HDP Security Reference 3

Security Reference

Date of Publish: 2018-07-15

http://docs.hortonworks.com

Contents

Non-Ambari Security Overview

This chapter is a collection of security content for users who choose to have a non-Ambari environment.

It is recommended that all clusters use Ambari to plan, install, configure, maintain, and manage your HDP environment. This chapter is a collection of security content for users who choose to have a non-Ambari environment. Ambari streamlines many processes behind the scenes and runs checks on configurations for validity. Non-Ambari content is, therefore, more complicated.

Note: Using the for-Ambari content (as opposed to this non-Ambari content) is recommended. This non-Ambari content is provided only as a courtesy.

Related Information

Setting Up Kerberos Authentication for Non-Ambari Clusters

Configure Non-Ambari Ranger SSL

Enable Audit Logging in Non-Ambari Clusters

Setting Up Kerberos Authentication for Non-Ambari Clusters

This section provides information for enabling security for a manually installed version of HDP.

Preparing Kerberos

This subsection provides information on setting up Kerberos for an HDP installation.

Non-Ambari Kerberos Overview

To create secure communication among its various components, HDP uses Kerberos. Kerberos is a third-party authentication mechanism, in which users and services that users wish to access rely on the Kerberos server to authenticate each to the other. This mechanism also supports encrypting all traffic between the user and the service.

The Kerberos server itself is known as the Key Distribution Center, or KDC. At a high level, it has three parts:

- A database of users and services (known as principals) and their respective Kerberos passwords
- An authentication server (AS) which performs the initial authentication and issues a Ticket Granting Ticket (TGT)
- A Ticket Granting Server (TGS) that issues subsequent service tickets based on the initial TGT.

A user principal requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal's Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS.

Because a service principal cannot provide a password each time to decrypt the TGT, it uses a special file, called a keytab, which contains its authentication credentials.

The service tickets allow the principal to access various services. The set of hosts, users, and services over which the Kerberos server has control is called a realm.

Install and Configure the KDC (Non-Ambari)

To use Kerberos with HDP, either use an existing KDC or install a new one for HDP only. This section gives a very high level description of the installation process when setting up Kerberos for non-Ambari clusters.

Procedure

1. Install the KDC server:

OS Flavor	Command
RHEL, CentOS, or Oracle Linux	yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
SLES	zypper install krb5 krb5-server krb5-client
Ubuntu or Debian	apt-get install krb5 krb5-server krb5-client

Note:

The host on which you install the KDC must itself be secure.

2. When the server is installed you must edit the two main configuration files. Update the KDC configuration by replacing EXAMPLE.COM with your domain and kerberos.example.com with the FQDN of the KDC host. Configuration files are in the following locations:

OS Flavor	Configuration File Location
RHEL, CentOS, or Oracle Linux	/etc/krb5.conf /var/kerberos/krb5kdc/kdc.conf
SLES	/etc/krb5.conf /var/lib/kerberos/krb5kdc/kdc.conf
Ubuntu or Debian	/etc/krb5.conf /var/kerberos/krb5kdc/kdc.conf

3. Copy the updated krb5.conf to every cluster node.

Related Information

RHEL documentation

SLES documentation

Ubuntu and Debian documentation

Create the Database and Set Up the First Administrator

How to create a database and configure the admin when setting up Kerberos for non-Ambari clusters..

Procedure

1. Use the utility kdb5_util to create the Kerberos database:

OS	Run
RHEL, CentOS, or Oracle Linux	/usr/sbin/kdb5_util create -s
SLES	kdb5_util create -s
Ubuntu or Debian	kdb5_util -s create

Note:

The -s option stores the master server key for the database in a stash file. If the stash file is not present, you must log into the KDC with the master password (specified during installation) each time it starts. This will automatically regenerate the master server key.

2. Set up the KDC Access Control List (ACL):

OS	Action
RHEL, CentOS, or Oracle Linux	Add administrators to /var/kerberos/ krb5kdc/ kadm5.acl

OS	Action
SLES	Add administrators to /var/lib/kerberos/krb5kdc/kadm5.acl.

Note:

For example, the following line grants full access to the database for users with the admin extension: */admin@EXAMPLE.COM *

3. Start kadmin for the change to take effect.
4. Create the first user principal. This must be done at a terminal window on the KDC machine itself, while you are logged in as root. Notice the .local. Normal kadmin usage requires that a principal with appropriate access already exist. The kadmin.local command can be used even if no principals exist:

```
/usr/sbin/kadmin.local -q "addprinc $username/admin
```

Now this user can create additional principals either on the KDC machine or through the network. The following instruction assumes that you are using the KDC machine.

5. On the KDC, start Kerberos:

OS	Run
RHEL, CentOS, or Oracle Linux	`/sbin/service krb5kdc start` `/sbin/service kadmin start`
SLES	`rckrb5kdc start` `rckadmind start`
Ubuntu or Debian	`/etc/init.d/krb5-kdc start` `/etc/init.d/kadmin start`

Create Service Principals and Keytab Files for HDP (Non-Ambari)

How to create service principals and keytab files when setting up Kerberos for non-Ambari clusters.

About this task

Each service in HDP must have its own principal. Because services do not login with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal.

First create the principal, using mandatory naming conventions. Then create the keytab file with that principal's information, and copy the file to the keytab directory on the appropriate service host.

Procedure

1. To create a service principal you will use the kadmin utility. This is a command-line driven utility into which you enter Kerberos commands to manipulate the central database. To start kadmin, enter:

```
'kadmin $USER/admin@REALM'
```

2. To create a service principal, enter the following:

```
kadmin: addprinc -randkey $principal_name/$service-host-FQDN@$hadoop.realm
```

You must have a principal with administrative permissions to use this command. The randkey is used to generate the password.

The $principal_name part of the name must match the values in the following table.

In the example each service principal's name has appended to it the fully qualified domain name of the host on which it is running. This is to provide a unique principal name for services that run on multiple hosts, like DataNodes and TaskTrackers. The addition of the hostname serves to distinguish, for example, a request from DataNode A from a request from DataNode B.

This is important for two reasons:

- If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised
- If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same timestamp, then the authentication would be rejected as a replay request.

Note: The NameNode, Secondary NameNode, and Oozie require two principals each.

If you are configuring High Availability (HA) for a Quorom-based NameNode, you must also generate a principle (jn/$FQDN) and keytab (jn.service.keytab) for each JournalNode. JournalNode also requires the keytab for its HTTP service. If the JournalNode is deployed on the same host as a NameNode, the same keytab file (spnego.service.keytab) can be used for both. In addition, HA requires two NameNodes. Both the active and standby NameNodes require their own principle and keytab files. The service principles of the two NameNodes can share the same name, specified with the dfs.namenode.kerberos.principal property in hdfs-site.xml, but the NameNodes still have different fully qualified domain names.

Table 1: Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	nn/$FQDN
HDFS	NameNode HTTP	HTTP/$FQDN
HDFS	SecondaryNameNode	nn/$FQDN
HDFS	SecondaryNameNode HTTP	HTTP/$FQDN
HDFS	DataNode	dn/$FQDN
MR2	History Server	jhs/$FQDN
MR2	History Server HTTP	HTTP/$FQDN
YARN	ResourceManager	rm/$FQDN
YARN	NodeManager	nm/$FQDN
Oozie	Oozie Server	oozie/$FQDN
Oozie	Oozie HTTP	HTTP/$FQDN
Hive	Hive Metastore HiveServer2	hive/$FQDN
Hive	WebHCat	HTTP/$FQDN
HBase	MasterServer	hbase/$FQDN
HBase	RegionServer	hbase/$FQDN
Storm	Nimbus server DRPC daemon	nimbus/$FQDN **

Service	Component	Mandatory Principal Name
Storm	Storm UI daemon Storm Logviewer daemon Nodes running process controller (such as Supervisor)	storm/$FQDN **
Kafka	KafkaServer	kafka/$FQDN
Zeppelin	Zeppelin Server	zeppelin/$FQDN
ZooKeeper	ZooKeeper	zookeeper/$FQDN
JournalNode Server*	JournalNode	jn/$FQDN
Gateway	Knox	knox/$FQDN

* Only required if you are setting up NameNode HA.

** For more information, see “Configure Kerberos Authentication for Storm”.

For example: To create the principal for a DataNode service, issue this command:

```
kadmin: addprinc -randkey dn/$datanode-host@$hadoop.realm
```

3. Extract the related keytab file and place it in the keytab directory of the appropriate respective components. The default directory is /etc/krb5.keytab.

```
kadmin: xst -k $keytab_file_name $principal_name/
fully.qualified.domain.name
```

You must use the mandatory names for the $keytab_file_name variable shown in the following table.

Table 2: Service Keytab File Names

Component	Principal Name	Mandatory Keytab File Name
NameNode	nn/$FQDN	nn.service.keytab
NameNode HTTP	HTTP/$FQDN	spnego.service.keytab
SecondaryNameNode	nn/$FQDN	nn.service.keytab
SecondaryNameNode HTTP	HTTP/$FQDN	spnego.service.keytab
DataNode	dn/$FQDN	dn.service.keytab
MR2 History Server	jhs/$FQDN	nm.service.keytab
MR2 History Server HTTP	HTTP/$FQDN	spnego.service.keytab
YARN	rm/$FQDN	rm.service.keytab
YARN	nm/$FQDN	nm.service.keytab
Oozie Server	oozie/$FQDN	oozie.service.keytab
Oozie HTTP	HTTP/$FQDN	spnego.service.keytab
Hive Metastore HiveServer2	hive/$FQDN	hive.service.keytab
WebHCat	HTTP/$FQDN	spnego.service.keytab

Component	Principal Name	Mandatory Keytab File Name
HBase Master Server	hbase/$FQDN	hbase.service.keytab
HBase RegionServer	hbase/$FQDN	hbase.service.keytab
Storm	storm/$FQDN	storm.service.keytab
Kafka	kafka/$FQDN	kafka.service.keytab
Zeppelin Server	zeppelin/$FQDN	zeppelin.server.kerberos.keytab
ZooKeeper	zookeeper/$FQDN	zk.service.keytab
Journal Server*	jn/$FQDN	jn.service.keytab
Knox Gateway**	knox/$FQDN	knox.service.keytab

* Only required if you are setting up NameNode HA.

** Only required if you are using a Knox Gateway.

For example: To create the keytab files for the NameNode, issue these commands:

```
kadmin: xst -k nn.service.keytab nn/$namenode-host kadmin: xst -k
 spnego.service.keytab HTTP/$namenode-host
```

When you have created the keytab files, copy them to the keytab directory of the respective service hosts.

4. Verify that the correct keytab files and principals are associated with the correct service using the klist command. For example, on the NameNode:

```
klist –k -t /etc/security/nn.service.keytab
```

Do this on each respective service in your cluster.

Configuring HDP for Kerberos

How to configure HDP when setting up Kerberos for non-Ambari clusters.

Configuring HDP for Kerberos has two parts:

- Creating a mapping between service principals and UNIX usernames.

  Hadoop uses group memberships of users at various places, such as to determine group ownership for files or for access control.

  A user is mapped to the groups it belongs to using an implementation of the GroupMappingServiceProvider interface. The implementation is pluggable and is configured in core-site.xml.

  By default Hadoop uses ShellBasedUnixGroupsMapping, which is an implementation of GroupMappingServiceProvider. It fetches the group membership for a username by executing a UNIX shell command. In secure clusters, since the usernames are actually Kerberos principals, ShellBasedUnixGroupsMapping will work only if the Kerberos principals map to valid UNIX usernames. Hadoop provides a feature that lets administrators specify mapping rules to map a Kerberos principal to a local UNIX username.
- Adding information to three main service configuration files.

  There are several optional entries in the three main service configuration files that must be added to enable security on HDP.

This section provides information on configuring HDP for Kerberos.

Create Mappings Between Principals and UNIX Usernames

HDP uses a rule-based system to create mappings between service principals and their related UNIX usernames. The rules are specified in the core-site.xml configuration file as the value to the optional key hadoop.security.auth_to_local.

About this task

The default rule is simply named DEFAULT. It translates all principals in your default domain to their first component. For example, myusername@APACHE.ORG and myusername/admin@APACHE.ORG both become myusername, assuming your default domain is APACHE.ORG.

While mapping the Kerberos principals, if the Kerberos principal names are in the UPPERCASE or CaMeLcase, the names will not be recognized on the Linux machine (as Linux users are always in lower case). You must add the extra switch "/L" in the rule definition to force the conversion to lower case.

Creating Rules

To accommodate more complex translations, you can create a hierarchical set of rules to add to the default. Each rule is divided into three parts: base, filter, and substitution.

Procedure

- The Base:

  The base begins with the number of components in the principal name (excluding the realm), followed by a colon, and the pattern for building the username from the sections of the principal name. In the pattern section $0 translates to the realm, $1 translates to the first component, and $2 to the second component.

  ```
  [1:$1@$0] translates myusername@APACHE.ORG to myusername@APACHE.ORG
  [2:$1] translates myusername/admin@APACHE.ORG to myusername
  [2:$1%$2] translates myusername/admin@APACHE.ORG to “myusername%admin"
  ```

- The Filter:

  The filter consists of a regular expression (regex) in a parentheses. It must match the generated string for the rule to apply.

  ```
  (.*%admin) matches any string that ends in %admin
  (.*@SOME.DOMAIN) matches any string that ends in @SOME.DOMAIN
  ```

- The Substitution:

  The substitution is a sed rule that translates a regex into a fixed string.

  ```
  s/@ACME\.COM// removes the first instance of @ACME.DOMAIN
  s/@[A-Z]*\.COM// remove the first instance of @ followed by a name
   followed by COM.
  s/X/Y/g replace all of X's in the name with Y
  ```

Example

If your default realm was APACHE.ORG, but you also wanted to take all principals from ACME.COM that had a single component joe@ACME.COM, the following rule would do this:

Example

To translate names with a second component, you could use these rules:

```
RULE:[1:$1@$0](.@ACME.COM)s/@.//
RULE:[2:$1@$0](.@ACME.COM)s/@.// DEFAULT
```

Example

To treat all principals from APACHE.ORG with the extension /admin as admin, your rules would look like this:

```
RULE[2:$1%$2@$0](.%admin@APACHE.ORG)s/./admin/
DEFAULT
```

Example

To force username conversion from CaMeLcase or UPPERCASE to lowercase, you could model the following auth_to_local rule examples which have the lowercase switch added:

```
RULE:[1:$1]/L
RULE:[2:$1]/L
RULE:[2:$1;$2](^.*;admin$)s/;admin$///L
RULE:[2:$1;$2](^.*;guest$)s/;guest$//g/L
```

```
RULE:[1:$1]/L
RULE:[2:$1]/L
RULE:[2:$1;$2](^.*;admin$)s/;admin$///L
RULE:[2:$1;$2](^.*;guest$)s/;guest$//g/L
```

And based on these rules, here are the expected output for the following inputs:

"JOE@FOO.COM" to "joe" "Joe/root@FOO.COM" to "joe" "Joe/admin@FOO.COM" to "joe" "Joe/guestguest@FOO.COM" to "joe"

Adding Security Information to Configuration Files

Reference material of how to add security information to configuration files when setting up Kerberos for non-Ambari clusters.

To enable security on HDP, you must add optional information to various configuration files.

Before you begin, set JSVC_Home in hadoop-env.sh.

- For RHEL/CentOS/Oracle Linux:

  ```
  export JSVC_HOME=/usr/libexec/bigtop-utils
  ```

- For SLES and Ubuntu:

  ```
  export JSVC_HOME=/usr/hdp/current/bigtop-utils
  ```

core-site.xml

Reference material for adding security information to the core-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

Add the following information to the core-site.xml file on every host in your cluster:

Table 3: General core-site.xml and Knox

Property Name	Property Value	Description
hadoop.security.authentication	kerberos	Set the authentication type for the cluster. Valid values are: simple or kerberos.

Property Name	Property Value	Description
hadoop.rpc.protection	authentication; integrity; privacy	This is an [OPTIONAL] setting. If not set, defaults to authentication. authentication = authentication only; the client and server mutually authenticate during connection setup. integrity = authentication and integrity; guarantees the integrity of data exchanged between client and server as well as authentication. privacy = authentication, integrity, and confidentiality; guarantees that data exchanged between client and server is encrypted and is not readable by a “man in the middle”.
hadoop.security.authorization	true	Enable authorization for different protocols.
hadoop.security.auth_to_local	The mapping rules. For example: RULE:[2:$1@$0] ([jt]t@.*EXAMPLE.COM)s/.*/ mapred/ RULE:[2:$1@$0] ([nd]n@.*EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.*/ hbase/ RULE:[2:$1@$0] (rs@.*EXAMPLE.COM)s/.*/hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. “Create Mappings Between Principals and UNIX Usernames” for more information.

Following is the XML for these entries:

```
<property>
     <name>hadoop.security.authentication</name>
     <value>kerberos</value>
     <description> Set the authentication for the cluster.
     Valid values are: simple or kerberos.</description>
</property>

<property>
     <name>hadoop.security.authorization</name>
     <value>true</value>
     <description>Enable authorization for different protocols.</
description>
</property>

<property>
    <name>hadoop.security.auth_to_local</name>
    <value>
    RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/mapred/
    RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/hdfs/
    RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.*/hbase/
    RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/.*/hbase/
    DEFAULT
    </value>
    <description>The mapping from kerberos principal names
    to local OS user names.</description>
</property>
```

When using the Knox Gateway, add the following to the core-site.xml file on the master nodes host in your cluster:

Table 4: core-site.xml Master Node Settings -- Knox Gateway

Property Name	Property Value	Description
hadoop.proxyuser.knox.groups	users	Grants proxy privileges for Knox user.
hadoop.proxyuser.knox.hosts	$knox_host_FQDN	Identifies the Knox Gateway host.

Following is the XML for Knox settings:

```
<property>
     <name>hadoop.security.authentication</name>
     <value>kerberos</value>
     <description>Set the authentication for the cluster.
     Valid values are: simple or kerberos.</description>
</property>

<property>
     <name>hadoop.security.authorization</name>
     <value>true</value>
     <description>Enable authorization for different protocols.
     </description>
</property>

<property>
     <name>hadoop.security.auth_to_local</name>
     <value>
     RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/mapred/
     RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/hdfs/
     RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.*/hbase/
     RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/.*/hbase/
     DEFAULT
     </value>
     <description>The mapping from kerberos principal names
     to local OS user names.</description>
</property>

<property>
     <name>hadoop.proxyuser.knox.groups</name>
     <value>users</value>
</property>
```

Related Information

Create Mappings Between Principals and UNIX Usernames

hdfs-site.xml

Reference material for adding security information to the hdfs-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

To the hdfs-site.xml file on every host in your cluster, you must add the following information:

Table 5: hdfs-site.xml File Property Settings

Property Name	Property Value	Description
dfs.permissions.enabled	true	If true, permission checking in HDFS is enabled. If false, permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.

Property Name	Property Value	Description
dfs.permissions.supergroup	hdfs	The name of the group of super-users.
dfs.block.access.token.enable	true	If true, access tokens are used as capabilities for accessing DataNodes. If false, no access tokens are checked on accessing DataNodes.
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the NameNode.
dfs.secondary.namenode.kerberos. principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the secondary NameNode.
dfs.web.authentication.kerberos. principal	HTTP/_HOST@EXAMPLE.COM	The HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint. The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP SPNEGO specification.
dfs.web.authentication.kerberos. keytab	/etc/security/keytabs/spnego.service.keytab	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM	The Kerberos principal that the DataNode runs as. "_HOST" is replaced by the real host name.
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals.
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals. <question?>
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab	The filename of the keytab file for the DataNode.
dfs.https.port	50470	The HTTPS port to which the NameNode binds.
dfs.namenode.https-address	Example: ip-10-111-59-170.ec2.internal:50470	The HTTPS address to which the NameNode binds.
dfs.datanode.data.dir.perm	750	The permissions that must be set on the dfs.data.dir directories. The DataNode will not come up if all existing dfs.data.dir directories do not have this setting. If the directories do not exist, they will be created with this permission.
dfs.cluster.administrators	hdfs	ACL for who all can view the default servlets in the HDFS.
dfs.namenode.kerberos.internal. spnego.principal	${dfs.web.authentication.kerberos.principal}	
dfs.secondary.namenode.kerberos. internal.spnego.principal	${dfs.web.authentication.kerberos.principal}	

Following is the XML for these entries:

```
<property>
     <name>dfs.permissions</name>
     <value>true</value>
     <description> If "true", enable permission checking in
```

```
     HDFS. If "false", permission checking is turned
     off, but all other behavior is
     unchanged. Switching from one parameter value to the other does
     not change the mode, owner or group of files or
     directories. </description>
</property>

<property>
     <name>dfs.permissions.supergroup</name>
     <value>hdfs</value>
     <description>The name of the group of
     super-users.</description>
</property>

<property>
     <name>dfs.namenode.handler.count</name>
     <value>100</value>
     <description>Added to grow Queue size so that more
     client connections are allowed</description>
</property>

<property>
     <name>ipc.server.max.response.size</name>
     <value>5242880</value>
</property>

<property>
     <name>dfs.block.access.token.enable</name>
     <value>true</value>
     <description> If "true", access tokens are used as capabilities
     for accessing datanodes. If "false", no access tokens are checked on
     accessing datanodes. </description>
</property>

<property>
     <name>dfs.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description> Kerberos principal name for the
     NameNode </description>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description>Kerberos principal name for the secondary NameNode.
     </description>
</property>

<property>
     <!--cluster variant -->
     <name>dfs.secondary.http.address</name>
     <value>ip-10-72-235-178.ec2.internal:50090</value>
     <description>Address of secondary namenode web server</description>
</property>

<property>
     <name>dfs.secondary.https.port</name>
     <value>50490</value>
     <description>The https port where secondary-namenode
     binds</description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.principal</name>
```

```
     <value>HTTP/_HOST@EXAMPLE.COM</value>
     <description> The HTTP Kerberos principal used by Hadoop-Auth in the
 HTTP endpoint.
     The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP
     SPNEGO specification.
     </description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.keytab</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
     <description>The Kerberos keytab file with the credentials for the HTTP
     Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
     </description>
</property>

<property>
     <name>dfs.datanode.kerberos.principal</name>
     <value>dn/_HOST@EXAMPLE.COM</value>
     <description>
     The Kerberos principal that the DataNode runs as. "_HOST" is replaced
 by the real
     host name.
     </description>
</property>

<property>
     <name>dfs.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.secondary.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.datanode.keytab.file</name>
     <value>/etc/security/keytabs/dn.service.keytab</value>
     <description>
     The filename of the keytab file for the DataNode.
     </description>
</property>

<property>
     <name>dfs.https.port</name>
     <value>50470</value>
     <description>The https port where namenode
     binds</description>
</property>

<property>
     <name>dfs.https.address</name>
     <value>ip-10-111-59-170.ec2.internal:50470</value>
     <description>The https address where namenode binds</description>
</property>
```

```
<property>
     <name>dfs.datanode.data.dir.perm</name>
     <value>750</value>
     <description>The permissions that should be there on
     dfs.data.dir directories. The datanode will not come up if the
     permissions are different on existing dfs.data.dir directories. If
     the directories don't exist, they will be created with this
     permission.</description>
</property>

<property>
     <name>dfs.access.time.precision</name>
     <value>0</value>
     <description>The access time for HDFS file is precise upto this
     value.The default value is 1 hour. Setting a value of 0
     disables access times for HDFS.
     </description>
</property>

<property>
     <name>dfs.cluster.administrators</name>
     <value> hdfs</value>
     <description>ACL for who all can view the default
     servlets in the HDFS</description>
</property>

<property>
     <name>ipc.server.read.threadpool.size</name>
     <value>5</value>
     <description></description>
</property>

<property>
     <name>dfs.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

In addition, you must set the user on all secure DataNodes:

```
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/grid/0/var/run/hadoop/
$HADOOP_SECURE_DN_USER
```

yarn-site.xml

Reference material for adding security information to the yarn-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

You must add the following information to the yarn-site.xml file on every host in your cluster:

Table 6: yarn-site.xml Property Settings

Property	Value	Description
yarn.resourcemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the ResourceManager.

Property	Value	Description
yarn.resourcemanager.keytab	/etc/krb5.keytab	The keytab for the ResourceManager.
yarn.nodemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the NodeManager.
yarn.nodemanager.keytab	/etc/krb5.keytab	The keytab for the NodeManager.
yarn.nodemanager.container-executor.class	org.apache.hadoop.yarn.server. nodemanager.LinuxContainer Executor	The class that will execute (launch) the containers.
yarn.nodemanager.linux-container-executor.path	hadoop-3.0.0-SNAPSHOT/bin/container-executor	The path to the Linux container executor.
yarn.nodemanager.linux-container-executor.group	hadoop	A special group (e.g., hadoop) with executable permissions for the container executor, of which the NodeManager UNIX user is the group member and no ordinary application user is. If any application user belongs to this special group, security will be compromised. This special group name should be specified for the configuration property.
yarn.timeline-service.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the Timeline Server.
yarn.timeline-service.keytab	/etc/krb5.keytab	The Kerberos keytab for the Timeline Server.
yarn.resourcemanager.webapp. delegation-token-auth-filter.enabled	true	Flag to enable override of the default Kerberos authentication filter with the RM authentication filter to allow authentication using delegation tokens (fallback to Kerberos if the tokens are missing). Only applicable when the http authentication type is Kerberos.
yarn.timeline-service.http-authentication.type	kerberos	Defines authentication used for the Timeline Server HTTP endpoint. Supported values are: simple | kerberos | $AUTHENTICATION_HANDLER _CLASSNAME
yarn.timeline-service.http-authentication.kerberos.principal	HTTP/localhost@EXAMPLE.COM	The Kerberos principal to be used for the Timeline Server HTTP endpoint.
yarn.timeline-service.http-authentication.kerberos.keytab	authentication.kerberos.keytab /etc/ krb5.keytab	The Kerberos keytab to be used for the Timeline Server HTTP endpoint.

Following is the XML for these entries:

```
<property>
     <name>yarn.resourcemanager.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.resourcemanager.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.nodemanager.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.nodemanager.keytab</name>
```

```
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.nodemanager.container-executor.class</name>

 <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</
value>
</property>

<property>
     <name>yarn.nodemanager.linux-container-executor.path</name>
     <value>hadoop-3.0.0-SNAPSHOT/bin/container-executor</value>
</property>

<property>
     <name>yarn.nodemanager.linux-container-executor.group</name>
     <value>hadoop</value>
</property>

<property>
     <name>yarn.timeline-service.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.timeline-service.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.resourcemanager.webapp.delegation-token-auth-
filter.enabled</name>
     <value>true</value>
</property>

<property>
     <name>yarn.timeline-service.http-authentication.type</name>
     <value>kerberos</value>
</property>

<property>
     <name>yarn.timeline-service.http-authentication.kerberos.principal</
name>
     <value>HTTP/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.timeline-service.http-authentication.kerberos.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>
```

mapred-site.xml

Reference material for adding security information to the mapred-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

You must add the following information to the mapred-site.xml file on every host in your cluster:

Table 7: mapred-site.xml Property Settings

Property Name	Property Value	Description
mapreduce.jobhistory.keytab	/etc/security/keytabs/jhs.service.keytab	Kerberos keytab file for the MapReduce JobHistory Server.
mapreduce.jobhistory.principal	jhs/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the MapReduce JobHistory Server.
mapreduce.jobhistory.webapp.address	TODO-JOBHISTORYNODE-HOSTNAME:19888	MapReduce JobHistory Server Web UI host:port
mapreduce.jobhistory.webapp.https. address	TODO-JOBHISTORYNODE-HOSTNAME:19889	MapReduce JobHistory Server HTTPS Web UI host:port
mapreduce.jobhistory.webapp.spnego-keytab-file	/etc/security/keytabs/spnego.service.keytab	Kerberos keytab file for the spnego service.
mapreduce.jobhistory.webapp.spnego-principal	HTTP/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the spnego service.

Following is the XML for these entries:

```
<property>
     <name>mapreduce.jobhistory.keytab</name>
     <value>/etc/security/keytabs/jhs.service.keytab</value>
</property>

<property>
     <name>mapreduce.jobhistory.principal</name>
     <value>jhs/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.address</name>
     <value>TODO-JOBHISTORYNODE-HOSTNAME:19888</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.https.address</name>
     <value>TODO-JOBHISTORYNODE-HOSTNAME:19889</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.spnego-keytab-file</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.spnego-principal</name>
     <value>HTTP/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>
```

hbase-site.xml

Reference material for adding security information to the hbase-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

For HBase to run on a secured cluster, HBase must be able to authenticate itself to HDFS. Add the following information to the hbase-site.xml file on your HBase server. However, include the phoenix.queryserver.kerberos.principal and phoenix.queryserver.kerberos.keytab property entries only if you will be configuring Kerberos authentication for a Phoenix Query Server.

Note:

There are no default values for the property settings. The entries in the "Sample Setting" column are only examples.

Table 8: hbase-site.xml Property Settings for HBase Server and Phoenix Query Server

Property Name	Sample Setting	Description
hbase.master.keytab.file	/etc/security/keytabs/hbase.service.keytab	The keytab for the HMaster service principal.
hbase.master.kerberos.principal	hbase/_HOST@EXAMPLE.COM	The Kerberos principal name that should be used to run the HMaster process. If _HOST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
hbase.regionserver.keytab.file	/etc/security/keytabs/hbase.service.keytab	The keytab for the HRegionServer service principal.
hbase.regionserver.kerberos.principal	hbase/_HOST@EXAMPLE.COM	The Kerberos principal name that should be used to run the HRegionServer process. If _HOST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
hbase.superuser	hbase	A comma-separated list of users or groups that are allowed full privileges, regardless of stored ACLs, across the cluster. Only used when HBase security is enabled.
hbase.coprocessor.region.classes	Setting 1:org.apache.hadoop.hbase. security.token.TokenProvider, Setting 2:org.apache.hadoop.hbase. security.access.SecureBulkLoadEndpoint, Setting 3:org.apache.hadoop.hbase. security.access.AccessController	A comma-separated list of coprocessors that are loaded by default on all tables. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own coprocessor, add the class to HBase's classpath and add the fully qualified class name here. Coprocessors can also be loaded programmatically using HTableDescriptor.
hbase.coprocessor.master.classes	org.apache.hadoop.hbase.security. access.AccessController	A comma-separated list of MasterObserver coprocessors that are loaded by the active HMaster process. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own MasterObserver, add the class to HBase's classpath and add the fully qualified class name here.
hbase.coprocessor.regionserver.classes	org.apache.hadoop.hbase.security. access.AccessController	A comma-separated list of RegionServerObserver coprocessors that are loaded by the HRegionServer processes. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own RegionServerObserver, add the class to the HBase classpath and fully qualified class name here.
phoenix.queryserver.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	The Kerberos principal for the Phoenix Query Server process. The Phoenix Query Server is an optional component; this property only needs to be set when the query server is installed.

Property Name	Sample Setting	Description
phoenix.queryserver.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	The path to the Kerberos keytab file for the Phoenix Query Server process. The Phoenix Query Server is an optional component; this property only needs to be set when the query server is installed.

Tip:

Phoenix Query Server users: See “Configuring Phoenix Query Server” for the required setting in the core-site.xml file to complete Kerberos setup of the query server.

The following lists the XML for the hbase-site.xml file entries:

```
<property>
     <name>hbase.master.keytab.file</name>
     <value>/etc/security/keytabs/hbase.service.keytab</value>
     <description>Full path to the Kerberos keytab file to use for logging
     in the configured HMaster server principal.
     </description>
</property>

<property>
     <name>hbase.master.kerberos.principal</name>
     <value>hm/_HOST@EXAMPLE.COM</value>
      <description>Ex. "hbase/_HOST@EXAMPLE.COM".
     The Kerberos principal name that should be used to run the HMaster
     process. The principal name should be in the form: user/
hostname@DOMAIN.
     If "_HOST" is used as the hostname portion, it will be replaced with
     the actual hostname of the running instance.
     </description>
</property>

<property>
     <name>hbase.regionserver.keytab.file</name>
     <value>/etc/security/keytabs/hbase.service.keytab</value>
     <description>Full path to the kerberos keytab file to use for logging
     in the configured HRegionServer server principal.
     </description>
</property>

<property>
     <name>hbase.regionserver.kerberos.principal</name>
     <value>hbase/_HOST@EXAMPLE.COM</value>
     <description>Ex. "hbase/_HOST@EXAMPLE.COM".
     The kerberos principal name that
     should be used to run the HRegionServer process. The
     principal name should be in the form:
     user/hostname@DOMAIN. If _HOST
     is used as the hostname portion, it will be replaced
     with the actual hostname of the running
     instance. An entry for this principal must exist
     in the file specified in hbase.regionserver.keytab.file
     </description>
</property>

<!--Additional configuration specific to HBase security -->

<property>
     <name>hbase.superuser</name>
     <value>hbase</value>
     <description>List of users or groups (comma-separated), who are
```

```
     allowed full privileges, regardless of stored ACLs, across the cluster.
     Only used when HBase security is enabled.
     </description>
</property>

<property>
     <name>hbase.coprocessor.region.classes</name>
     <value>org.apache.hadoop.hbase.security.token.TokenProvider,
     org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,
     org.apache.hadoop.hbase.security.access.AccessController</value>
     <description>A comma-separated list of coprocessors that are loaded
     by default on all tables. For any override coprocessor method,
     these classes will be called in order. After implementing your
     own coprocessor, just put it in HBase's classpath and add the
     fully qualified class name here. A coprocessor can also be loaded on
     demand by setting HTableDescriptor.
     </description>
</property>

<property>
     <name>hbase.coprocessor.master.classes</name>
     <value>org.apache.hadoop.hbase.security.access.AccessController</
value>
     <description>A comma-separated list of MasterObserver coprocessors that
     are loaded by the active HMaster process. For any implemented
 coprocessor
     methods, the listed classes will be called in order. After implementing
 your
     own MasterObserver, add the class to HBase's classpath and add the
 fully
     qualified class name here.
     </description>
</property>

<property>
    <name>hbase.coprocessor.regionserver.classes</name>
    <value>org.apache.hadoop.hbase.security.access.AccessController</value>
    <description>A comma-separated list of RegionServerObserver coprocessors
    that are loaded by the HRegionServer processes. For any implemented
    coprocessor methods, the listed classes will be called in order. After
    implementing your own RegionServerObserver, add the class to the HBase
    classpath and fully qualified class name here.
    </description>
</property>
<property>
    <name>phoenix.queryserver.kerberos.principal</name>
    <value>HTTP/_HOST@EXAMPLE.COM</value>
    <description>The Kerberos principal for the Phoenix Query Server
    process. The Phoenix Query Server is an optional component; this
    property only needs to be set when the query server is installed.
    </description>
</property>

<property>
    <name>phoenix.queryserver.kerberos.keytab</name>
    <value>/etc/security/keytabs/spnego.service.keytab</value>
    <description>The path to the Kerberos keytab file for the
    Phoenix Query Server process. The Phoenix Query Server is an optional
    component; this property only needs to be set when the query server
    is installed.</description>
</property>
```

Related Information

Configure Phoenix Query Server

hive-site.xml

Reference material for adding security information to the hive-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

HiveServer2 supports Kerberos authentication for all clients.

Add the following information to the hive-site.xml file on every host in your cluster:

Table 9: hive-site.xml Property Settings

Property Name	Property Value	Description
hive.metastore.sasl.enabled	true	If true, the Metastore Thrift interface will be secured with SASL and clients must authenticate with Kerberos.
hive.metastore.kerberos.keytab.file	/etc/security/keytabs/hive.service.keytab	The keytab for the Metastore Thrift service principal.
hive.metastore.kerberos.principal	hive/_HOST@EXAMPLE.COM	The service principal for the Metastore Thrift server. If _HOST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.

Following is the XML for these entries:

```
<property>
     <name>hive.metastore.sasl.enabled</name>
     <value>true</value>
     <description>If true, the metastore thrift interface will be secured
 with SASL.
     Clients must authenticate with Kerberos.</description>
</property>

<property>
     <name>hive.metastore.kerberos.keytab.file</name>
     <value>/etc/security/keytabs/hive.service.keytab</value>
     <description>The path to the Kerberos Keytab file containing the
     metastore thrift server's service principal.
     </description>
</property>

<property>
     <name>hive.metastore.kerberos.principal</name>
     <value>hive/_HOST@EXAMPLE.COM</value>
     <description>The service principal for the metastore thrift server. The
     special string _HOST will be replaced automatically with the correct
     hostname.</description>
</property>
```

oozie-site.xml

Reference material for adding security information to the oozie-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

To the oozie-site.xml file, add the following information:

Table 10: oozie-site.xml Property Settings

Property Name	Property Value	Description
oozie.service.AuthorizationService. security.enabled	true	Specifies whether security (user name/admin role) is enabled or not. If it is disabled any user can manage the Oozie system and manage any job.
oozie.service.HadoopAccessorService. kerberos.enabled	true	Indicates if Oozie is configured to use Kerberos.
local.realm	EXAMPLE.COM	Kerberos Realm used by Oozie and Hadoop. Using local.realm to be aligned with Hadoop configuration.
oozie.service.HadoopAccessorService. keytab.file	/etc/security/keytabs/oozie.service.keytab	The keytab for the Oozie service principal.
oozie.service.HadoopAccessorService. kerberos.principaloozie/ _HOSTl@EXAMPLE.COM	oozie/_HOSTl@EXAMPLE.COM	Kerberos principal for Oozie service.
oozie.authentication.type	kerberos	
oozie.authentication.kerberos. principal	HTTP/_HOST@EXAMPLE.COM	Whitelisted job tracker for Oozie service.
oozie.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	Location of the Oozie user keytab file.
oozie.service.HadoopAccessorService. nameNode.whitelist		
oozie.authentication.kerberos. name.rules	RULE:[2:$1@$0] ([jt]t@.*EXAMPLE.COM)s/.*/ mapred/ RULE:[2:$1@$0] ([nd]n@.*EXAMPLE.COM)s/.*/hdfs/ RULE: [2:$1@$0](hm@.*EXAMPLE.COM)s/.*/ hbase/ RULE:[2:$1@$0] (rs@.*EXAMPLE.COM)s/.*/hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. “Create Mappings Between Principals and UNIX Usernames” for more information.
oozie.service.ProxyUserService. proxyuser.knox.groups	users	Grant proxy privileges to the Knox user. Note only required when using a Knox Gateway.
oozie.service.ProxyUserService. proxyuser.knox.hosts	$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

Related Information

Create Mappings Between Principals and UNIX Usernames

webhcat-site.xml

Reference material for adding security information to the webhcat-site.xml configuration file when setting up Kerberos for non-Ambari clusters.

To the webhcat-site.xml file, add the following information:

Table 11: webhcat-site.xml Property Settings

Property Name	Property Value	Description
templeton.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	
templeton.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	

Property Name	Property Value	Description
templeton.kerberos.secret	secret	
hadoop.proxyuser.knox.groups	users	Grant proxy privileges to the Knox user. Note only required when using a Knox Gateway.
hadoop.proxyuser.knox.hosts	$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

limits.conf

Reference material for adding security information to the limits.conf configuration file when setting up Kerberos for non-Ambari clusters.

Adjust the Maximum Number of Open Files and Processes

In a secure cluster, if the DataNodes are started as the root user, JSVC downgrades the processing using setuid to hdfs. However, the ulimit is based on the ulimit of the root user, and the default ulimit values assigned to the root user for the maximum number of open files and processes may be too low for a secure cluster. This can result in a “Too Many Open Files” exception when the DataNodes are started.

Therefore, when configuring a secure cluster you should increase the following root ulimit values:

- nofile: The maximum number of open files. Recommended value: 65536
- nproc: The maximum number of processes. Recommended value: 65536

To set system-wide ulimits to these values, log in as root and add the following lines to the /etc/security/limits.conf file on every host in your cluster:

```
* - nofile 65536
* - nproc 65536
```

To set only the root user ulimits to these values, log in as root and add the following lines to the /etc/security/limits.conf file.

```
root - nofile 65536
root - nproc 65536
```

You can use the ulimit -a command to view the current settings:

```
[root@node-1 /]# ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 14874
max locked memory (kbytes, -l) 64
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 10240
cpu time (seconds, -t) unlimited
max user processes (-u) 14874
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
```

You can also use the ulimit command to dynamically set these limits until the next reboot. This method sets a temporary value that will revert to the settings in the /etc/security/limits.conf file after the next reboot, but it is useful for experimenting with limit settings. For example:

```
[root@node-1 /]# ulimit -n 65536
```

The updated value can then be displayed:

```
[root@node-1 /]# ulimit -n
65536
```

Configuring HTTP Cookie Persistence

Reference material for enabling HTTP cookie persistence when setting up Kerberos for non-Ambari clusters.

During HTTP authentication, a cookie is dropped. This is a persistent cookie that is valid across browser sessions. For clusters that require enhanced security, it is desirable to have a session cookie that gets deleted when the user closes the browser session.

You can use the following core-site.xml property to specify cookie persistence across browser sessions.

```
<property>
   <name>hadoop.http.authentication.cookie.persistent</name>
   <value>true</value>
</property>
```

The default value for this property is false.

Configuring HBase and ZooKeeper

How to configure HBase and ZooKeeper when setting up Kerberos for non-Ambari clusters.

Configure HBase Master

How to configure HBase Master when setting up Kerberos for non-Ambari clusters.

Procedure

Edit the $HBASE_CONF_DIR/hbase-site.xml file on your HBase Master server to add the following information.

$HBASE_CONF_DIR is the directory to store the HBase configuration files. For example, /etc/hbase/conf

HBase Master Configuration (Kerberos, non-Ambari)

There are no default values. The following are all examples.

```
<property>
        <name>hbase.master.keytab.file</name>
        <value>/etc/security/keytabs/hbase.service.keytab</value>
        <description>Full path to the Kerberos keytab file to use
                     for logging in the configured HMaster server principal.

        </description>
</property>
<property>
        <name>hbase.master.kerberos.principal</name>
        <value>hbase/_HOST@EXAMPLE.COM</value>
        <description>Ex. "hbase/_HOST@EXAMPLE.COM".
        The Kerberos principal name that should be used to run the HMaster
 process.
        The principal name should be in the form: user/hostname@DOMAIN.  If
 "_HOST" is used as the hostname portion,
        it will be replaced with the actual hostname of the running
 instance.
        </description>
</property>
```

```
<property>
        <name>hbase.regionserver.keytab.file</name>
        <value>/etc/security/keytabs/hbase.service.keytab</value>
        <description>Full path to the Kerberos keytab file to use for
 logging
        in the configured HRegionServer server principal.
        </description>
</property>
<property>
        <name>hbase.regionserver.kerberos.principal</name>
        <value>hbase/_HOST@EXAMPLE.COM</value>
        <description>Ex. "hbase/_HOST@EXAMPLE.COM".The Kerberos principal
 name thatshould be used to run the HRegionServer process.
The principal name should be in the form: user/hostname@DOMAIN.
If _HOSTis used as the hostname portion, it will be replaced with the actual
 hostname of the runninginstance.
An entry for this principal must existin the file specified in
 hbase.regionserver.keytab.file
        </description>
</property>
```

```
<!--Additional configuration specific to HBase security -->

<property>
        <name>hbase.superuser</name>
        <value>hbase</value>
        <description>List of users or groups (comma-separated), who are
 allowed full privileges, regardless of stored ACLs, across the cluster.
Only used when HBase security is enabled.
        </description>
</property>
<property>
        <name>hbase.coprocessor.region.classes</name>

 <value>org.apache.hadoop.hbase.security.token.TokenProvider,org.apache.hadoop.hbase.sec
 </value>
        <description>A comma-separated list of Coprocessors that are loaded
 by default on all tables.
        </description>
</property>
<property>
        <name>hbase.security.authentication</name>
        <value>kerberos</value>

</property>
<property>
        <name>hbase.rpc.engine</name>
        <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>

</property>
<property>
        <name>hbase.security.authorization</name>
        <value>true</value>
        <description>Enables HBase authorization. Set the value of this
 property to false to disable HBase authorization.
        </description>

</property>
<property>
        <name>hbase.coprocessor.master.classes</name>
        <value>org.apache.hadoop.hbase.security.access.AccessController</
value>
```

```
</property>
<property>
        <name>hbase.bulkload.staging.dir</name>
        <value>/apps/hbase/staging</value>
        <description>Directory in the default filesystem, owned by the hbase
 user, and has permissions(-rwx--x--x, 711) </description>

</property>
```

Related Information

Bulk loading in secure mode: HBase Secure BulkLoad

Create JAAS configuration files

How to create JAAS configuration files when setting up Kerberos for non-Ambari clusters.

Procedure

1. Create the following JAAS configuration files on the HBase Master, RegionServer, and HBase client host machines.

   These files must be created under the $HBASE_CONF_DIR directory, where $HBASE_CONF_DIR is the directory to store the HBase configuration files. For example, /etc/hbase/conf.

   a) On each machine running an HBase server, create the hbase-server.jaas file under the /etc/hbase/conf directory. HBase servers include the HMaster and RegionServer. In this file, add the following content:

   ```
   Client {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   storeKey=true
   useTicketCache=false
   keyTab="/etc/security/keytabs/hbase.service.keytab"
   principal="hbase/$fully.qualified.domain.name";
   };
   ```

   a) On HBase client machines, create the hbase-client.jaas file under the /etc/hbase/conf directory and add the following content:

   ```
   Client {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=false
   useTicketCache=true;
   };
   ```

2. Create the following JAAS configuration files on the ZooKeeper Server and client host machines.

   These files must be created under the $ZOOKEEPER_CONF_DIR directory, where $ZOOKEEPER_CONF_DIR is the directory to store the HBase configuration files. For example, /etc/zookeeper/conf:

   a) On ZooKeeper server host machines, create the zookeeper-server.jaas file under the /etc/zookeeper/conf directory and add the following content:

   ```
   Server {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   storeKey=true
   useTicketCache=false
   keyTab="/etc/security/keytabs/zookeeper.service.keytab"
   principal="zookeeper/$ZooKeeper.Server.hostname";
   };
   ```

b) On ZooKeeper client host machines, create the zookeeper-client.jaas file under the /etc/zookeeper/conf directory and add the following content:

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=false
useTicketCache=true;
};
```

3. Edit the hbase-env.sh file on your HBase server to add the following information:

   Where HBASE_CONF_DIR is the HBase configuration directory. For example, /etc/hbase/conf.

```
export HBASE_OPTS ="-Djava.security.auth.login.config=$HBASE_CONF_DIR/
hbase-client.jaas"
export HBASE_MASTER_OPTS ="-
Djava.security.auth.login.config=$HBASE_CONF_DIR/hbase-server.jaas"
export HBASE_REGIONSERVER_OPTS="-
Djava.security.auth.login.config=$HBASE_CONF_DIR/hbase-server.jaas"
```

4. Edit zoo.cfg file on your ZooKeeper server to add the following information:

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

5. Edit zookeeper-env.sh file on your ZooKeeper server to add the following information:

   Where $ZOOKEEPER_CONF_DIR is the ZooKeeper configuration directory. For example, /etc/zookeeper/conf.

```
export SERVER_JVMFLAGS ="-
Djava.security.auth.login.config=$ZOOKEEPER_CONF_DIR/zookeeper-
server.jaas"
export CLIENT_JVMFLAGS ="-
Djava.security.auth.login.config=$ZOOKEEPER_CONF_DIR/zookeeper-
client.jaas"
```

Start HBase and ZooKeeper services

How to start HBase and ZooKeeper services when setting up Kerberos for non-Ambari clusters.

Procedure

Start the HBase and ZooKeeper services using the instructions provided in the HDP Reference Manual, Starting HDP Services.

If the configuration is successful, you should see the following in your ZooKeeper server logs:

```
11/12/05 22:43:39 INFO zookeeper.Login: successfully logged in.
11/12/05 22:43:39 INFO server.NIOServerCnxnFactory: binding to port
 0.0.0.0/0.0.0.0:2181
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh thread started.
11/12/05 22:43:39 INFO zookeeper.Login: TGT valid starting at:        Mon
 Dec 05 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT expires:                  Tue
 Dec 06 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh sleeping until: Tue Dec
 06 18:36:42 UTC 2011
..
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler:
  Successfully authenticated client: authenticationID=hbase/
ip-10-166-175-249.us-west-1.compute.internal@HADOOP.LOCALDOMAIN;
```

```
  authorizationID=hbase/ip-10-166-175-249.us-
west-1.compute.internal@HADOOP.LOCALDOMAIN.
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler: Setting authorizedID:
 hbase
11/12/05 22:43:59 INFO server.ZooKeeperServer: adding SASL authorization for
 authorizationID: hbase
```

Configure Secure Client-Side Access for HBase

How to configure secure client-side access for HBase when setting up Kerberos for non-Ambari clusters.

About this task

HBase configured for secure client access is expected to be running on top of a secure HDFS cluster. HBase must be able to authenticate to HDFS services.

Procedure

1. Provide a Kerberos principal to the HBase client user using the instructions provided in “Creating Service Principals and Keytab Files for HDP”.

Option	Steps
Provide Kerberos principal to normal HBase clients.	For normal HBase clients, Hortonworks recommends setting up a password to the principal. Set maxrenewlife. The client principal's maxrenewlife should be set high enough so that it allows enough time for the HBase client process to complete. Client principals are not renewed automatically. For example, if a user runs a long-running HBase client process that takes at most three days, we might create this user's principal within kadmin with the following command: `addprinc -maxrenewlife 3days`
Provide Kerberos principal to long running HBase clients.	Set-up a keytab file for the principal and copy the resulting keytab files to where the client daemon will execute. Ensure that you make this file readable only to the user account under which the daemon will run.

2. On every HBase client, add the following properties to the $HBASE_CONF_DIR/hbase-site.xml file:

```
<property>
   <name>hbase.security.authentication</name>
    <value>kerberos</value>
</property>
```

Note:

The client environment must be logged in to Kerberos from KDC or keytab via the kinit command before communication with the HBase cluster is possible. Note that the client will not be able to communicate with the cluster if the hbase.security.authentication property in the client- and server-side site files fails to match.

```
<property>
        <name>hbase.rpc.engine</name>
```

```
        <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>

</property>
```

Related Information

Create Service Principals and Keytab Files for HDP (Non-Ambari)

Optional: Configure Client-Side Operation for Secure Operation- Thrift Gateway

How to configure a client side operation for secure operations (Thrift Gateway) when setting up Kerberos for non-Ambari clusters.

Procedure

Add the following to the $HBASE_CONF_DIR/hbase-site.xml file for every Thrift gateway:

```
<property>
      <name>hbase.thrift.keytab.file</name>
      <value>/etc/hbase/conf/hbase.keytab</value>
    </property>
    <property>
      <name>hbase.thrift.kerberos.principal</name>
      <value>$USER/_HOST@HADOOP.LOCALDOMAIN</value>
    </property>
```

Substitute the appropriate credential and keytab for $USER and $KEYTAB respectively.

The Thrift gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the Thrift gateway itself. All client access via the Thrift gateway will use the Thrift gateway's credential and have its privilege.

Optional: Configure Client-Side Operation for Secure Operation- REST Gateway

How to configure client-side a operation for secure operation (REST Gateway) when setting up Kerberos for non-Ambari clusters.

Procedure

Add the following to the $HBASE_CONF_DIR/hbase-site.xml file for every REST gateway:

```
<property>
      <name>hbase.rest.keytab.file</name>
      <value>$KEYTAB</value>
    </property>
    <property>
      <name>hbase.rest.kerberos.principal</name>
      <value>$USER/_HOST@HADOOP.LOCALDOMAIN</value>
    </property>
```

Substitute the appropriate credential and keytab for $USER and $KEYTAB respectively.

The REST gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the REST gateway itself. All client access via the REST gateway will use the REST gateway's credential and have its privilege.

Configure HBase for Access Control Lists (ACL)

How to configure HBase for ACLs when setting up Kerberos for non-Ambari clusters.

About this task

Use the following instructions to configure HBase for ACL.

Procedure

1. Open kinit as HBase user.
   a) Create a keytab for principal hbase@REALM and store it in the hbase.headless.keytab file. See instructions provided in “Creating Service Principals and Keytab Files for HDP” for creating principal and keytab file.
   b) Open kinit as HBase user. Execute the following command on your HBase Master: kinit -kt hbase.headless.keytab hbase.
2. Start the HBase shell. On the HBase Master host machine, execute the following command: hbase shell.
3. Set ACLs using HBase shell: grant '$USER', '$permissions'.

   Where:

   - $USER is any user responsible for create/update/delete operations in HBase.

     Note:

     You must set the ACLs for all those users who will be responsible for create/update/delete operations in HBase.
   - $permissions is zero or more letters from the set "RWCA": READ('R'), WRITE('W'), CREATE('C'), ADMIN('A').

Related Information

Create Service Principals and Keytab Files for HDP (Non-Ambari)

Configure Phoenix Query Server

How to configure Phoenix Query Server when setting up Kerberos for non-Ambari clusters.

About this task

The HBase configuration provides most of the settings that enable secure Kerberos environments for Phoenix. However, there are additional configuration properties that complete the setup of Kerberos security for the Phoenix Query Server.

Before you begin

The value of the hbase.security.authentication property in the $HBASE_CONF_DIR/hbase-site.xmlfile must be set to kerberos.

Procedure

1. Provide the Kerberos principal and keytab for the Phoenix Query Server in the $HBASE_CONF_DIR/hbase-site.xml file.

```
<property>
    <name>phoenix.queryserver.kerberos.principal</name>
    <value>HTTP/_HOST@EXAMPLE.COM</value>
    <description>The Kerberos principal name that should be used to run
 the Phoenix Query Server process.
    The principal name should be in the form: user/hostname@DOMAIN.  If
 "_HOST" is used as the hostname
    portion, it will be replaced with the actual hostname of the running
 instance.
    </description>
</property>

<property>
    <name>phoenix.queryserver.kerberos.keytab</name>
    <value>/etc/security/keytabs/spnego.service.keytab</value>
    <description>Full path to the Kerberos keytab file to use for logging
    in the configured Phoenix Query Server service principal.
    </description>
</property>
```

2. Add the fully-qualified domain name for each host running the Phoenix Query Server to the list of hosts that can impersonate end users in the $HADOOP_CONF_DIR/core-site.xml file. Alternatively, insert an asterisk (*) instead of host names if you want to allow all hosts to impersonate end users.

```
<property>
    <name>hadoop.proxyuser.HTTP.hosts</name>
    <value>server1.domain.com,server2.domain.com</value>
    <description>A comma-separated list of fully-qualified
    domain names of hosts running services with the Hadoop
    user "HTTP" that can impersonate end users.
    Alternatively, insert an asterisk (*) instead of
    listing host names if you want to allow all hosts to
    impersonate end users.</description>
</property>
```

```
<property>
    <name>hadoop.proxyuser.HTTP.users</name>
    <value>user1,user2</value>
    <description>A comma-separated list of groups that
    user "HTTP" can impersonate end users.
    Alternatively, insert an asterisk (*) instead of
    listing group names if you want to allow all users to
    impersonate end users.</description>
</property>
```

```
<property>
    <name>hadoop.proxyuser.HTTP.users</name>
    <value>user1,user2</value>
    <description>A comma-separated list of users that
    user "HTTP" can impersonate end users.
    Alternatively, insert an asterisk (*) instead of
    listing group names if you want to allow all users to
    impersonate end users.</description>
</property>
```

Set up One-Way Trust with Active Directory

How to set up one-way trust with AD when setting up Kerberos for non-Ambari clusters.

About this task

In environments where users from Active Directory (AD) need to access Hadoop Services, set up one-way trust between Hadoop Kerberos realm and the AD (Active Directory) domain.

Note:

Hortonworks recommends setting up one-way trust after fully configuring and testing your Kerberized Hadoop Cluster.

Procedure

1. Configure Kerberos Hadoop Realm on the AD DC:

   Configure the Hadoop realm on the AD DC server and set up the one-way trust.

   a) Add the Hadoop Kerberos realm and KDC host to the DC: ksetup /addkdc $hadoop.realm $KDC-host.

   b) Establish one-way trust between the AD domain and the Hadoop realm: netdom trust $hadoop.realm /Domain: $AD.domain /add /realm /passwordt:$trust_password.

   c) (Optional) If Windows clients within the AD domain need to access Hadoop Services, and the domain does not have a search route to find the services in Hadoop realm, run the following command to create a hostmap for Hadoop service host: ksetup /addhosttorealmmap $hadoop-service-host $hadoop.realm.

Note:

Run the above for each $hadoop-host that provides services that need to be accessed by Windows clients. For example, Oozie host, WebHCat host, etc.

d) (Optional) Define the encryption type: ksetup /SetEncTypeAttr $hadoop.realm $encryption_type.

Set encryption types based on your security requirements. Mismatched encryption types cause problems.

Note:

Run ksetup /GetEncTypeAttr $krb_realm to list the available encryption types. Verify that the encryption type is configured for the Hadoop realm in the krb5.conf.

2. Configure the AD Domain on the KDC and Hadoop cluster hosts:

Add the AD domain as a realm to the krb5.conf on the Hadoop cluster hosts. Optionally configure encryption types and UDP preferences.

a) Open the krb5.conf file with a text editor and make the following changes:

1. To libdefaults, add the following properties.

- Set the Hadoop realm as default:

```
[libdefaults]
default_domain = $hadoop.realm
```

- Set the encryption type:

```
[libdefaults]
default_tkt_enctypes = $encryption_types
default_tgs_enctypes = $encryption_types
permitted_enctypes = $encryption_types
```

where the $encryption_types match the type supported by your environment.

For example:

```
default_tkt_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-
md5 des-cbc-md5 des-cbc-crc
default_tgs_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-
md5 des-cbc-md5 des-cbc-crc
permitted_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-
md5 des- cbc-md5 des-cbc-crc
```

- If TCP is open on the KDC and AD Server:

```
[libdefaults]
udp_preference_limit = 1
```

2. Add a realm for the AD domain:

```
[realms]
$AD.DOMAIN = {
kdc = $AD-host-FQDN
admin_server = $AD-host-FQDN
default_domain = $AD-host-FQDN
}
```

3. Save the krb5.conf changes to all Hadoop Cluster hosts.

b) Add the trust principal for the AD domain to the Hadoop MIT KDC:

```
kadmin
kadmin:addprinc krbtgt/$hadoop.realm@$AD.domain
```

This command will prompt you for the trust password. Use the same password as the earlier step.

Note:

If the encryption type was defined, then use the following command to configure the AD principal:

```
kadmin:addprinc -e "$encryption_type"krbtgt/$hadoop. realm@
$AD.domain
```

When defining encryption, be sure to also enter the encryption type (e.g., 'normal')

Configuring Proxy Users

Where to find information on how to configure proxy users when setting up Kerberos for non-Ambari clusters.

For information about configuring a superuser account that can submit jobs or access HDFS on behalf of another user, see the following information on the Apache site: “Proxy user - Superusers Acting on Behalf of Other Users”.

Related Information

Proxy user - Superusers Acting on Behalf of Other Users

Configure Non-Ambari Ranger SSL

How to configure Ranger SSL in an non-Ambari cluster.

Configuring Non-Ambari Ranger SSL Using Public CA Certificates

If you have access to Public CA issued certificates, use the following steps to configure non-Ambari Ranger SSL.

Configure Ranger Admin

How to configure Ranger Admin, when setting up non-Ambari Ranger SSL using Public CA certificates.

Procedure

1. Stop Ranger Admin: ranger-admin stop.
2. Open the ranger-admin-site.xml file in a text editor: vi /usr/hdp/current/ranger-admin/ews/webapp/WEB-INF/classes/conf/ranger-admin-site.xml.
3. Update ranger-admin-site.xml as follows:
   - ranger.service.http.port -- Comment out the value for this property.
   - ranger.service.http.enabled -- Set the value of this property to false.
   - ranger.service.https.atrrib.ssl.enabled -- Set the value of this property to true.
   - ranger.service.https.port -- Make sure that this port is available, or change the value to an available port number.
   - ranger.https.attrib.keystore.file -- Provide the location of the Public CA issued keystore file.
   - ranger.service.https.attrib.keystore.pass -- Enter the password for the keystore.
   - ranger.service.https.attrib.keystore.keyalias -- Enter the alias name for the keystore private key.
   - ranger.externalurl -- Set the value of this property in the format: https://<hostname of policy manager>:<https port>.
4. Add or update the following properties with the values shown below:

```
<property>
   <name>ranger.service.https.attrib.clientAuth</name>
   <value>want</value>
</property>

<property>
   <name>ranger.service.https.attrib.client.auth</name>
```

```
   <value>want</value>
</property>

<property>
   <name>ranger.https.attrib.keystore.file</name>
   <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
 </property>

<property>
   <name>ranger.service.https.attrib.keystore.file</name>
   <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
 </property>
```

5. Save the changes to ranger-admin-site.xml, then use the following command to start Ranger Admin: ranger-admin start.

Results

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the ranger.service.https.port property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS.

Configure Ranger Usersync

How to configure Ranger Usersync, when configuring non-Ambari Ranger SSL using public CA certificates.

Procedure

1. Stop the Ranger Usersync service:

```
ranger-usersync stop
```

2. Change to the Usersync install directory and open the install.properties file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/
vi install.properties
```

3. Set the value of POLICY_MGR_URL in the format: https://<hostname of policy manager>:<https port> and save your changes.
4. Run the following commands to install the new settings.

```
cd /usr/hdp/current/ranger-usersync/
./setup.sh
```

5. Start the Ranger Usersync service: ranger-usersync start.

Configure the Ranger HDFS Plugin for SSL

How to configure Ranger plugins, when configuring non-Ambari Ranger SSL using public CA certificates.

About this task

This section shows how to configure the non-Ambari Ranger HDFS plugin for SSL. You can use the same procedure for other Ranger components.

Procedure

1. Stop the NameNode: su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop namenode".
2. Open the HDFS install.properties file in a text editor: vi /usr/hdp/<version>/ranger-hdfs-plugin/install.properties.
3. Update install.properties as follows:
   - POLICY_MGR_URL -- Set this value in the format: https://<hostname of policy manager>:<https port>

- SSL_KEYSTORE_FILE_PATH -- The path to the location of the Public CA issued keystore file.
- SSL_KEYSTORE_PASSWORD -- The keystore password.
- SSL_TRUSTSTORE_FILE_PATH -- The truststore file path.
- SSL_TRUSTSTORE_PASSWORD -- The truststore password.

4. See if JAVA_HOME is available: echo $JAVA_HOME.
5. If JAVA_HOME is not available , use the following command to set JAVA_HOME (Note that Ranger requires java 1.8): export JAVA_HOME=<path for java 1.8>.
6. Run the following commands to switch to the HDFS plugin install directory and run the install agent to update the plugin with the new configuration settings.

```
cd /usr/hdp/<version>/ranger-hdfs-plugin/
./enable-hdfs-plugin.sh
```

7. Log into the Ranger Policy Manager UI as the admin user. Click the Edit button of your repository (in this case, hadoopdev) and provide the CN name of the keystore as the value for Common Name For Certificate, then save your changes.
8. Start the NameNode: su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh start namenode".
9. In the Policy Manager UI, select Audit > Plugins. You should see an entry for your repo name with HTTP Response Code 200.

Configuring a Self-Signed Certificate (Non-Ambari Ranger SSL)

If you do not have access to Public CA issued certificates, use the following steps to create a self-signed certificate and configure Non-Ambari Ranger SSL.

Configure Ranger Admin

How to configure Ranger Admin, when setting up non-Ambari Ranger SSL using self-signed certificates.

Procedure

1. Stop Ranger Admin: ranger-admin stop.
2. Change to the Ranger Admin directory and create a self-signed certificate.

```
cd /etc/ranger/admin/conf
keytool -genkey -keyalg RSA -alias rangeradmin -keystore ranger-admin-
keystore.jks -storepass xasecure -validity 360 -keysize 2048
chown ranger:ranger ranger-admin-keystore.jks
chmod 400 ranger-admin-keystore.jks
```

   a) When prompted, provide the host name as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore.
   b) When prompted for your password, press the Enter key. This will not work for Java keytool version 1.5.

3. Open the ranger-admin-site.xml file in a text editor: vi /usr/hdp/current/ranger-admin/ews/webapp/WEB-INF/classes/conf/ranger-admin-site.xml.
4. Update ranger-admin-site.xml as follows:

   - ranger.service.http.port -- Comment out the value for this property.
   - ranger.service.http.enabled -- Set the value of this property to false.
   - ranger.service.https.atrrib.ssl.enabled -- Set the value of this property to true.
   - ranger.service.https.port -- Make sure that this port is available, or change the value to an available port number.
   - ranger.https.attrib.keystore.file -- Provide the location of the keystore file created previously:/etc/ranger/admin/conf/ranger-admin-keystore.jks.
   - ranger.service.https.attrib.keystore.pass -- Enter the password for the keystore (in this case, xasecure).

- ranger.service.https.attrib.keystore.keyalias -- Enter the alias name for the keystore private key (in this case, rangeradmin).
- ranger.externalurl -- Set the value of this property in the format: https://<hostname of policy manager>:<https port>.

5. Add or update the following properties with the values shown below:

```
<property>
  <name>ranger.service.https.attrib.clientAuth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.service.https.attrib.client.auth</name>
  <value>want</value>
</property>

<property>
  <name>ranger.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
</property>

<property>
  <name>ranger.service.https.attrib.keystore.file</name>
  <value>/etc/ranger/admin/conf/ranger-admin-keystore.jks</value>
 </property>
```

6. Save the changes to ranger-admin-site.xml, then use the following command to start Ranger Admin: ranger-admin start.

Results

When you attempt to access the Ranger Admin UI with the HTTPS protocol on the port specified by the ranger.service.https.port property, the browser should report that it does not trust the site. Click Proceed anyway and you should be able to access the Ranger Admin UI over HTTPS with the self-signed cert you just created.

Configure Ranger Usersync

How to configure Ranger Usersync, when setting up non-Ambari Ranger SSL using self-signed certificates.

Procedure

1. Stop the Ranger Usersync service: ranger-usersync stop.
2. Check to see if unixauthservice.jks is in the /etc/ranger/usersync/conf/ directory. If not, run the following commands in the CLI:

```
cd /etc/ranger/usersync/conf/
mkdir cert
keytool -genkeypair -keyalg RSA -alias selfsigned -keystore /
etc/ranger/usersync/conf/cert/unixauthservice.jks -keypass
 UnIx529p -storepass UnIx529p -validity 3600 -keysize 2048 -dname
 'cn=unixauthservice,ou=authenticator,o=mycompany,c=US'
chown -R ranger:ranger /etc/ranger/usersync/conf/cert
chmod -R 400 /etc/ranger/usersync/conf/cert
```

3. Change to the Usersync install directory and open the install.properties file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/
vi install.properties
```

4. Set the value of POLICY_MGR_URL in the format: https://<hostname of policy manager>:<https port> and save your changes.

5. Create a truststore for the Ranger Admin's self-signed keystore. When prompted for a password, press the Enter key.

```
cd /etc/ranger/usersync/conf/
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks
 -alias rangeradmin -file ranger-admin-trust.cerchown -R ranger:ranger /
etc/ranger/usersync/conf/cert
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -
keystore mytruststore.jks -storepass changeit
chown ranger:ranger mytruststore.jks
```

6. Change to the Usersync conf directory and open the ranger-ugsync-site.xml file in a text editor.

```
cd /usr/hdp/current/ranger-usersync/conf/
vi ranger-ugsync-site.xml
```

7. Edit the following properties, then save your changes:
   - ranger.usersync.truststore.file -- Enter the path to the truststore file.
   - ranger.usersync.truststore.password -- Enter the truststore password.
8. Run the following commands to install the new settings.

```
cd /usr/hdp/current/ranger-usersync/
./setup.sh
```

9. Start the Ranger Usersync service: ranger-usersync start.

Configure Ranger Plugins

How to configure Ranger Plugins, when setting up non-Ambari Ranger SSL using self-signed certificates.

About this task

The following steps describe how to configure the Ranger HDFS plugin for SSL with a self-signed certificate in a non-Ambari cluster. You can use the same procedure for other Ranger components.

Procedure

1. Stop the NameNode: su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop namenode".
2. Change to the Ranger HDFS plugin directory and create a self-signed certificate.

```
cd /etc/hadoop/conf
keytool -genkey -keyalg RSA -alias rangerHdfsAgent -keystore ranger-
plugin-keystore.jks -storepass myKeyFilePassword -validity 360 -keysize
 2048
chown hdfs:hdfs ranger-plugin-keystore.jks
chmod 400 ranger-plugin-keystore.jks
```

   a) When prompted, provide an identifiable string as the value for the "What is your first and last name?" question. then provide answers to the subsequent questions to create the keystore.
   b) When prompted for a password, press the Enter key.

Note:

Important note: In the case where multiple servers talking to ranger admin for downloading policies for the same service/repository (e.g. HBase Master and Region servers, multiple NameNodes or Hive servers in an HA environment, be sure to use the repo name or a common string across all of the nodes (such as HbasePlugin, HdfsPlugin etc). (Note this and enter the same value in Common Name For Certificate field in the edit repository page in the Policy Manager UI).

3. Create a truststore for the agent and add the Admin public key as a trusted entry. When prompted for a password, press the Enter key.

```
cd /etc/hadoop/conf
keytool -export -keystore /etc/ranger/admin/conf/ranger-admin-keystore.jks
 -alias rangeradmin -file ranger-admin-trust.cer
keytool -import -file ranger-admin-trust.cer -alias rangeradmintrust -
keystore ranger-plugin-truststore.jks -storepass changeit
chown hdfs:hdfs ranger-plugin-truststore.jks
chmod 400 ranger-plugin-truststore.jks
```

4. Open the HDFS install.properties file in a text editor: vi /usr/hdp/<version>/ranger-hdfs-plugin/install.properties.
5. Update install.properties as follows:
   - POLICY_MGR_URL -- Set this value in the format: https://<hostname of policy manager>:<https port>
   - SSL_KEYSTORE_FILE_PATH -- The path to the location of the keystore file.
   - SSL_KEYSTORE_PASSWORD -- The keystore password.
   - SSL_TRUSTSTORE_FILE_PATH -- The truststore file path.
   - SSL_TRUSTSTORE_PASSWORD -- The truststore password.
6. See if JAVA_HOME is available: echo $JAVA_HOME.
7. If JAVA_HOME is not available , use the following command to set JAVA_HOME (Note that Ranger requires java 1.8): export JAVA_HOME=<path for java 1.8>.
8. Run the following commands to switch to the HDFS plugin install directory and run the install agent to update the plugin with the new configuration settings.

```
cd /usr/hdp/<version>/ranger-hdfs-plugin/
./enable-hdfs-plugin.sh
```

9. Stop Ranger Admin: ranger-admin stop.
10. Add the agent's self-signed cert to the Admin's trustedCACerts.
11. Start Ranger Admin.ranger-admin start.

```
cd /etc/ranger/admin/conf
keytool -export -keystore /etc/hadoop/conf/ranger-plugin-keystore.jks
 -alias rangerHdfsAgent -file ranger-hdfsAgent-trust.cer -storepass
 myKeyFilePassword
keytool -import -file ranger-hdfsAgent-trust.cer -alias
 rangerHdfsAgentTrust -keystore <Truststore file used by Ranger Admin -
 can be the JDK cacerts> -storepass changeit
```

12. Log into the Ranger Policy Manager UI as the admin user. Click the Edit button of your repository (in this case, hadoopdev) and provide the CN name of the keystore as the value for Common Name For Certificate, then save your changes.
13. Start the NameNode.su -l hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh start namenode".
14. In the Policy Manager UI, select Audit > Plugins. You should see an entry for your repo name with HTTP Response Code 200.

Enable Audit Logging in Non-Ambari Clusters

How to enable audit logging for HDFS.

About this task

It is recommended that Ranger audits be written to both Solr and HDFS. Audits to Solr are primarily used to enable queries from the Ranger Admin UI. HDFS is a long-term destination for audits; audits stored in HDFS can be exported to any SIEM system, or to another audit store.

Procedure

1. Enable audit logging:
   a) Set the XAAUDIT.HDFS.ENABLE value to "true" for the component plug-in in the install.properties file, which can be found here:/usr/hdp/<version>/ranger-<component>=plugin.
   b) Configure the NameNode host in the XAAUDIT.HDFS.HDFS_DIR field.
   c) Create a policy in the HDFS service from the Ranger Admin for individual component users (hive/hbase/knox/storm/yarn/kafka/kms) to provide READ and WRITE permissions for the audit folder (i.e., for enabling Hive component to log Audits to HDFS, you need to create a policy for the hive user with Read and WRITE permissions for the audit directory).
   d) Set the Audit to HDFS caches logs in the local directory, which can be specified in XAAUDIT.HDFS.LOCAL_BUFFER_DIRECTORY (this can be like /var/log/<component>/**), which is the path where the audit is stored for a short time. This is similar for archive logs that need to be updated.
2. Enable auditing reporting from the Solr database:
   a) Modify the following properties in the Ranger service install.properties to enable auditing to the Solr database in Ranger:
      - audit_store=solr
      - For HDP Search's Solr Instance: http:<solr_host>:8983/solr/ranger_audits

        For Ambari Infra's Solr Instance: http:<solr_host>:8886/solr/ranger_audits
      - audit_solr_user=ranger_solr
      - audit_solr_password-NONE
   b) Restart Ranger: service ranger-admin restart.
3. Enable auditing to the Solr database for a plug-in (e.g., HBase):
   a) Set the following properties in install.properties of the plug-in to begin audit logging to the Solr database:
      - XAAUDIT.SOLR.IS.ENABLED=true
      - XAAUDIT.SOLR.ENABLE=true
      - For HDP Search's Solr Instance: XAAUDIT.SOLR.URL=http://solr_host:8983/solr/ranger_audits

        For Ambari Infra's Solr Instance: XAAUDIT.SOLR.URL=http://solr_host:8886/solr/ranger_audits
      - XAAUDIT.SOLR.USER-ranger_solr
      - XAAUDIT.SOLR.PASSWORD=NONE
      - XAAUDIT.SOLR.FILE_SPOOL_DIR=/var/log/hadoop/hdfs/audit/solr/spool
   b) Enable the Ranger HBase plug-in.
   c) Restart the HBase component.

Knox Reference

This chapter is a collection of Knox content for non-default use cases.

Configuring Gateway Security

The Knox Gateway offers some security features: “Web Application Security” and “Configuring Knox With a Secured Hadoop Cluster”.

Implementing Web Application Security

The Knox Gateway is a Web API (REST) Gateway for Hadoop clusters. REST interactions are HTTP based, and therefore the interactions are vulnerable to a number of web application security vulnerabilities. The web application security provider allows you to configure protection filter plugins.

Configure a Protection Filter Against CSRF

A Cross Site Request Forgery (CSRF) attack attempts to force a user to execute functionality without their knowledge. Typically the attack is initiated by presenting the user with a link or image that when clicked invokes a request to another site with which the user already has an established an active session. CSRF is typically a browser based attack.

About this task

The only way to create a HTTP request from a browser with a custom HTTP header is to use Javascript XMLHttpRequest or Flash, etc. Browsers have built-in security that prevent web sites from sending requests to each other unless specifically allowed by policy. This means that a website www.bad.com cannot send a request to http://bank.example.com with the custom header X-XSRF-Header unless they use a technology such as a XMLHttpRequest. That technology would prevent such a request from being made unless the bank.example.com domain specifically allowed it. This then results in a REST endpoint that can only be called via XMLHttpRequest (or similar technology).

Procedure

1. Open the cluster topology descriptor file, $cluster-name .xml, in a text editor.
2. Add a WebAppSec webappsec provider to topology/gateway with a parameter for each service as follows:

```
<provider>
    <role>webappsec</role>
    <name>WebAppSec</name>
    <enabled>true</enabled>
    <param>
        <name>csrf.enabled</name>
        <value>$csrf_enabled</value>
    </param>
    <param><!-- Optional -->
        <name>csrf.customHeader</name>
        <value>$header_name</value>
    </param>
    <param><!-- Optional -->
        <name>csrf.methodsToIgnore</name>
        <value>$HTTP_methods</value>
    </param>
</provider>
```

   where:

   - $csrf_enabled is either true or false.
   - $header_name when the optional parameter csrf.customHeader is present the value contains the name of the header that determines if the request is from a trusted source. The default, X-XSRF-Header, is described by the NSA in its guidelines for dealing with CSRF in REST.
   - $http_methods when the optional parameter csrf.methodsToIgnore is present the value enumerates the HTTP methods to allow without the custom HTTP header. The possible values are GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, or PATCH. For example, specifying GET allows GET requests from the address bar of a browser.

3. Save the file.
   The gateway creates a new WAR file with modified timestamp in $gateway /data/deployments.

What to do next

Validate CSRF Filtering

The following curl command can be used to request a directory listing from HDFS while passing in the expected header X-XSRF-Header.

```
curl -k -i --header "X-XSRF-Header: valid" -v -u guest:guest-password
 https://localhost:8443/gateway/sandbox/webhdfs/v1/tmp?op=LISTSTATUS
```

Note:

The above LISTSTATUS request only works if you remove the GET method from the csrf.methodsToIgnore list.

Omitting the –header “X-XSRF-Header: valid” above results in an HTTP 400 bad_request. Disabling the provider, by setting csrf.enabled to false allows a request that is missing the header.

Knox Admin UI Quicklink Requirements for Unsecured Clusters

If you are in an unsecured cluster and navigate to the Knox Admin UI via quicklink, and the UI loads but no topologies are visible, some configurations must be adjusted for your deployment.

Context

The Knox Admin UI is hosted in a topology called manager.xml; this topology is not manageable by Ambari. It is manageable by Knox Admin UI, but there is an access issue before changing your configurations. Rather than adding another topology to the fixed set managed by Ambari, a number of enhancements have been made to have reasonable defaults and centralized configuration for admin capabilities within gateway-site.xml.

Defaults/Central Config

Knox Admin UI no longer uses HTTP Basic Auth, but instead uses KnoxSSO.

KnoxSSO is set up out-of-the-box to still use the DEMO LDAP server for Knox. It needs to be configured for enterprise AD/LDAP or another SSO mechanism as appropriate for the deployment.

Authorization checks within the manager.xml and admin.xml topologies now default to gateway-site.xml properties called gateway.knox.admin.users and gateway.knox.admin.groups. These are comma-separated lists of users and groups that should have access to the admin capabilities in manager.xml and admin.xml topologies.

In order for the groups capability to work out of the box, it is assumed that local OS accounts with groups are available on the Knox machine. This is very often the case for secure clusters but not necessarily for unsecured clusters. In unsecured clusters, it is possible that LDAP configuration will need to be added to gateway-site.xml. This is done via the Hadoop Group Provider values with a specific prefix for this use: gateway.group.config. All of the config that begins with that prefix will be found and used to configure the group lookup mechanism for the deployment. See the following for more details on Hadoop Group Provider: “Hadoop Group Lookup Identity Assertion Provider”. Since the admin topologies have already been seeded with the prefix to look for, the configuration only needs to be added to the gateway-site.xml and the server restarted.

The manager.xml and admin.xml topologies have been defaulted to be considered auto-deploy topologies since they now depend on gateway-site.xml config and need to be redeployed to uptake that config. They should automatically do so on gateway restart but if for some reason they don't they can be redeployed manually by touching the files or using the Knox CLI to redeploy them from the Knox machine/s.

Set up the Knox Token Service for Ranger APIs

How to configure the Knox Token Service for Ranger APIs.

About this task

Once logged into Knox SSO, the UI service uses a cookie named hadoop-jwt. The Knox Token Service enables clients to acquire this same JWT token to use for accessing REST APIs. By acquiring the token and setting it as a bearer token on a request, a client is able to access REST APIs that are protected with the JWT Federation Provider.

Procedure

The Knox Token Service configuration can be configured in any topology. For example, from Ambari>Knox>Configs>Advanced knoxsso-topology, add:

```
<service>
   <role>KNOXTOKEN</role>
   <param>
      <name>knox.token.ttl</name>
      <value>numeric_value_in_miliseconds</value>
   </param>
   <param>
      <name>knox.token.audiences</name>
      <value>tokenbased</value>
   </param>
   <param>
      <name>knox.token.target.url</name>
      <value>https://host:port/gateway/tokenbased</value>
   </param>
</service>
```

where the values of the parameters are specific to your environment:

Parameter	Description	Optional/Required	Default
knox.token.ttl	The lifespan of the token in miliseconds. Once it expires, a new token must be acquired from KnoxToken service.	Required	30000 (30 seconds)
knox.token.audiences	Comma separated list of audiences to add to the JWT token. Used to ensure that a token received by a participating application knows that the token was intended for use with that application. In the event that an endpoint has expected audiences, and they are not present, the token must be rejected. In the event where the token has audiences, and the endpoint has none expected, then the token is accepted.	Optional	
knox.token.target.url	Indicates the intended endpoint for which the token may be used. The KnoxShell token credential collector can pull this URL from a knoxtokencache file to be used in scripts. Eliminates the need to prompt for or hard-code endpoints in your scripts.	Optional	

Example

From Ambari>Knox>Configs>Advanced knoxsso-topology, add:

```
<service>
   <role>KNOXTOKEN</role>
   <param>
```

```
      <name>knox.token.ttl</name>
      <value>36000000</value>
   </param>
   <param>
      <name>knox.token.audiences</name>
      <value>tokenbased</value>
   </param>
   <param>
      <name>knox.token.target.url</name>
      <value>https://localhost:8443/gateway/tokenbased</value>
   </param>
</service>
```

What to do next

Acquire a token from the Knox Token service as configured in the sandbox topology

```
curl -ivku guest:guest-password https://localhost:8443/gateway/sandbox/
knoxtoken/api/v1/token
```

Resulting in a JSON response that contains the token, the expiration and the optional target endpoint:

```
`{"access_token":"eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJndWVzdCIsImF1ZCI6InRva2VuYmFzZWQiLCJpc
lXM3tuEYuZPaZVf8PenzjtBbuQsCg9VVImuu2r1YNVJlcTQ7OV-
eW50L6OTI0uZfyrFwX6C7jVhf7d7YR1NNxs4eVbXpS1TZ5fDIRSfU3MU","target_url":"https://
localhost:8443/gateway/tokenbased","token_type":"Bearer
 ","expires_in":1489942188233}`
```

The following curl example shows how to add a bearer token to an Authorization header:

```
curl -ivk -H "Authorization: Bearer
 eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJndWVzdCIsImF1ZCI6InRva2VuYmFzZWQiLCJpc3MiOiJLTk9YU1NPI
lXM3tuEYuZPaZVf8PenzjtBbuQsCg9VVImuu2r1YNVJlcTQ7OV-
eW50L6OTI0uZfyrFwX6C7jVhf7d7YR1NNxs4eVbXpS1TZ5fDIRSfU3MU" https://
localhost:8443/gateway/tokenbased/webhdfs/v1/tmp?op=LISTSTATUS
```

Ambari CLI Wizard for Knox SSO Reference

Reference information, including syntax, for the command ambari-server setup-sso. This command is used to set up and configure Knox SSO.

Silent Mode

In silent mode all configuration details may be set on the command line via arguments. However, one argument is a password that is needed for authentication to use Ambari's REST API. This argument may be left off of the command causing the CLI to prompt for it.

For example:

```
# ambari-server setup-sso --ambari-admin-username=admin ...
Using python  /usr/bin/python
Setting up SSO authentication properties...
Enter Ambari Admin password:
```

You can run ambari-server setup-sso --help for syntax help.

Required Syntax, Enabling SSO

```
 --sso-enabled=true
```

```
                        Indicates whether to enable/disable SSO
  --sso-enabled-ambari=<true|false>
                        Indicates whether to enable/disable SSO
authentication
                        for Ambari, itself
  --sso-manage-services=<true|false>
                        Indicates whether Ambari should manage the SSO
                        configurations for specified services
  --sso-enabled-services=<service list>
                        A comma separated list of services that are expected
                        to be configured for SSO (you are allowed to use '*'
                        to indicate ALL services)
  --sso-provider-url=<URL>
                        The URL of SSO provider; this must be provided when
                        --sso-enabled is set to 'true'
  --sso-public-cert-file=SSO_PUBLIC_CERT_FILE
                        The path where the public certificate PEM is
located;
                        this must be provided when --sso-enabled is set to
                        'true'
  --ambari-admin-username=<username>
                        Ambari administrator username for accessing Ambari's
                        REST API
```

Optional Syntax, Enabling SSO

```
  --sso-jwt-cookie-name=<cookie name>
                        The name of the JWT cookie
                        Default value: hadoop-jwt
  --sso-jwt-audience-list=<audience list>
                        A comma separated list of JWT audience(s)
                        Default value <empty>
  --ambari-admin-password=<password>
                        Ambari administrator password for accessing Ambari's
                        REST API
```

Required Syntax, Disabling SSO

```
  --sso-enabled=false
                        Indicates whether to enable/disable SSO
  --ambari-admin-username=AMBARI_ADMIN_USERNAME
                        Ambari administrator username for accessing Ambari's
                        REST API
```

Optional Syntax, Disabling SSO

```
  --ambari-admin-password=<password>
                        Ambari administrator password for accessing Ambari's
                        REST API
```

Interactive Mode

In interactive mode some configuration details may be set on the command line via arguments and the CLI will prompt for the rest.

```
# ambari-server setup-sso
Using python  /usr/bin/python
Setting up SSO authentication properties...
Enter Ambari Admin login: admin
Enter Ambari Admin password:
```

```
SSO is currently not configured
Do you want to configure SSO authentication [y/n] (y)? y
Provider URL (https://knox.example.com:8443/gateway/knoxsso/api/v1/websso):
Public Certificate PEM (empty line to finish input):
MIICVTCCAb6gAwIBAgIIKwH4/V7SjxEwDQYJKoZIhvcNAQEFBQAwbTELMAkGA1UE
...
6fSqZSwbBXwFKf0gIBttufyldePpAsM7Yg==

Use SSO for Ambari [y/n] (n)? y
Manage SSO configurations for eligible services [y/n] (n)? y
 Use SSO for all services [y/n] (n)? n
   Use SSO for ATLAS [y/n] (n)? y
JWT Cookie name (hadoop-jwt):
JWT audiences list (comma-separated), empty for any ():
Ambari Server 'setup-sso' completed successfully.
```

In either case, the CLI collects the data and submits it to Ambari via the REST API. This then triggers processes in Ambari to enable or disable SSO as needed.

SSO via the REST API

The SSO configuration may be managed using Ambari's REST API, via the following entry point: /api/v1/services/AMBARI/components/AMBARI_SERVER/configurations.

This entry point supports the following request types:

GET - retrieve the SSO configuration data
POST - explicitly set the SSO configuration data, replacing all properties
PUT - update the SSO configuration data, only the specified properties are updated
DELETE - removes the SSO configuration data

Getting the SSO Configuration

To retrieve the SSO configuration data:

```
GET /api/v1/services/AMBARI/components/AMBARI_SERVER/configurations/sso-
configuration
```

Example 404 response:

```
{
  "status" : 404,
  "message" : "The requested resource doesn't exist:
 RootServiceComponentConfiguration not found where Configuration/
service_name=AMBARI AND Configuration/component_name=AMBARI_SERVER AND
 Configuration/category=sso-configuration."
}
```

Example 200 response:

```
{
  "href" : "http://ambari_server.host:8080/api/v1/services/AMBARI/
components/AMBARI_SERVER/configurations/sso-configuration",
  "Configuration" : {
    "category" : "sso-configuration",
    "component_name" : "AMBARI_SERVER",
    "service_name" : "AMBARI",
    "properties" : {
      "ambari.sso.authentication.enabled" : "false",
      "ambari.sso.enabled_services" : "AMBARI, ATLAS",
      "ambari.sso.jwt.audiences" : "",
```

```
      "ambari.sso.jwt.cookieName" : "hadoop-jwt",
      "ambari.sso.manage_services" : "true",
      "ambari.sso.provider.certificate" : "-----BEGIN CERTIFICATE-----
\nMIIC...TYptEVg==\n-----END CERTIFICATE-----",
      "ambari.sso.provider.originalUrlParamName" : "originalUrl",
      "ambari.sso.provider.url" : "https://knox.host:8443/gateway/knoxsso/
api/v1/websso"
    },
    "property_types" : {
      "ambari.sso.authentication.enabled" : "PLAINTEXT",
      "ambari.sso.enabled_services" : "PLAINTEXT",
      "ambari.sso.jwt.audiences" : "PLAINTEXT",
      "ambari.sso.jwt.cookieName" : "PLAINTEXT",
      "ambari.sso.manage_services" : "PLAINTEXT",
      "ambari.sso.provider.certificate" : "PLAINTEXT",
      "ambari.sso.provider.originalUrlParamName" : "PLAINTEXT",
      "ambari.sso.provider.url" : "PLAINTEXT"
    }
  }
}
```

Setting the SSO Configuration

To set the SSO configuration data, replacing any previously existing data:

```
POST /api/v1/services/AMBARI/components/AMBARI_SERVER/configurations
```

Example payload:

```
{
  "Configuration": {
    "category" : "sso-configuration",
    "properties": {
      "ambari.sso.manage_services" : "true",
      "ambari.sso.enabled_services": "AMBARI, ATLAS",
      "ambari.sso.provider.url": "https://knox.host:8443/gateway/knoxsso/
api/v1/websso",
      "ambari.sso.provider.certificate": "-----BEGIN CERTIFICATE-----
\nMIIC...TYptEVg==\n-----END CERTIFICATE-----",
      "ambari.sso.authentication.enabled": "true",
      "ambari.sso.jwt.audiences": "",
      "ambari.sso.jwt.cookieName": "hadoop-jwt",
      "ambari.sso.provider.originalUrlParamName": "originalUrl"
    }
  }
}
```

Updating the SSO Configuration

To update the SSO configuration data, only replacing or adding specific properties:

```
PUT /api/v1/services/AMBARI/components/AMBARI_SERVER/configurations/sso-
configuration
```

Example payload:

```
{
  "Configuration": {
    "properties": {
      "ambari.sso.manage_services" : "true",
      "ambari.sso.enabled_services": "AMBARI, ATLAS, RANGER",
      "ambari.sso.authentication.enabled": "false"
    }
```

```
  }
}
```

Deleting the SSO Configuration

To delete the SSO configuration data, removing all properties:

```
DELETE /api/v1/services/AMBARI/components/AMBARI_SERVER/configurations/sso-
configuration
```

Related Information

Enable Knox SSO Using the Ambari CLI

Ranger Install Reference

This chapter is a collection of Ranger install content for non-default use cases.

Use Consolidated DB Schema Script to Reduce Ranger Install Time

You can reduce Ranger cluster setup time by setting up the database and Java patches before installing Ranger. This is accomplished by running the consolidated DB schema script.

About this task

This is an optional step you can take before running the Ranger install via Ambari to speed up the Ranger install.

Procedure

1. cd /usr/hdp/current/ranger-admin
2. export JAVA_HOME=/usr/jdk64/jdk1.8.0_112
3. Run vi install.properties and edit the following properties:

   ```
   $DBFLAVOR_core_file=db/$DBFLAVOR/optimized/current/ranger_core_db_
   $DBFLAVOR.sql
   db_password=rangeradmin
   audit_solr_urls=http://localhost:6083/solr/ranger_audits
   ```

   e.g. mysql_core_file=db/mysql/optimized/current/ranger_core_db_mysql.sql
4. From ranger-admin, run: ./setup.sh
5. Before running Ranger install via Ambari, install the ranger-admin package from console: yum install ranger_$VERSION-admin.x86_64
6. dba_script.py -q
7. python db_setup.py
8. Via Ambari, run the Ranger install. During installation, turn off **Setup Database and Database User** and run **Test Connection**.