Hortonworks Data Platform

Configuring Storm for Kerberos Over Ambari

(September 30, 2015)

docs.cloudera.com

Hortonworks Data Platform: Configuring Storm for Kerberos Over Ambari

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain, free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to contact us directly to discuss your specific needs.

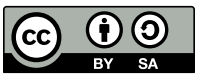

Table of Contents

List of Tables

1. Overview

This chapter describes how to configure Storm for Kerberos security on an Ambari-managed cluster.

For information about configuring Storm for Kerberos security on a cluster that is not managed by Ambari, see the Non-Ambari Cluster Installation Guide.

1.1. Prerequisites

Before you enable Kerberos, your cluster must meet the following prerequisites. (Note: Links point to Ambari version 2.1.2.0. If your cluster runs a different version of Ambari, refer to the Ambari document for your version of software.)

Prerequisite	References
Ambari-managed cluster with Storm installed and running. • Ambari Version 1.7.0.0 or later • Stack version HDP 2.2.0 or later	Installing, Configuring, and Deploying a HDP Cluster in *Automated Install with Ambari*
Key Distribution Center (KDC) server installed and running.	Installing and Configuring the KDC in the *Ambari Security Guide*
JCE installed on all hosts on the cluster (including the Ambari server).	Enabling Kerberos Security in the *Ambari Security Guide*

When all prerequisites are fulfilled, enable Kerberos security. For more information, see Running the Kerberos Security Wizard in the *Ambari Security Guide*.)

2. Designating a Storm Client Node

At this point in the configuration process there is no notion of a Storm client node (you won't be able to select "client" via Ambari).

To specify a Storm client node, choose one of the following two approaches, described in the following subsections:

- Dedicate or use an existing independent gateway node as a storm client
- Use one of your existing storm nodes (such as nimbus, supervisors, or drpc) as a client. Choose this option if you prefer not to add a gateway node for Storm.

2.1. Dedicate or Use an Existing Gateway Node

To dedicate or use an existing gateway node (edge node):

1. Install the storm package on the node:

   ```
   sudo yum install storm_<version>
   ```

   For example, for HDP 2.3:

   ```
   sudo yum install storm_2_3*
   ```

2. ***HDP 2.2 only***: Create a file at `/etc/storm/conf/client_jaas.conf`, and add the following entry to it:

   ```
   StormClient {
       com.sun.security.auth.module.Krb5LoginModule required
       useTicketCache=true
       renewTicket=true
       serviceName="nimbus"
       };
   ```

3. Add the following settings to the `/etc/storm/conf/storm.yaml` configuration file.

   *For **HDP 2.2***:

   ```
   nimbus.host: <nimbus-host>
   nimbus.thrift.port: 6667
   java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
   storm.thrift.transport: "backtype.storm.security.auth.kerberos.
   KerberosSaslTransportPlugin"
   ```

   where `<nimbus-host>` is the host that is running Nimbus. For example:

   ```
   nimbus.host: "c6401.ambari.apache.org"
   ```

   *For **HDP 2.3***:

```
nimbus.seeds: <nimbus-host-array>
nimbus.thrift.port: 6667
java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
storm.thrift.transport: "backtype.storm.security.auth.kerberos.
KerberosSaslTransportPlugin"
```

where <nimbus-host-array> is an array of hostnames running Nimbus. (The value should come from /etc/storm/conf/storm.yaml.) For example:

```
 nimbus.seeds: ["c6401.ambari.apache.org",
"c6402.ambari.apache.org"]
```

2.2. Use an Existing Storm Node

To use one of your existing Storm nodes (such as nimbus, supervisors, or drpc) as a Storm client node, complete the following steps for every user who requires Storm access (for example, to run Storm commands or deploy topologies):

1. Create a .storm directory in the user's home directory. For example, user john should have a directory called /home/john/.storm/.
2. Add the following settings to the /etc/storm/conf/storm.yaml configuration file:

   *For **HDP 2.2***:

   ```
   nimbus.host: <nimbus-host>
   nimbus.thrift.port: 6667
   java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
   storm.thrift.transport: "backtype.storm.security.auth.kerberos.
   KerberosSaslTransportPlugin"
   ```

   where <nimbus-host> is the host that is running Nimbus. For example:

   ```
    nimbus.host: "c6401.ambari.apache.org"
   ```

   *For **HDP 2.3***:

   ```
   nimbus.seeds: <nimbus-host-array>
   nimbus.thrift.port: 6667
   java.security.auth.login.config: "/etc/storm/conf/client_jaas.conf"
   storm.thrift.transport: "backtype.storm.security.auth.kerberos.
   KerberosSaslTransportPlugin"
   ```

   where <nimbus-host-array> is an array of hostnames running Nimbus (the value should come from /etc/storm/conf/storm.yaml). For example:

   ```
    nimbus.seeds: ["c6401.ambari.apache.org",
   "c6402.ambari.apache.org"]
   ```

As mentioned earlier, repeat these steps for every user who requires Storm access.

2.3. Running Storm Commands

After configuring the client/gateway node, run kinit (with the principal's keytab) before issuing Storm commands.

3. Running Workers as Users

In Storm secure mode, workers can run as the user (owner of the topology) who deployed the topology. To enable, complete the following steps:

1. (***HDP 2.2 only***) On all Storm nodes, change ownership of the `conf` directory from `storm` to `root`:

   ```
   sudo chown -R root:hadoop /etc/storm/conf
   ```

   This restricts access to the `worker-launcher.cfg` file in the `conf` directory. Users can still modify and update Storm configuration properties, because Ambari runs as `root`.

2. Make sure all users who are going to deploy topologies have a UNIX account on all of the Storm nodes. Workers will run under the UNIX account for topologies deployed by the user.

   Example: For user `testuser1` and principal `testuser1/c6401.ambari.apache.org`, make sure there is a corresponding `testuser1` UNIX account.

3. Add the following configuration under "Custom storm-site" in the Ambari Storm configuration screen:

   ```
   supervisor.run.worker.as.user : true
   ```

4. Restart Storm components.

4. Accessing the Storm UI

The Storm UI uses SPNEGO AUTH when in Kerberos mode.

Before accessing the UI, configure your browser for SPNEGO authorization, as shown in the following table.

Then `kinit` before accessing the Storm UI.

Table 4.1. Browser Settings for Storm UI

Browser	Configuration
Safari	No changes needed.
Firefox	1. Go to about:config and search for `network.negotiate-auth.trusted-uris`. 2. Double-click and add the following value: `"http://storm-ui-hostname:ui-port"` 3. Replace the `storm-ui-hostname` value with the hostname where your UI is running. 4. Replace the `ui-port` value with the Storm UI port.
Chrome	From the command line, issue: `google-chrome --auth-server-whitelist="<storm-ui-hostname>" --auth-negotiate-delegate-whitelist="<storm-ui-hostname>"`
Internet Explorer	• Configure trusted websites to include `"storm-ui-hostname"`. • Allow negotiation for the UI website.

5. Accessing the Storm UI (Active Directory Trust Configuration)

If your cluster is configured with Active Directory Trust, use the Active Directory ticket to communicate with MIT KDC for secure negotiation. Here are the additional configuration steps:

1. Make sure UI Kerberos authentication-to-local rules are configured properly. Once a principal from Active Directory is used for negotiation with MIT KDC, you need a rule to translate it to the local account on the Storm UI node. Many times those can be copied from `core-site.xml`.

   For example:

   ```
   ui.filter.params:
      "type": "kerberos"
      "kerberos.principal": "HTTP/nimbus.host1.com"
      "kerberos.keytab": "/vagrant/keytabs/http.keytab"
      "kerberos.name.rules": "RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/
   $MAPRED_USER/ RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/$HDFS_USER/DEFAULT"
   ```

   Note: Rules are listed as strings, and are not separated by commas.

2. Create mappings for MIT domain KDC and associated resources used for the domain, in this case Storm UI.

   On a Windows workstation, you would run the following commands from the command line:

   ```
   ksetup /AddKDC $DOMAIN $KDC
   ```

   ```
   ksetup /AddHostToRealmMap $hadoop_resource $Domain
   ```

   Note: this step adds registry entries in `HKLM\System\CurrentControlSet\Control\Lsa\Kerberos\HostToRealm`.

Troubleshooting

To troubleshoot configuration issues, try accessing the Storm UI within the cluster using the `curl` command.

For example:

```
curl -i --negotiate -u:anyUser -b ~/cookiejar.txt -c ~/
cookiejar.txt http://storm-ui-hostname:8080/api/v1/cluster/summary
```

This will help you determine whether the Kerberos UI configuration is working.

To isolate the issue, use Storm service keytabs and user principals.

Two other important things to check are:

- Make sure that the trust is working properly.
- Make sure that the encryption types match on both KDCs.

6. Storm Security Properties

The following table lists important Storm security properties.

Configuration Property	Description	Example
nimbus.authorizer	This is a pluggable authorizer for a Storm Nimbus node. SimpleACLAuthorizer is the default implementation. Note: Admins can also grant permissions via the Ranger authorizer UI. For more information, see the Ranger User's Guide.	"backtype.storm.security.auth.authorizer.SimpleACLAuthoriz
nimbus.admins	Add Nimbus admin users. These users will have super user permissions on all topologies deployed, and will be able to perform other admin operations (such as rebalance, activate, deactivate and kill), even if they are not the owners of the topology. By default, only users who deployed the topologies have access to admin operations such as rebalance, activate, deactivate, and kill.	"John" "Abc"

topology.users:	This and the following config can be added as part of the topology file. The users listed in this setting will have owner privileges for the specified topology.	`Config conf = new Config()`
		`conf.put(”topology.users”,Lists.`
		`newArrayList(“test_user1”,`
		`”test_user2"));`
		`StormSubmitter.`
		`submitTopology(topologyName,`
		` conf, builder.`
		`createTopology());`

topology.groups	Similar to topology.users. Use this to add group-level permissions to a topology.	`Config conf = new Config()`
		`conf.put(”topology.groups”,`
		`Lists.newArrayList(“test_group1”,`
		`”test_group2"));`
		`StormSubmitter.`
		`submitTopology(topologyName,`
		` conf, builder.`
		`createTopology());`

7. Known Issues

Issue: Ambari does not show the security configuration on the Storm configuration tab, so you cannot add users to nimbus.admins.

Workaround: To give permissions to other users, use `topology.users` or `topology.groups`.

Issue: In AD+MIT setup, when trying to access Nimbus on a Kerberized cluster a HTTP 413 full HEAD error is received. (STORM-633)

Workaround: Add `ui.header.buffer.bytes : “65536”` under “Custom storm-site" on the Ambari Storm configuration tab.

Issue: Log viewer. We recommend against creating HTTP principal keytabs for supervisors. This can cause the SPNEGO protocol to fail.

Workaround:

1. Add the HTTP principal for Storm supervisor nodes too. For example:

   ```
   sudo /usr/sbin/kadmin.local -q 'addprinc -randkey HTTP/
   <supervisor-hostname>
   ```

   where

   `<supervisor-hostname>` is your hostname and domain for Kerberos; for example: `supervisor1.host1.com@HOST1.COM`

2. Add this principal for all hosts that run supervisor machines.

   For example:

   ```
   sudo /usr/sbin/kadmin.local -q "ktadd -k /etc/security/keytabs/
   spnego.service.keytab HTTP/supervisor1.host1.com@HOST1.COM"
   ```

3. Add the newly created HTTP principals to the `spnego.service.keytab` file.

4. Make sure that the spnego.service.keytab file has “storm” user privileges for read operations.

5. Distribute this keytab to all supervisor hosts.

6. On the supervisor node, edit `/etc/storm/conf/storm.yaml`. Change the `ui.filter.parameters` as follows, replacing `<supervisor-hostname>` with the hostname of your supervisor process:

   ```
   "type": "kerberos"

   "kerberos.principal": "HTTP/<supervisor-hostname>"

   "kerberos.keytab": "/vagrant/keytabs/http.keytab"
   ```

7. On each supervisor machine change the Kerberos.principal hostname to that supervisor's hostname.

8. Restart the log viewer.

9. Add supervisor hosts to `network.negotiate-auth.trusted-uris` (similar to the steps needed to access the Storm UI).