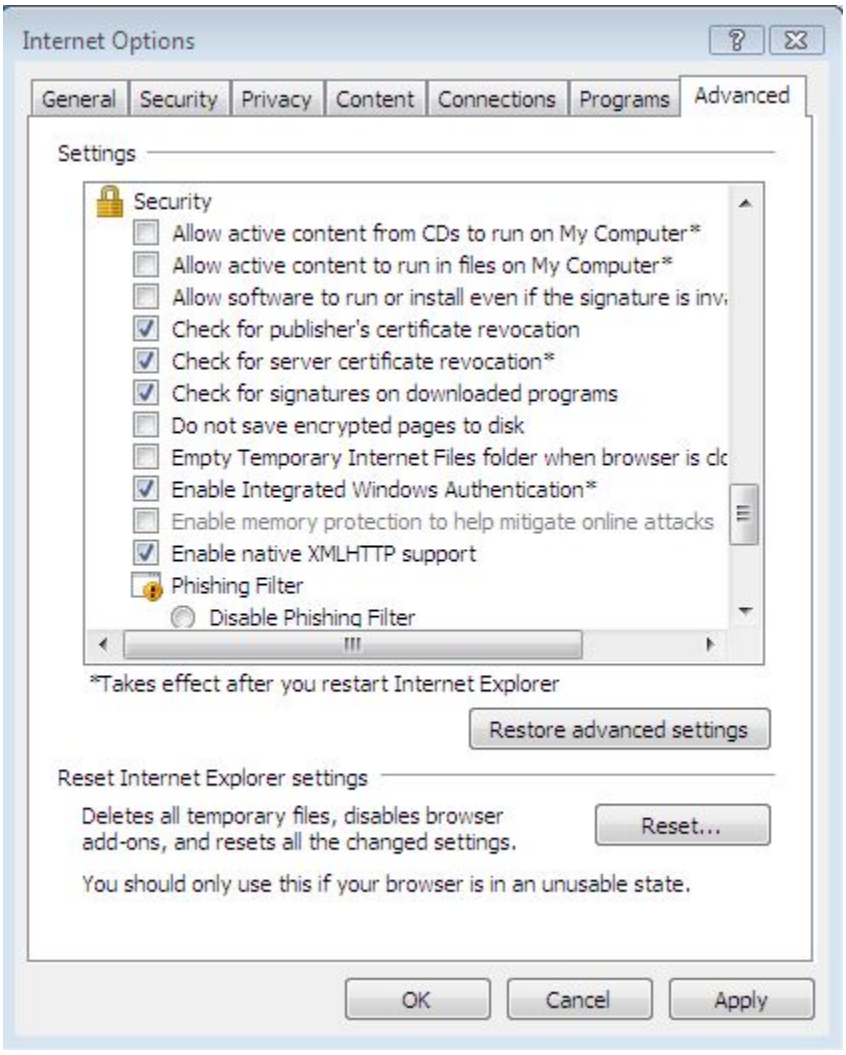
## Configure Internet Explorer for SSO

Here's how to configure Internet Explorer 7.0 to use SPNEGO and Kerberos. The settings for other versions of IE might vary; see your browser's documentation for more information.

1. Start Internet Explorer 7.0.

2. On the **Tools** menu, click **Internet Options**.

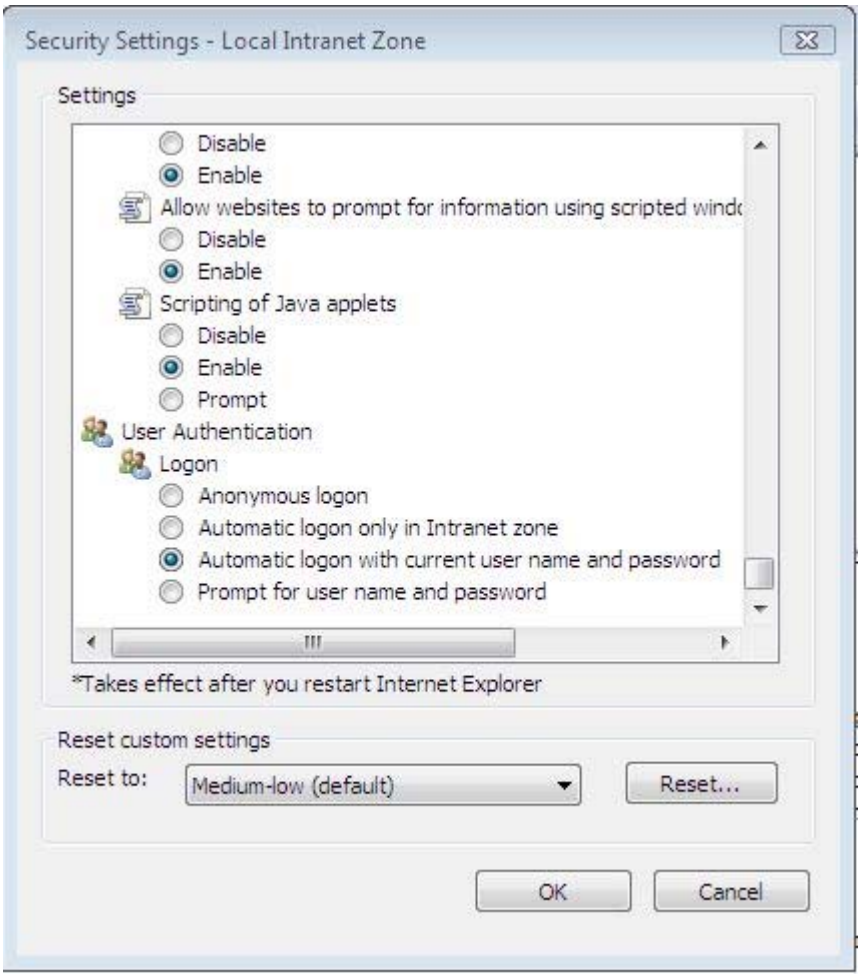
3. Click the **Advanced** tab and make sure that the **Enable Integrated Windows Authentication** box is selected:




4. Click the **Security** tab.

5. Select a zone -- for example, **Local intranet** -- and then click **Custom level**.

6. In the **Settings** list, under **User Authentication**, click **Automatic logon with current user name and password** for a trusted site, or **Automatic logon only in Intranet zone** for a site you added to IE's list of Intranet sites. For more information, see your browser's documentation.




7. Return to the **Security** tab for **Internet Options** and set your web server as a trusted site.

8. Restart Internet Explorer.

## Test Authentication

The first test is to determine if Integrated Windows Authentication is working for all domain users. As depicted in the above configuration examples, protect your web pages to be accessible by members of the “MYDOMAIN\domain^users”.

**Note:** Use Internet Explorer because it supports Integrated Windows Authentication. Verify that the “Enable Integrated Windows Authentication” checkbox is selected in the Internet Explorer Internet Options dialog on the Advanced tab. Also, make sure the target server is allowed to be trusted in the Intranet Zone.

To Single-Sign-On for a domain user:

1. Logon to a Windows computer that is joined to the same domain you joined

your Linux or Unix system to. Logon as a domain user.

2. Access the protected web site from Internet Explorer using the host name. This should use Kerberos authentication if the DNS settings for the client and server are configured accurately.

3. Access the protected web site using the IP address of the server because doing so will result in the Internet Explorer using NTLM for the server.

The authentication should succeed without a need to provide a user name and password.

# Troubleshooting

In the case of authentication failure, there are some diagnostic tools that may help diagnose a problem.

## Apache Tomcat Log File

The Apache Tomcat log file may contain helpful information as to the nature of the problem. Typically the tomcat logs are located under ${CATALINA_HOME_DIR}/logs. It is possible to set the "java.security.debug" variable in the Tomcat environment to elevate the log level and to help check for security issues.

## Likewise authentication groups

Use */opt/likewise/bin/lw-list-groups-for-user* to list the groups the current user belongs to.

## The Microsoft KERBTRAY utility

The Microsoft KERBTRAY utility, part of Microsoft Windows 2000 Resource Kit, will help determine whether Internet Explorer obtained a Kerberos ticket for your web server. Integrated Windows Authentication requires that the client obtains a Kerberos ticket from Active Directory. To check whether the web browser obtained a Kerberos ticket for your web server, perform the following steps:

1. Log on to a Windows computer that is joined to the domain. Log on as a domain user.

2. Install the Microsoft KERBTRAY utility. The setup should be available at the following URL:

   http://download.microsoft.com/download/win2000platform/kerbtray/1.00.0.1/NT5/EN-US/kerbtray_setup.exe.

3. Launch KERBTRAY.EXE. This will install an icon in your taskbar tray.

4. Access the protected web site from Internet Explorer using the server name in the URL rather than the IP address.

Example: https://myserver

5. Double click on the KERBTRAY icon in the taskbar tray. This will show a dialog displaying all Kerberos tickets.

6. Look for the name of your domain in the list of the tickets. Under your domain look for service principal that will look like HTTP/myserver.mydomain.com.

## KLIST Linux or Unix Utility

The klist Linux or Unix utility, part of the krb5-client package, may be used to check the Kerberos keytab file on the Linux or Unix system. The output of this command will display all service principal tickets that are contained in the keytab file. This command may be unavailable on some systems. To use klist to examine the contents of a Kerberos keytab file:

**NOTE:** Replace myserver with your own server name and mydomain.com with your domain name.

1. Locate the Kerberos keytab file for the protected web site. You may need to find the Krb5Keytab directive in the Apache configuration file.

2. Execute the klist –k <keytab file name> command.
```
# klist -k krb5_myserver.keytab
```

3. Verify that the correct service principal names are displayed. The names to look for are HTTP/myserver@MYDOMAIN.COM and HTTP/myserver.mydomain.com@MYDOMAIN.COM. It is normal to see multiple entries for the same name.

   Example output:
```
# klist -k krb5_myserver.keytab
Keytab name: FILE:krb5_myserver.keytab
KVNO Principal
---- --------------------------------------------------------------------
--------------
   6 HTTP/myserver@MYDOMAIN.COM
   6 HTTP/myserver@MYDOMAIN.COM
   6 HTTP/myserver@MYDOMAIN.COM
   6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
   6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
   6 HTTP/myserver.mydomain.com@MYDOMAIN.COM
```

If the service principal names are not correct, go back to the Generate Kerberos keytab file step above to generate a new Kerberos keytab file.

## Common Issues

As Authentication problems may be difficult to diagnose, start with double checking all the configuration parameters including the validity of the generated Kerberos keytab file. If all the configuration parameters appear to be correct then examine the list of common problems below.

| Problem | Explanation |
|---|---|
| System clock out of sync | For Kerberos authentication to work the system clocks on all involved systems must not be more than 5 minutes apart. Make sure that the time on the Active Directory server, Linux or Unix web server and the client are synchronized. |
| User accessing the web site is not in the configured group. | Check the user's group membership using /opt/likewise/bin/lw-list-groups-for-user. |
| Internet Explorer web browser does not consider the URL as part of the Local Intranet zone. | This problem is common if the web site is accessed using a URL that includes the full domain name such as https://myserver.mydomain.com. Internet Explorer will only try to obtain Kerberos tickets for websites that are in the Local Intranet zone. Try accessing the web site using just the server name, for example https://myserver. Alternatively, you can add the URL to a list of Local Intranet sites using the Sites/Advanced buttons in the Internet Options dialog on the Security tab. |
| Service Principal Name of the web site is mapped to more than one object in the Active Directory | This is a rather rare problem, but very difficult to diagnose because of lack of good error that is returned to the web server. This can happen if the ktpass Windows utility was used on the Domain Controller to generate a Kerberos keytab file.<br><br>To diagnose this problem log onto the Active Directory domain controller and open the Event Viewer. Look for event of type=Error, source=KDC, and event ID=11. The text of the event will be similar to the below message.<br>There are multiple accounts with name HTTP/myserver.mydomain.com of type DS_SERVICE_PRINCIPAL_NAME.<br><br>Fixing this problem will require locating the computer or user objects used to map the service principal name in the Active Directory. The Active Directory Users and Computers MMC snap-in allows running custom LDAP queries. Use a LDAP query similar to this one "(servicePrincipalName=HTTP/myserver.mydomain.com)" to locate the Active Directory objects. Once object have been located then the spurious User object may be deleted. If the object cannot be deleted then use the ADSI Edit MMC snap-in to manually remove the "HTTP/myserver.mydomain.com" string from the servicePrincipalName object property. |