Configuring an 2.x IdP to use StoredID Connector

The IdP 2.x installation manual had been recommending to use the ComputeID connector to generate eduPersonTargetedID values. While that made the setup simpler (no database setup needed), it has the consequence that there is no database of eduPersonTargetedID values issued.

This page describes how to change the IdP configuration to use a StoredID Connector instead to store the values in a database.

Earlier versions of the manual for Upgrading a 2.x IdP to 3.x were instructing to follow the instructions here when performing an upgrade from a 2.x IdP that was using ComputedIdConnector for eduPersonTargetedID (i.e., not storing the values in a database).

While it is still strongly recommended to store the values in a database, it is no longer deemed necessary to change the configuration of the version 2 IdP, as the version 3 IdP would be producing the same values.

However, if the existing persistent ID values are not stored in a database, it is crucial to use the identical salt value on the old 2.x and the new 3.x IdP.

This document is left here as a resource to configure a IdP 2.x IdP to store the persistent ID values in a database, but it is no longer a prerequisite for upgrading to IdP 3.x.

Motivation

As in the migration to IdP v3, there would be also a migration from eduPersonTargetedID (used as an attribute) to SAML2 Persistent NameID (used as the NameID in SAML assertions), there will also be a need to switch to a different generator of persistent IDs. The new generator (used with SAML2 Persistent NameIDs) can be configured to use the same database and reuse persistent IDs already assigned. When configured with identical salt, for matching usernames, it would also generate same values as the previous generator.

Earlier versions of this document were incorrectly stating these values would be different. The wording in the rest of the document is kept with the original assumption the values would be different.

Therefore, to ensure a smooth migration, it is necessary to re-configure eduPersonTargetedID on the existing IdP to switch to using a database (with the StoredID connector which generates same values as the ComputeId connector) and populate the database with all persistent IDs already in use.

With this change in place, any persistent ID passed in an SSO session would be recorded in the database. To cover all persistent IDs ever issued, it would be possible to:

- scan the IdP audit log and extract the combination of all user names and Service Provider entityIDs that ever received a dynamically computed persistent ID.
- use this list with automated tools that trigger the IdP into re-calculating the persistent ID - and with the change in place, storing it in the database.

After that, the migration to IdPv3 would preserve persistent IDs for all users.

Assumptions

This manual expects an IdP 2.x installation based on the manual Installing a Shibboleth 2.x IdP.

It expects a local MySQL database server has been already set up and a database created (for storing auEduPersonSharedToken values).

It also expects MySQL JDBC driver has been already installed in Tomcat.

Fixing missing assumptions - MySQL database

If the MySQL-related assumptions above do not hold, the following steps would make them hold:

- Install MySQL server:

```
yum install mysql mysql-server
service mysqld start
chkconfig mysqld on
```

- Create MySQL database and user account (substite an appropriate password here):

```
# mysql
mysql> CREATE DATABASE idp_db;
mysql> create user 'idp_admin'@'localhost' identified by 'IDP_ADMIN_PASSWORD';
mysql> grant all privileges on idp_db.* to 'idp_admin'@'localhost';
```

- Install MySQL JDBC driver into Tomcat:

```
yum install mysql-connector-java
ln -s /usr/share/java/mysql-connector-java.jar /usr/share/java/$TOMCAT_NAME
service $TOMCAT_NAME restart
```

And in addition to fixing missing assumptions, also explicitly install the MySQL driver into the IdP deployment directory - so that the `aacli.sh` tool used later can find the driver as well:

```
ln -s /usr/share/java/mysql-connector-java.jar /opt/shibboleth-idp/lib/
```

Creating MySQL table

The persistent ID values are stored in the `shibpid` table.

For simplicity, we recommend creating the table in the `idp_db` database which already holds the `auEduPersonSharedToken` values (in the `tb_st` table).

Create the table with the following DDL (reused from https://wiki.shibboleth.net/confluence/display/SHIB2/StoredIDDataConnectorDDL): - run `mysql idb_db` and enter:

```
CREATE TABLE IF NOT EXISTS shibpid (
  localEntity TEXT NOT NULL,
  peerEntity TEXT NOT NULL,
  principalName VARCHAR(255) NOT NULL DEFAULT '',
  localId VARCHAR(255) NOT NULL,
  persistentId VARCHAR(36) NOT NULL,
  peerProvidedId VARCHAR(255) DEFAULT NULL,
  creationDate timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  deactivationDate TIMESTAMP NULL DEFAULT NULL,
  KEY persistentId (persistentId),
  KEY persistentId_2 (persistentId, deactivationDate),
  KEY localEntity (localEntity(16), peerEntity(16), localId),
  KEY localEntity_2 (localEntity(16), peerEntity(16),
    localId, deactivationDate)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Changes to Attribute Resolver

Edit `/opt/shibboleth-idp/conf/attribute-resolver.xml` and make the following changes:

- Comment out the existing `computedID` connnector.
- Copy in the following definition of the `storedID` connector:

```
<resolver:DataConnector xsi:type="dc:StoredId" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
                        id="storedID"
                        sourceAttributeID="uid"
                        generatedAttributeID="computedID"
                        salt="ThisIsRandomText">

     <resolver:Dependency ref="myLDAP" />

     <ApplicationManagedConnection jdbcDriver="com.mysql.jdbc.Driver"
                                   jdbcURL="jdbc:mysql://localhost/idp_db?autoReconnect=true&amp;
sessionVariables=wait_timeout=31536000"
                                   jdbcUserName="idp_admin"
                                   jdbcPassword="IDP_ADMIN_PASSWORD" />

</resolver:DataConnector>
```

- Adjust the definition the same way the computeID connector definition was adjusted (as per eduPersonTargetedID section in the IdP 2.x install manual):
    - In particular, reusing the `sourceAttributeID` and `salt` values from the existing `computeID` connector.
- Change the dependency in the `eduPersonTargetedID` attribute definition to the new connector:

```
    <resolver:AttributeDefinition xsi:type="ad:SAML2NameID" id="eduPersonTargetedID"
                                  nameIdFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
sourceAttributeID="computedID">
        <resolver:Dependency ref="storedID" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1XMLObject" name="urn:oid:
1.3.6.1.4.1.5923.1.1.1.10" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2XMLObject" name="urn:oid:
1.3.6.1.4.1.5923.1.1.1.10" friendlyName="eduPersonTargetedID" />
    </resolver:AttributeDefinition>
```

- Restart Tomcat and test your IdP against the Tuakiri Attribute Validator: https://attributes.tuakiri.ac.nz/

Triggering recalculation of Persistent ID values

With the IdP now configured to store Persistent ID values in the database, any newly calculated values are stored in the database as they are used. However, to ensure all users keep their identity at all services they ever accessed, we would either need to wait until they log into all of the services again (yes, unrealistic), or we trigger the IdP to act as if the user logged into the service. For that, we use the command-line interface to the Attribute Authority - a `acli.sh`.

- First, we can the IdP logs to get all unique combinations of username and service provider entityID that were ever used in sessions where eduPersonTargetedID values were issued:
  (the following code assumes you use the default logging format in the IdP-Audit log)

```
 grep -h '[,|]eduPersonTargetedID[,|]' /opt/shibboleth-idp/logs/idp-audit*.log | cut -d '|' -f 4,9 | tr
'|' ' ' | sort -u > principal-sp-pairs.txt
```

- We check the number of records found:

```
wc -l principal-sp-pairs.txt
```

- Check the file actually contains pairs of usernames and Service Provider entity IDs.
- We do a dry run with the first entry:

```
time head -n 1 principal-sp-pairs.txt | while read SP PRINCIPAL ; do /opt/shibboleth-idp/bin/aacli.sh --
principal "$PRINCIPAL" --requester "$SP" ; done
```

- And we run the command for the full sequence (sorry, may take significant amount of time - up to 10s per invocation):

```
cat principal-sp-pairs.txt | while read SP PRINCIPAL ; do /opt/shibboleth-idp/bin/aacli.sh --principal
"$PRINCIPAL" --requester "$SP" ; done
```

- And check the database: connect to the `idp_db` MySQL database and run:

```
select count(*) from shibpid;
```

- The number should be greater or equal to the number of records found above (greater if there are additional sessions created in the meantime - new user-service provider combinations).

Troubleshooting aacli.sh invocations

The `aacli.sh` tool may fail to run for a number of reasons. Here are the most common ones - and possible remedies:

- ClassNotFoundException: org.apache.xerces.util.SecurityManager
  - This is because the IdP is using "endorsed" Xerces XML parser classes.
  - This is no longer needed - and is even discouraged because of a vulnerability in the XML parser: http://shibboleth.net/community/advisories/secadv_20141103.txt
  - The solution is to edit `$IDP_HOME/conf/internal.xml` and changethe bean definition defined under the `http://apache.org/xml/properties/security-manager` from

    ```
    <bean class="org.apache.xerces.util.SecurityManager"/>
    ```

    to

    ```
    <bean class="com.sun.org.apache.xerces.internal.util.SecurityManager"/>
    ```

- Class com.mysql.jdbc.Driver not found:
  - The JDBC driver has not been installed in `/opt/shibboleth-idp/lib`
  - Use the above documentation to install the MySQL driver there.

- JNDI database resource not found (`Unable to retrieve data source for data connector IDPDB from JNDI location java:comp/env/jdbc/IDP`)
  - This happens when the IdP (attribute resolver) uses Container Managed database connections.
  - Unfortunately, aacli.sh only supports Application Managed connection.
  - The only workaround is to (temporarily) change the attribute resolver configuration to use Application Managed connections.

- The principal-sp-pairs.txt file contains garbage (or no data)
  - Possible, the log format has been customized in this IdP deployment.
  - Solution: adjust the parsing code accordingly.