IdPMultipleLDAP

Tuning Shibboleth 2 IdP for multiple LDAP servers

Disjunct LDAP access

It's easy to configure an IdP to simultaneously use multiple **LDAP directories with different user bases**. However, be careful that the same user key is not present in both directories, or you might end up releasing combined and incorrect attribute sets for some users. You'll need to point to both LDAP servers in two ways: the attribute resolver for attributes, and authentication with both directories. The attribute query will try both directories no matter which directory contains the user. There is no way for the IdP to guess which directory the user is in, so it can only try them in the order you define them.

For sanity's sake, you shouldn't use one IdP to represent disparate security domains.

Authentication

JAAS, which the IdP uses, supports authentication chaining. See their documentation for more information.

The authentication process may try one or both directories. There is no way for the IdP to guess which directory the user is in, so it can only try them in the order you define them.

Attributes

Define an LDAP data connector for each directory, and add two Dependency elements to every attribute you want to pull from both directories. The attribute query will try both directories no matter which directory contains the user. There is no way for the IdP to guess which directory the user is in, or whether the user is present in both.

Clustered LDAP access

In this scenario the goal is to configure the IdP to use multiple instances of **LDAP servers containing the same data** in order to have a clustered and therefore highly available Identity Provider.

The initial version of the following section was provided by Vladislav Nespor (ETHZ - Swiss Federal Institute of Technology).

Audience

This section is intended for administrators of the Shibboleth 2 IdP who plan to use more than one LDAP server in the JAAS configuration file ($SHIB_HOME/conf/login.config) and/or in the attribute resolver configuration file ($SHIB_HOME/conf/attribute-resolver.xml).

Round 1

To find out, if this document is relevant for you:

1. try to configure more than one LDAP server in login.config and/or attribute-resolver.xml
2. restart the IdP
3. select a suitable test resource/SP that will show you user attributes
4. check if you can authenticate and if you get attributes released. This step is just to verify, that there are no other problems.

Round 2

If you have made sure that the IdP's overall configuration is working:

1. replace the first LDAP server in your configuration with a host which is on the net, but no LDAP daemon is running on it
2. restart the IdP (remove old cookies from you browser)
3. try to authenticate again.

Round 3

If this still is working try the same procedure, but this time add a host which is not running at all (the host is down, or disconnected from the net).

If you experience any problems (authentication failures or missing attributes) in either the second or the third round, information given below might help you.

NOTE: Of course, you can also test with real LDAP servers and bring down the LDAP service for the first test, and the LDAP host for the second test.

Environment

The steps below are based on a practical experience with the installation and configuration of the Shibboleth IdP version 2.1.2 in the following environment:

OS	Solaris 10 Generic_137137-09
Java	1.6.0_11
Tomcat	6.0.18
Apache	2.2.11

Host property syntax

In the login.config, the LDAP authentication module edu.vt.middleware.ldap.jaas.LdapLoginModule supports various connection properties. One of them is the host property, where you can configure hostnames of the LDAP servers. The example in the Shibboleth documentation uses the syntax

```
host="ldap1.example.org ldap2.example.org ldap3.example.org"
```

This syntax is translated by the LdapLoginModule into Java property

```
java.naming.provider.url=ldap://ldap1.example.org ldap2.example.org ldap3.
example.org
```

Note that the *ldap://* part is added only at the beginning of the server list.
(You can observe this in the log file idp-process.log, if you set the log level to ALL of the edu.vt.middleware.ldap property in the logging.xml configuration file).

NOTE: If you do not explicitly set the port property in the configuration, the LdapLoginModule also adds the standard port 389 at the end of the server list in the java.naming.provider.url property.)

In this case the Java client would try to contact the first server in the list. If this server is running a LDAP daemon, the LDAP client would establish a connection to the LDAP service, and it would continue with the authentication or with the attribute query. But if the LDAP daemon is down on the first server, the client would have problems to switch to the next LDAP server, as URL addresses of the following servers are not complete.

The host property syntax, which works in this case, includes full URL addresses for all LDAP servers. (You can add also the ports, if the servers do not use the standard LDAP ports 389 or 636). For example

```
host="ldap://ldap1.example.org ldap://ldap2.example.org ldap://ldap3.
example.org"
```

will be translated by the LdapLoginModule into

```
java.naming.provider.url=ldap://ldap1.example.org ldap://ldap2.example.org
ldap://ldap3.example.org
```

and the Java client would be able to switch to the next server, if the LDAP daemon is down on the first host. For the correct syntax of the java.naming.provider.url property, see the Java LDAP documentation.

Connection timeout

The LDAP service provider for JNDI defines the com.sun.jndi.ldap.connect.timeout environment property. The value is the connection timeout in milliseconds. The LDAP service provider aborts the connection attempt within this connection timeout, if it can not establish a network connection with the LDAP server. The default connection timeout is the network timeout (two hours in the case of TCP). For more information see the section 3.4 in the JNDI guide.

If the first LDAP server, configured either in login.config or attribute-resolver.xml, is down (the TCP connection can not be established), the attempt will be interrupted by Shibboleth (some internal timeout) resulting in an authentication failure or a failure of the attribute resolver (missing attributes).

In the case of the attribute resolver, the problem can be solved by defining the LDAPProperty in the LDAP data connector section in attribute-resolver.xml.
For example:

Samle LDAP DataConnector using multiple LDAP servers

```
<resolver:DataConnector id="myLDAP"
        xsi:type="LDAPDirectory"
        xmlns="urn:mace:shibboleth:2.0:resolver:dc"
        ldapURL="ldap://ldap1.example.org ldap://ldap2.example.org
ldap://ldap3.example.org"
        baseDN="..."
        principal="..."
        principalCredential="...">

        <FilterTemplate>
            <![CDATA[
                (uid=$requestContext.principalName)
            ]]>
        </FilterTemplate>

        <LDAPProperty name="com.sun.jndi.ldap.connect.timeout" value="500"
/>
    </resolver:DataConnector>
```

Unfortunately, for the moment there is no possibility to define the com.sun.jndi.ldap.connect.timeout environment property as a parameter of the edu.vt.middleware.ldap.jaas.LdapLoginModule in login.config. Neither can this property be defined as a system property on the command line, e.g in CATALINA_OPTS.
Therefore, the only possibility is to include com.sun.jndi.ldap.connect.timeout in the JNDI application file jndi.properties (see Java environment properties).
The jndi.properties file should be in the applications classpath, or in the $JRE_HOME/lib directory (the directory which contains the Java Runtime Environment, usually $JAVA_HOME/jre). For example, the jndi.properties could contain a line

```
com.sun.jndi.ldap.connect.timeout=500
```

If the jndi.properties file is used, it is not necessary to define the LDAPProperty in the attribute-resolver.xml configuration file, as explained above.

NOTE: If you use DNS aliases as LDAP hostnames in you configuration, and you use Java version 1.5 and lower, you may also consider to set the environment property:

```
sun.net.inetaddr.ttl=0
```

in the JNDI application file jndi.properties. This property indicates the caching policy for successful name lookups from the name service. The default value of this property is -1 ("cache forever") for Sun Java JVMs with versions 1.5 and lower, and 30 seconds for version 1.6 if a security manager is not set. If the IP address of any of the LDAP servers changes in the name service, and the sun.net.inetaddr.ttl is -1, then you would need to restart Shibboleth IdP in order to activate this change in the application.