ZooKeeper SSL User Guide

Introduction

This document describes how to use SSL feature of ZooKeeper.

By default network communication of ZooKeeper isn't encrypted. However, each user and service can leverage the SSL feature and/or custom authentication implementation in order to use ZooKeeper in secure mode.

Netty communication

ZooKeeper was initially designed and implemented using the Java NIO package. Later on, we added Netty to optionally replace NIO, since Netty better supports SSL. SSL is only supported on top of Netty communication, which means if you want to use SSL you have to enable Netty. We will discuss how to do it in the following section.

SSL

It's been added in ZOOKEEPER-2125.

Client-Server Communication

The communication between a ZooKeeper client and a server has Netty and SSL support. Note that Netty needs to be enabled to use SSL.

Client

ZooKeeper client can use Netty by setting property:

Java system property
`zookeeper.clientCnxnSocket="org.apache.zookeeper.ClientCnxnSocketNetty"`

In order to do secure communication on client, set this property:

Java system property
`zookeeper.client.secure=true`

Note that with "secure" property set the client could and should only connect to server's "`secureClientPort`" which will be described shortly.

Then set up keystore and truststore environment by setting the following properties:

Java system property

```
zookeeper.ssl.keyStore.location="/path/to/your/keystore"
zookeeper.ssl.keyStore.password="keystore_password"
zookeeper.ssl.trustStore.location="/path/to/your/truststore"
zookeeper.ssl.trustStore.password="truststore_password"
```

Server

ZooKeeper server can use Netty by setting this property:

Java system property

```
zookeeper.serverCnxnFactory="org.apache.zookeeper.server.NettyServerCnxnFactory"
```

ZooKeeper server also needs to provide a listening port to accept secure client connections. This port is different from and running in parallel with the known “clientPort”. It should be added in “zoo.cfg”:

zoo.cfg

```
...
secureClientPort=2281
```

All secure clients (mentioned above) should connect to this port.

Then set up keystore and truststore environment like what client does.

Example

An example setup for running bin/zkServer.sh:

environmental variable

```
export SERVER_JVMFLAGS="
-Dzookeeper.serverCnxnFactory=org.apache.zookeeper.server.NettyServerCnxnFactory
-Dzookeeper.ssl.keyStore.location=/root/zookeeper/ssl/testKeyStore.jks
-Dzookeeper.ssl.keyStore.password=testpass
-Dzookeeper.ssl.trustStore.location=/root/zookeeper/ssl/testTrustStore.jks
-Dzookeeper.ssl.trustStore.password=testpass"
```

and set additionally in “zoo.cfg”:

zoo.cfg

```
 …
 secureClientPort=2281
```

For bin/zkCli.sh:

environmental variable

```
export CLIENT_JVMFLAGS="
-Dzookeeper.clientCnxnSocket=org.apache.zookeeper.ClientCnxnSocketNetty
-Dzookeeper.client.secure=true
-Dzookeeper.ssl.keyStore.location=/root/zookeeper/ssl/testKeyStore.jks
-Dzookeeper.ssl.keyStore.password=testpass
-Dzookeeper.ssl.trustStore.location=/root/zookeeper/ssl/testTrustStore.jks
-Dzookeeper.ssl.trustStore.password=testpass"
```

Start the ZK server, and then connect client to server's port 2281 should work like normal.

Quorum

There is currently no support for SSL for the communication between ZooKeeper servers.

Authentication

It's been added in ZOOKEEPER-2123.

When connecting to ZooKeeper via the secure port, the client is automatically authenticated with credentials associated with the client certificate. Specifically, the connection adds auth info with the scheme "x509" and the ACL ID set to the client certificate principal name.

X509 Authentication Provider

By default, authentication is performed by the X509AuthenticationProvider, corresponding to the auth scheme "x509." This is initialized with server certificates and trusted client certificates specified according to the following properties:

```
zookeeper.ssl.keyStore.location
zookeeper.ssl.keyStore.password
zookeeper.ssl.trustStore.location
zookeeper.ssl.trustStore.password
```

The keyStore JKS file contains the server certificate and private key. This certificate needs to be trusted by the clients, i.e. include the server's certificate or its CA in the client's trustStore JKS files. Meanwhile the trustStore JKS file on the server contains the client certificates or CA to trust.

Once authentication is complete and a ZooKeeper session is established, the client may set ACLs against the "x509" scheme. x509 uses the client's authenticated X500 Principal as an ACL ID identity. The ACL expression is the exact X500 Principal name of an authenticated client.

Similar to the digest auth scheme, an x509 "superUser" can be configured by the server. Set the property `zookeeper.X509AuthenticationProvider.superUser` to an X500 Principal that corresponds to a client that should have full privileges to all znodes regardless of ACLs.

Custom Authentication Provider

ZooKeeper can be configured to use a different X509-based trust mechanism. This is useful in certificate key infrastructures that do not use JKS.

To specify a custom authentication provider, extend the `org.apache.zookeeper.server.auth.X509AuthenticationProvider`. It may be necessary to extend `javax.net.ssl.X509ExtendedKeyManager` and `javax.net.ssl.X509ExtendedTrustManager` to get the desired behavior from the SSL stack. Then override `X509AuthenticationProvider.getKeyManager()` and `getTrustManager()` so that the SSLEngine will pick up the custom implementation.

To configure the ZooKeeper server to use the custom provider for authentication, choose a scheme name and set the property `zookeeper.authProvider.[scheme]` to the fully-qualified class name of the custom implementation. This will load the provider into the ProviderRegistry. Then set the property `zookeeper.ssl.authProvider=[scheme]` and that provider will be used for secure authentication.