Secure ZooKeeper Client in Apache Knox

Introduction / Motivation

Apache Knox 0.14.0 introduces the ability to manage topology deployments across multiple gateway instances by modifying znode contents in Apache ZooKeeper.

Due to the sensitive nature of some of the content, and the potential for an untrusted actor to intentionally provide malicious configuration, it is highly recommended that the ZooKeeper node(s) require authentication, and that the znodes used by Knox have appropriate ACLs applied.

This article will describe the minimal configuration necessary to accomplish this, and demonstrate the application of digest-backed(for simplicity) SASL authentication.

SASL Authentication

SASL is a framework for applications to add authentication support by way of a variety of authentication mechanisms. ZooKeeper employs SASL.
The Knox remote configuration registry facility currently supports the Kerberos and DIGEST-MD5 mechanisms for ZooKeeper interactions.

The ZooKeeper server and client use the Java Authentication and Authorization Service (JAAS) and the associated configuration to apply authentication.

RemoteConfigurationRegistryClient Configuration

Knox clients for remote configuration registries (e.g., ZooKeeper) are configured in ***{GATEWAY_HOME}*/config/gateway-site.xml**.

Each client configuration is a single property, the name of which is prefixed with **gateway.remote.config.registry.** , and suffixed by the client identifier. The value of such a property, is a registry-type-specific set of semicolon-delimited properties for that client, including the type of registry with which it will interact.

This configuration is the means by which the supported authentication mechanisms can be specified for each client.

N.B., Due to a limitation of the ZooKeeper client, there can only be a single secure ZooKeeper client configured for Knox at any given time. This will be resolved subsequent to the release of ZooKeeper 3.6.0, such that multiple ZooKeeper clients with distinct authentication configuration can be defined.

The configuration for a single-node ZooKeeper, without any client authentication required, looks similar to this:

```
<property>
    <name>gateway.remote.config.registry.sandbox-zookeeper-client</name>
    <value>type=ZooKeeper;address=localhost:2181</value>
    <description>ZooKeeper configuration registry client details.</description>
</property>
```

This article will be limited to a single-node ZooKeeper for these reasons:

- The ZooKeeper available in the HDP Sandbox is a single-node.
- The focus of this article is on the secure client configuration, and ensemble-level security configuration is beyond that scope.

For interactions with a ZooKeeper that requires client authentication, some additional properties need to be added to the client configuration.

This is the DIGEST-MD5 configuration:

```
<property>
    <name>gateway.remote.config.registry.sandbox-zookeeper-client</name>
    <value>type=ZooKeeper;address=localhost:2181;authType=Digest;principal=myzkuser;credentialAlias=myzkpass<
/value>
    <description>ZooKeeper configuration registry client details.</description>
</property>
```

You'll notice that the following have been added to the property value:

- **authType** The *Digest* value is what specifies that the client will attempt to authenticate via the DIGEST-MD5 mechanism.
- **principal** The identity, as which the client is requesting to be authenticated.
- **credentialAlias** A gateway alias for the password associated with the principal.

More details about these client configurations are available in the User Guide.

ZNode ACLs

By default, znodes have effectively no restrictions (i.e., world:anyone:cdrwa) on their ability to be modified. This poses a problem for an application like Knox, which needs to trust the content of one or more znodes. If anyone can create configuration, then there is the potential that Knox could consume and apply malicious configuration. Measures need to be taken to ensure that the ability to create or modify configuration is limited to trusted actors. Once such measure is the application of more restrictive ACLs.

An ACL specifies a set of znode permissions for a user. At the very least, write permissions should be limited to authenticated users. I would further suggest that read permissions be similarly limited, since the gateway configuration does contain Hadoop cluster network details.

Fortunately, when the Knox ZooKeeper client is configured for authentication, the remote registry monitor will make appropriate changes to the ACLs for the znodes with which it is concerned.

Try it!

1. HDP Sandbox (https://hortonworks.com/downloads/#sandbox)
2. Configure ZooKeeper JAAS
    a. Create JAAS configuration (e.g., sasl-zk-jaas.conf) with content:

    ```
    Server {     org.apache.zookeeper.server.auth.DigestLoginModule required     user_knox="knoxtest"; };
    ```

    b. Edit **zookeeper-env.sh** (e.g., /usr/hdp/current/zookeeper-server/conf/zookeeper-env.sh), adding a reference to the JAAS config you just created

    export SERVER_JVMFLAGS="-Xmx1024m -Djava.security.auth.login.config=/sasl-zk-jaas.conf

    c. Edit **zoo.cfg** (e.g., /usr/hdp/current/zookeeper-server/conf/zoo.cfg), adding the following:

    authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
    requireClientAuthScheme=sasl

    d. Restart ZooKeeper (e.g., '/usr/hdp/current/zookeeper-server/bin/zkServer.sh start')

3. Create the Knox configuration znodes
    a. Create a client JAAS config (e.g., sasl-zk-client-jaas.conf)

    Client {
    org.apache.zookeeper.server.auth.DigestLoginModule required
    username="knox"
    password="knoxtest";
    }

    b. **export CLIENT_JVMFLAGS=-Djava.security.auth.login.config=/sasl-zk-client-jaas.conf**

    c. ***{ZOOKEEPER_HOME}*/bin/zkCli.sh -server sandbox.hortonworks.com:2181**

    create /knox "1"
    create /knox/config "1"
    create /knox/shared-providers "1"
    create /knox/config/descriptors "1"
    setAcl /knox/config/shared-providers sasl:knox:cdrwa
    setAcl /knox/config/descriptors sasl:knox:cdrwa

4. Configure the client in ***{GATEWAY_HOME}*/conf/gateway-site.xml** (no authentication config)

```
<property>
    <name>gateway.remote.config.registry.sandbox-zookeeper-client</name>
    <value>type=ZooKeeper;address=localhost:2181</value>
    <description>ZooKeeper configuration registry client details.</description>
</property>
```

5. **bin/knoxcli.sh** registry client should not work because of no authentication

   ***{GATEWAY_HOME}*/bin/knoxcli.sh upload-descriptor mysandbox.yml --registry-client sandbox-zookeeper-client**

6. Create the ZK auth password alias

   ***{GATEWAY_HOME}*/bin/knoxcli.sh create-alias sandbox-zk-pwd --value knoxtest**

7. Add auth to the client config in ***{GATEWAY_HOME}*/conf/gateway-site.xml**

   ```
   <property>
      <name>gateway.remote.config.registry.sandbox-zookeeper-client</name>
      <value>type=ZooKeeper;address=localhost:2181;authType=Digest;principal=knox;credentialAlias=sandbox-zk-pwd</value>
      <description>ZooKeeper configuration registry client details.</description>
   </property
   ```

8. **bin/knoxcli.sh** registry client should succeed now that authentication is configured

   ***{GATEWAY_HOME}*/bin/knoxcli.sh upload-descriptor mysandbox.yml --registry-client sandbox-zookeeper-client**

Summary

Hopefully, this helps provide some understanding of what is required to secure the interactions between the gateway and Apache ZooKeeper.

The [Remote Configuration Registry Clients](http://knox.apache.org/books/knox-0-14-0/user-guide.html#Remote+Configuration+Registry+Clients) section of the [User Guide](http://knox.apache.org/books/knox-0-14-0/user-guide.html) provides more information, including the details necessary to configure Kerberos authentication for these interactions.

Bonus Tip

If you get in a bad place with respect to ACLs and authentication errors, you can configure super user support:

1. **export ZK_CLASSPATH=/etc/zookeeper/conf/:/usr/hdp/current/zookeeper-server/lib/*:/usr/hdp/current/zookeeper-server/***

2. **java -cp $ZK_CLASSPATH org.apache.zookeeper.server.auth.DigestAuthenticationProvider super:superpwd**

(super:superpwd->super:**G+ys1zTeJy/1iLhgQS08pRQoMvo=**)

3. **SERVER_JVMFLAGS=-Dzookeeper.DigestAuthenticationProvider.superDigest=super:G+ys1zTeJy/1iLhgQS08pRQoMvo=**

4. Restart ZooKeeper