Security 3

Configuring NiFi Authentication and Proxying with Apache Knox

Date of Publish: 2018-12-18

https://docs.hortonworks.com/

Contents

Configuring NiFi Authentication and Proxying with Apache Knox

You can use the Apache Knox Gateway to provide authentication access security for your NiFi services. The Apache Knox Gateway (Knox) enables you to extend Apache Hadoop® services to users outside of a Hadoop cluster without reducing Hadoop security.

Knox also simplifies Hadoop security for users who access the cluster data and execute jobs. Knox integrates with identity management and single sign-on (SSO) systems used in enterprises, enabling you to use identities from these systems to access Hadoop clusters. For more information about Knox, see *Apache Knox Gateway Overview*.

Note:

If you are using Hortonworks Data Platform (HDP) 2.6.4 or earlier, HA proxying of NiFi is not supported.

Configuring NiFi for Knox Authentication

After you install NiFi, you must update the NiFi configurations in Apache Ambari.

About this task

Important:

We recommend that NiFi is installed on a different host than Knox.

Procedure

1. In **Advanced nifi-ambari-ssl-config**, the **Initial Admin Identity** value must specify a user that Apache Knox can authenticate.
2. In **Advanced nifi-ambari-ssl-config**, add a node identity for the Knox node:
   - <property name="Node Identity 1">CN=$NIFI_HOSTNAME, OU=NIFI</property>
   - <property name="Node Identity 2">CN=$NIFI_HOSTNAME, OU=NIFI</property>
   - <property name="Node Identity 3">CN=$NIFI_HOSTNAME, OU=NIFI</property>
   - <property name="Node Identity 4">CN=$KNOX_HOSTNAME, OU=KNOX</property>
3. Update the **nifi.web.proxy.context.path** property in **Advanced nifi-properties**:

   ```
   nifi.web.proxy.context.path=/$GATEWAY_CONTEXT/flow-management/nifi-app
   ```

   $GATEWAY_CONTEXT is the value in the **Advanced gateway-site gateway.path** field in the Ambari Configs for Knox.
4. Update the **nifi.web.proxy.host** property in **Advanced nifi-properties** with a comma-separated list of the host name and port for each Knox host, if you are deploying in a container or cloud environment.

   For example:

   ```
   knox-host1:18443, knox-host2:443
   ```

Configuring Knox for NiFi

Preparing to Generate Knox Certificates using the TLS Toolkit

Proxies must communicate securely with NiFi using two-way SSL. To set up two-way SSL, you must generate certificates for Knox to use when communicating with NiFi. You can do this by using the TLS Toolkit. Use these steps to create a configuration for the TLS Toolkit to generate the certificates for Knox.

Procedure

1. As the Knox user, create a nifi-ca-config.json file on each Knox node, in a location accessible to Knox.

   For example, to create the file on a Knox node at /home/knox using the vi text editor, enter the following:

   ```
   sudo su - knox
   vi /home/knox/nifi-ca-config.json
   ```

2. Populate the nifi-ca-config.json file with the following information:

   ```
   {
     "dn" : "CN=$KNOX_HOSTNAME, OU=KNOX",
     "keyStore" : "/home/knox/knox-nifi-keystore.jks",
     "keyStoreType" : "jks",
     "keyStorePassword" : "$KEYSTORE_PASSWORD",
     "keyPassword" : "$KEY_PASSWORD",
     "token" : "$NIFI_CA_TOKEN_VALUE",
     "caHostname" : "$NIFI_CA_HOSTNAME",
     "port" : $NIFI_CA_PORT,
     "trustStore" : "/home/knox/knox-nifi-truststore.jks",
     "trustStorePassword" : "$TRUSTSTORE_PASSWORD",
     "trustStoreType" : "jks"
   }
   ```

   where:

   - $KNOX_HOSTNAME is the FQDN for the cluster node on which Knox is installed.
   - $KEYSTORE_PASSWORD is the password you want the TLS Toolkit to use for the KeyStore and TrustStore for Knox when communicating with NiFi.
   - $KEY_PASSWORD is required when SSL is enabled. If you are using the NiFi Certificate Authority (CA), this value is automatically generated if the field is left blank. You can find this value in the **Key password** field in the NiFi **Advanced nifi-ambari-ssl-config** configuration section.
   - $NIFI_CA_TOKEN_VALUE is the token that NiFi CA uses to verify a NiFi node identity before issuing certificates. You can find this value in the **NiFi CA Token** field in the NiFi **Advanced nifi-ambari-ssl-config** configuration section.
   - $NIFI_CA_HOSTNAME is the FQDN for the cluster node on which the NiFi CA is installed.
   - $NIFI_CA_PORT is the port used by the NiFi CA.

     You can find this value in the **NiFi Certificate Authority port** field in the NiFi **Advanced nifi-ambari-ssl-config** configuration section.
   - $TRUSTSTORE_PASSWORD is the truststore password.

   Note: You can set keyStorePassword, keyPassword, and trustStorePassword to the Knox Master Secret to make it easier to import the keyStore and trustStore created by the NiFi Certificate Authority into the Knox keystore.

Example

Example nifi-ca-config.json file

```
{
  "dn" : "CN=slo-hdf-test5.field.hortonworks.com, OU=KNOX",
  "domainAlternativeNames" : null,
```

```
  "keyStore" : "/home/knox/knox-nifi-keystore.jks",
  "keyStoreType" : "jks",
  "keyStorePassword" : "admin",
  "keyPassword" : "admin",
  "token" : "token",
  "caHostname" : "slo-hdf-test5.field.hortonworks.com",
  "port" : 10443,
  "dnPrefix" : "CN=",
  "dnSuffix" : ", OU=NIFI",
  "reorderDn" : true,
  "trustStore" : "/home/knox/knox-nifi-truststore.jks",
  "trustStorePassword" : "admin",
  "trustStoreType" : "jks"
}
```

Generating Knox Certificates Using the TLS Toolkit

You can generate the certificates used by Knox when proxying NiFi using the TLS Toolkit.

Before you begin

Ensure that Java is set correctly for your environment.

Procedure

1. As the Knox user, start the TLS Toolkit. For the location of the TLS Toolkit, see the *HDF Release Notes* for release-specific download information.

   For example:

   ```
   /var/lib/ambari-agent/tmp/nifi-toolkit-1.7.0.3.2.0.0-520/bin/tls-
   toolkit.sh
   client --subjectAlternativeNames "CN=$KNOX_HOSTNAME, OU=KNOX"
   -F
   -f /home/knox/nifi-ca-config.json
   ```

   The toolkit requests a new certificate and creates two new files containing the keystore and truststore:

   ```
   /home/knox/knox-nifi-keystore.jks
   /home/knox/knox-nifi-truststore.jks
   ```

2. Import the Knox certificate for NiFi into the Knox gateway.jks file:

   ```
   keytool
   -importkeystore
   -srckeystore /home/knox/knox-nifi-keystore.jks
   -destkeystore /usr/hdf/current/knox-server/data/security/keystores/
   gateway.jks
   -deststoretype JKS
   -srcstorepass $KEYSTORE_PASSWORD
   -deststorepass $KNOX_MASTER_PASSWORD
   ```

   The gateway.jks file now contains a PrivateKeyEntry for NiFi.

3. Import the NiFi CA truststore into the Knox gateway.jks file:

   ```
   keytool
   -importkeystore
   ```

```
-srckeystore /home/knox/knox-nifi-truststore.jks
-destkeystore /usr/hdf/current/knox-server/data/security/keystores/
gateway.jks
-deststoretype JKS
-srcstorepass $TRUSTSTORE_PASSWORD
-deststorepass $KNOX_MASTER_PASSWORD
```

The gateway.jks file should now contain a trustedCertEntry for NiFi.

4. Verify that the proper keys are in the gateway.jks file:

```
keytool
-keystore /usr/hdf/current/knox-server/data/security/keystores/gateway.jks
-storepass $KEYSTORE_PASSWORD
-list
-v
```

Results

You see nifi-key and nifi-cert in addition to the gateway-identity key.

Configuring the Knox SSO Topology

If you are proxying NiFi and authenticating with Knox SSO, you must make several edits to the Knox SSO topology. If you not authenticating with Knox SSO, these steps are not necessary.

Procedure

1. Navigate to **Advanced knoxsso-topology** and, in the KNOXSSO service definition, edit the Knox SSO token time-to-live value. For example, for a 10 hour time-to-live:

```
<param>
   <name>knoxsso.token.ttl</name>
   <value>36000000</value>
</param>
```

2. Update the knoxsso.redirect.whitelist.regex property with a regex value that represents the host or domain in which the NiFi host is running. If the knoxsso.redirect.whitelist.regex property does not exist, you must add it. For example:

```
<param>
   <name>knoxsso.redirect.whitelist.regex</name>
   <value>^https?:\/\/(hdf-test\.field\.hortonworks\.com|localhost|
127\.0\.0\.1|0:0:0:0:0:0:0:1|::1):[0-9].*$</value>
</param>
```

Creating an Advanced Topology

Procedure

1. As the Knox user, create flow-management.xml in /usr/hdf/current/knox-server/conf/topologies.
2. Add the following content to flow-management.xml:

If you have modified the ShiroProvider in the topology specified in the **Advanced Topology** section of the Ambari Knox configs, and want to authenticate Flow Management topology users in the same manner as Advanced Topology users, ensure that those modifications are reflected in your added content.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<topology>
  <gateway>
    <provider>
      <role>authentication</role>
      <name>ShiroProvider</name>
      <enabled>true</enabled>
      <param>
        <name>sessionTimeout</name>
        <value>30</value>
      </param>
      <param>
        <name>redirectToUrl</name>
        <value>/gateway/knoxsso/knoxauth/login.html</value>
      </param>
      <param>
        <name>restrictedCookies</name>
        <value>rememberme,WWW-Authenticate</value>
      </param>
      <param>
        <name>main.ldapRealm</name>
        <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</value>
      </param>
      <param>
        <name>main.ldapContextFactory</name>

 <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapContextFactory</
value>
      </param>
      <param>
        <name>main.ldapRealm.contextFactory</name>
        <value>$ldapContextFactory</value>
      </param>
      <param>
        <name>main.ldapRealm.userDnTemplate</name>
        <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
      </param>
      <param>
        <name>main.ldapRealm.contextFactory.url</name>
        <value>ldap://localhost:33389</value>
      </param>
```

```
      <param>
        <name>main.ldapRealm.authenticationCachingEnabled</name>
        <value>false</value>
      </param>
      <param>
        <name>main.ldapRealm.contextFactory.authenticationMechanism</name>
        <value>simple</value>
      </param>
      <param>
        <name>urls./**</name>
        <value>authcBasic</value>
      </param>
    </provider>
    <provider>
      <role>identity-assertion</role>
      <name>Default</name>
      <enabled>true</enabled>
    </provider>
  </gateway>
  <service>
    <role>NIFI</role>
    <url>$NIFI_HTTP_SCHEME://$NIFI_HOST:$NIFI_HTTP_SCHEME_PORT</url>
    <param name="useTwoWaySsl" value="true"/>
  </service>
</topology>
```

Where:

- $NIFI_HTTP_SCHEME specifies whether NiFi is communicating over HTTP or HTTPS.
- $NIFI_HOST is the FQDN of the host machine on which NiFi is running.
- $NIFI_HTTP_SCHEME_PORT specifies the port on which NiFi is running. This value varies depending on HTTP or HTTPS usage. In **Advanced nifi-ambari-ssl-config**, if **Enable SSL?** is selected, use the port specified by **NiFi HTTP port (SSL)**, and if not, use the port specified by **NiFi HTTP port (non-SSL)**. The port values are available in **Advanced nifi-ambari-config**.

3. Save the configuration and restart Knox, or continue by configuring Knox SSO.

Configuring Knox SSO

Procedure

1. If you want to use Knox SSO authentication, perform the following steps:

   a. On each cluster node with Knox installed, replace the ShiroProvider federation provider in the flow-management.xml file with the following content:

```
<provider>
   <role>federation</role>
   <name>SSOCookieProvider</name>
   <enabled>true</enabled>
   <param>
      <name>sso.authentication.provider.url</name>
      <value>https://$KNOX_HOSTNAME:$KNOX_PORT/gateway/knoxsso/api/v1/
websso</value>
   </param>
</provider>
```

   where:

   - $KNOX_HOSTNAME is the FQDN of the Knox host.
   - $KNOX_PORT is the port Knox is using.

Your new flow-management.xml file looks similar to the following:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version
 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<topology>
  <gateway>
    <provider>
      <role>federation</role>
      <name>SSOCookieProvider</name>
      <enabled>true</enabled>
      <param>
        <name>sso.authentication.provider.url</name>
        <value>https://hdf-test.field.hortonworks.com:8443/gateway/
knoxsso/api/v1/websso</value>
      </param>
    </provider>
    <provider>
      <role>identity-assertion</role>
      <name>Default</name>
      <enabled>true</enabled>
    </provider>
  </gateway>
  <service>
    <role>NIFI</role>
    <url>https://hdf-test-nifi.field.hortonworks.com:9091</url>
    <param>
      <name>useTwoWaySsl</name>
      <value>true</value>
    </param>
  </service>
</topology>
```

b. If you want to access NiFi directly and still use Knox SSO:

- Export the Knox SSO certificate:

```
 $KNOX_INSTALL_DIR/bin/knoxcli.sh export-cert
```

- Set the following properties in the **Advanced nifi-properties** section in Ambari:

```
nifi.security.user.knox.url=https://$KNOX_HOSTNAME:$KNOX_PORT/
gateway/knoxsso/api/v1/websso
nifi.security.user.knox.publicKey=<path-to>/gateway-identity.pem
nifi.security.user.knox.cookieName=hadoop-jwt
```

```
nifi.security.user.knox.audiences=
```

The cookieName property must align with what is configured in Knox. The audiences property is used to only accept tokens from a particular audience. The audiences value is configured as part of Knox SSO.

2. Save the configuration and restart Knox.

Adding a NiFi Policy for Knox

A new NiFi installation includes the policies necessary for Knox. If you are configuring an existing NiFi installation to be proxied by Knox, you must manually create NiFi policies for Knox. You can create and add these policies using either NiFi or Apache Ranger.

Adding a Policy Using NiFi

You must add the Knox user and proxy policies only if you did not add the Knox cert DN to the list of node identities in **Advanced nifi-ambari-ssl-config** when NiFi was installed. You also must manually edit policies for the users who are going to log in to Knox and for the Knox node itself to be authorized as a proxy. At a minimum, the Knox node should be added to the policy for proxying user requests and the users.

Procedure

1. Create a user to represent the Knox identity in NiFi. Navigate to the NiFi **Global Menu | Users**. Click Add User . The value for the identify of this new user is the DN from the cert that Knox is using to communicate with NiFi.
2. From the NiFi **Global Menu | Policies**, select **proxy user requests** and add the Knox identity.
3. For component-level policies, on the root group, add permissions to "view the data" and to "modify the data" for the Knox identity.

Adding a Policy Using Ranger

If you are using Ranger, complete the following steps to add a policy for the Knox node user.

Procedure

1. Create a user to represent Knox in NiFi by taking the DN from the cert that Knox is using to communicate with NiFi.
2. For the component level policies, on the root group, add permissions to "view the data " and to "modify the data" for the Knox user:
   - Set WRITE to /proxy.
   - Set READ and WRITE to /data/process-group/<root-group-id>.

Accessing NiFi Using Knox

Procedure

1. Enter the following URL in a browser:

   ```
   https://$KNOX_HOST:$KNOX_PORT/$GATEWAY_CONTEXT/flow-management/nifi-app/nifi
   ```

   Where:

   - $KNOX_HOST:KNOX_PORT is the Knox host:gateway port.
   - $GATEWAY_CONTEXT is the gateway context, defined in the **gateway-site** configuration in Knox.
2. When prompted, enter your NiFi user name and password to be authenticated by Knox.

Results

Knox proxies the NiFi UI.