CDP Private Cloud Base 7.0.3

Security Troubleshooting for Cloudera Manager

Date published: 2020-11-30
Date modified: 2020-11-30

CLOUDERA

https://docs.cloudera.com/

Legal Notice

Contents

Troubleshooting Security Issues

Security-related issues can manifest in various ways and can be associated with various sources, including Kerberos authentication, HDFS encryption, and TLS/SSL (data in transit encryption). This section includes error messages and troubleshooting for common issues and some architectural details about some of the underlying components.

For information about Sentry troubleshooting, see "Troubleshooting Sentry".

Related Information

Getting Started: Operational database cluster

Error Messages and Various Failures

Cluster cannot run jobs after Kerberos enabled

Symptom: Cluster previously configured without Kerberos authentication may fail to run jobs for certain users on certain TaskTrackers (MRv1) or NodeManagers (YARN) after enabling Kerberos for the cluster. Errors may display in the TaskTracker or NodeManager logs. The following example errors are from TaskTracker on MRv1:

```
10/11/03 01:29:55 INFO mapred.JobClient: Task Id : attempt_201011021321_0004
_m_000011_0, Status : FAILED
Error initializing attempt_201011021321_0004_m_000011_0:
java.io.IOException: org.apache.hadoop.util.Shell$ExitCodeException:
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskControl
ler.java:212)
at org.apache.hadoop.mapred.LinuxTaskController.initializeUser(LinuxTaskC
ontroller.java:442)
at org.apache.hadoop.mapreduce.server.tasktracker.Localizer.initializeUserD
irs(Localizer.java:272)
at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.java:963)
at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.java:2209)
at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTracker.java:2
174)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
at org.apache.hadoop.util.Shell.runCommand(Shell.java:250)
at org.apache.hadoop.util.Shell.run(Shell.java:177)
at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.java:3
70)
at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskControl
ler.java:203)
... 5 more
```

Possible cause: The issue is caused by legacy content in directories for TaskTracker and NodeManager that may exist after configuring Kerberos authentication for a cluster that previously did not use Kerberos. The sequence of events leading to this issue is as follows:

1. Cluster that had not been configured for Kerberos authentication was used to run jobs, which created local user directory (or directories) on each TaskTracker or NodeManager host.
2. Cluster was then configured to use Kerberos authentication.
3. Users try running jobs on the newly secured cluster but local user directories on TaskTrackers or NodeManagers are owned by the wrong user or have overly-permissive permissions.

These directories should have been cleaned up at the time Kerberos authentication was enabled for the cluster.

Steps to resolve: Delete mapred.local.dir or yarn.nodemanager.local-dirs directories across the cluster for affected users.

NameNode fails to start

Symptom: Login failure occurs when attempting to start the NameNode. With debugging enabled on the cluster, the following KrbException messages may display:

```
Caused by: KrbException: Integrity check on decrypted field failed (31) - PR
EAUTH_FAILED}}
```

```
Caused by: KrbException: Identifier does not match expected value (906)
```

Possible cause: This issue may be due to incorrect configuration for AES-256. By default, certain operating systems —CentOS/Red Hat Enterprise Linux 5.6 (and higher), Ubuntu—use AES-256 encryption which requires installing the "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File" on all hosts (see "JCE Policy File for AES-256 Encryption" for details), or disabling AES-256 support in the kdc.conf or krb5.com (see "Disable JCE Policy File for AES-256 Encryption" for details).

Steps to resolve: KrbException 31 and KrbException 906 can be caused by various issues, but the most likely cause is incorrectly configured AES-256 encryption. Resolving the issue should start by determining the type of encryption configured for the cluster.

To verify the type of encryption configured for the cluster:

1. On the local KDC host, type this command to create a test principal:

   ```
   kadmin -q "addprinc test"
   ```

2. On a cluster host, type this command to start a Kerberos session as test:

   ```
   kinit test
   ```

3. On a cluster host, type this command to view the encryption type in use:

   ```
   klist -e
   ```

   If AES-256 is being used, output such as the following displays:

   ```
   Ticket cache: FILE:/tmp/krb5cc_0
   Default principal: test@SCM
   Valid starting     Expires            Service principal
   05/19/11 13:25:04  05/20/11 13:25:04  krbtgt/SCM@SCM
       Etype (skey, tkt): AES-256 CTS mode with 96-bit SHA-1 HMAC, AES-256
    CTS mode with 96-bit SHA-1 HMAC
   ```

To remove AES-256 encryption from the Kerberos configuration files:

- Remove aes256-cts:normal from the supported_enctypes field of the kdc.conf or krb5.conf file.
- After changing the configuration, restart the KDC and the kadmin servers.
- Change TGT principal (krbtgt/REALM@REALM) and other principal passwords as needed.
- In the [realms] section of the kdc.conf file, for the realm associated with HADOOP.LOCALDOMAIN, add (or replace if it exists already) the following variable:

  ```
  supported_enctypes = des3-hmac-sha1:normal arcfour-hmac:normal des-hmac-
  sha1:normal des-cbc-md5:normal des-cbc-crc:normal des-cbc-crc:v4 des-cbc-
  crc:afs3
  ```

- Recreate the hdfs keytab file and mapred keytab file using the instructions in "Managing Kerberos credentials using Cloudera Manager".

Clients cannot connect to NameNode

Symptom: The NameNode keytab file does not have an AES-256 entry but the client tickets do. The NameNode starts but clients cannot connect to it. The error message does not specify "AES256" but rather contains enctype code "18."

Possible cause: Issue related to AES-256 encryption and the JCE library.

Steps to resolve: Verify that Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File is installed (see "JCE Policy File for AES-256 Encryption" for instructions) or remove aes256-cts:normal from the supp orted_enctypes field of the kdc.conf or krb5.conf file, as detailed above.

Hadoop commands run in local realm but not in remote realm

Symptom: After enabling cross-realm trust, authenticating as a principal in the local realm lets you successfully run Hadoop commands, but authenticating as a principal in the remote realm does not.

Possible cause: This issue is often due to principals in the two realms having different encryption types or different passwords for the cross-realm principal in each realm. Because the local and remote realm each issue their own TGTs, the local commands run but the service ticket needed for the local and remote realms to communicate cannot be granted.

Steps to resolve: Add the cross-realm krbtgt principal and its encryption types to the MIT KDC server using kadmin.l ocal or kadmin as appropriate (local access vs. remote):

```
kadmin: addprinc -e "enc_type_list" krbtgt/LOCAL-REALM.EXAMPLE.COM@MAIN-RE
ALM.COMPANY.COM
```

Specify the types of encryption supported by the cross-realm principal (krbtgt), for example, AES, DES, or RC4. Multiple encryption types can be specified in the *enc_type_list* as long as one of the encryption types matches that of the tickets granted by the KDC in the remote realm. For example:

```
kadmin:  addprinc -e "aes256-cts:normal rc4-hmac:normal des3-hmac-sha1:norma
l" krbtgt/LOCAL-REALM.EXAMPLE.COM@MAIN-REALM.COMPANY.COM
```

Users cannot obtain credentials when running Hadoop jobs or commands

Symptom: Users attempt to authenticate but message such as the following displays:

```
13/01/15 17:44:48 DEBUG ipc.Client: Exception encountered while connecting t
o the server : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided
 (Mechanism level: Fail to create credential.
(63) - No service creds)]
```

Possible cause: Ticket message may be too large for the UDP protocol (which is used by SASL by default).

Steps to resolve: Force Kerberos to use TCP instead of UDP by adding the following parameter to libdefaults in the krb5.conf file on the clients where the problem is occurring:

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

Configure krb5.conf through Cloudera Manager, this will automatically get added to krb5.conf.

Note:

When sending a message to the KDC, the library will try using TCP before UDP if the size of the ticket message is larger than the setting specified for the udp_preference_limit property. If the ticket message is smaller than udp_preference_limit setting, then UDP will be tried before TCP. Regardless of the size, both protocols will be tried if the first attempt fails.

Bogus replay exceptions in service logs

Symptom: Multiple valid requests to Kerberos protected services are identified as *replay* attempts when they are not. The following exception shows up in the logs for one or more of the Hadoop daemons:

```
2013-02-28 22:49:03,152 INFO  ipc.Server (Server.java:doRead(571)) - IPC Ser
ver listener on 8020: readAndProcess threw exception javax.security.sasl.Sas
lException: GSS initiate failed [Caused by GSSException: Failure unspecified
 at GSS-API level (Mechanism l
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSExce
ption: Failure unspecified at GSS-API level (Mechanism level: Request is a r
eplay (34))]
        at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(Gss
Krb5Server.java:159)
        at org.apache.hadoop.ipc.Server$Connection.saslReadAndProcess(Serve
r.java:1040)
        at org.apache.hadoop.ipc.Server$Connection.readAndProcess(Server.ja
va:1213)
        at org.apache.hadoop.ipc.Server$Listener.doRead(Server.java:566)
        at org.apache.hadoop.ipc.Server$Listener$Reader.run(Server.java:363)
Caused by: GSSException: Failure unspecified at GSS-API level (Mechanism
level: Request is a replay (34))
        at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.
java:741)
        at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImp
l.java:323)
        at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.
java:267)
        at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(Gs
sKrb5Server.java:137)
        ... 4 more
Caused by: KrbException: Request is a replay (34)
        at sun.security.krb5.KrbApReq.authenticate(KrbApReq.java:300)
        at sun.security.krb5.KrbApReq.<init>(KrbApReq.java:134)
        at sun.security.jgss.krb5.InitSecContextToken.<init>(InitSecContext
Token.java:79)
        at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Contex
t.java:724)
        ... 7 more
```

This issue can also manifest as poor performance for clients of the cluster, including dropped connections, timeouts attempting to make RPC calls, and so on.

Possible cause: Kerberos uses a second-resolution timestamp to protect against replay attacks (where an attacker can record network traffic, and play back recorded requests later to gain elevated privileges). That is, incoming requests are cached by Kerberos for a little while, and if there are similar requests within a few seconds, Kerberos will be able to detect them as replay attack attempts (see "MIT Kerberos replay cache" for more information).

However, if there are multiple valid Kerberos requests coming in at the same time, these may also be misjudged as attacks for the following reasons:

- Multiple services in the cluster are using the same Kerberos principal. All secure clients that run on multiple machines should use unique Kerberos principals for each machine. For example, rather than connecting as a service principal myservice@EXAMPLE.COM, services should have per-host principals such as myservice/host123.example.com@EXAMPLE.COM.
- Clocks not synchronized: All hosts should run NTP so that clocks are kept in sync between clients and servers.

Steps to resolve:

While having different principals for each service, and clocks in sync helps mitigate the issue, there are, however, cases where even if all of the above are implemented, the problem still persists. In such a case, disabling the cache (and the replay protection as a consequence), will allow parallel requests to succeed. This compromise between usability and security can be applied by setting the KRB5RCACHETYPE environment variable to none.

Note that the KRB5RCACHETYPE is not automatically detected by Java applications. For Java-based components:

- Ensure that the cluster runs on JDK 8.
- To disable the replay cache, add -Dsun.security.krb5.rcache=none to the Java Opts/Arguments of the targeted JVM. For example, HiveServer2 or the Sentry service.

Cloudera Manager cluster services fail to start

Symptom: One or more of the cluster services fails to start. For example, the DataNode fails to start. Error messages in the log files may display the following error messages:

```
Exception in secureMain
java.lang.ExceptionInInitializerError
at javax.crypto.KeyGenerator.nextSpi(KeyGenerator.java:324)
at javax.crypto.KeyGenerator.<init>(KeyGenerator.java:157)
...
Caused by: java.lang.SecurityException: The jurisdiction policy files are
 not signed by a trusted signer!
at javax.crypto.JarVerifier.verifyPolicySigned(JarVerifier.java:289)
at javax.crypto.JceSecurity.loadPolicies(JceSecurity.java:316)
at javax.crypto.JceSecurity.setupJurisdictionPolicies(JceSecurity.java:261)
...
```

Possible cause: This is another example of a mismatch for AES-256 encryption. Services cannot start when the version of the JCE policy file does not match the version of Java installed on a node because the cryptographic signatures for the JCE policy files cannot be verified, resulting in the message shown above.

- Check that the encryption types are matched between your KDC and krb5.conf on all hosts.

  Solution: If you are using AES-256, follow the instructions at "JCE Policy File for AES-256 Encryption" to deploy the JCE policy file on all hosts.
- Services cannot start

Solution: Download the correct JCE policy files for the version of Java you are running:

- Java 8
- Java 7
- Java 6 [Legacy information]

Download and unpack the zip file. Copy the two JAR files to the $JAVA_HOME/jre/lib/security directory on each node within the cluster.

Error Messages

Incorrect permission Java exception (java.io.IOException)

Symptom: An incorrect permission error displays when trying to run a job:

```
java.io.IOException: Incorrect permission for
/var/folders/B3/B3d2vCm4F+mmWzVPB89W6E+++TI/-Tmp-/tmpYTil84/dfs/data/data1,
expected: rwxr-xr-x, while actual: rwxrwxr-x
       at org.apache.hadoop.util.DiskChecker.checkPermission(DiskChecker.
java:107)
       at org.apache.hadoop.util.DiskChecker.mkdirsWithExistsAndPermission
Check(DiskChecker.java:144)
       at org.apache.hadoop.util.DiskChecker.checkDir(DiskChecker.java:160)
       at org.apache.hadoop.hdfs.server.datanode.DataNode.makeInstance(DataN
ode.java:1484)
       at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateData
Node(DataNode.java:1432)
       at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNo
de(DataNode.java:1408)
```

```
       at org.apache.hadoop.hdfs.MiniDFSCluster.startDataNodes(MiniDFSClu
ster.java:418)
      ...
```

Possible cause: The daemon has umask 0002 rather than 0022.

Steps to resolve: Make sure that the umask for hdfs and mapred is 0022.

MapReduce (MRv1) Errors

These error messages are associated with MapReduce only (not YARN).

Jobs won't run and cannot access files in mapred.local.dir

Symptom: The TaskTracker log contains the following error message:

```
WARN org.apache.hadoop.mapred.TaskTracker: Exception while localization java
.io.IOException: Job initialization failed (1)
```

Possible cause:

Steps to resolve:

1. Add the mapred user to the mapred and hadoop groups on all hosts.
2. Restart all TaskTrackers.

Jobs cannot run and TaskTracker cannot create local mapred directory

Symptom: The TaskTracker log contains the following error message:

```
11/08/17 14:44:06 INFO mapred.TaskController: main : user is atm
11/08/17 14:44:06 INFO mapred.TaskController: Failed to create directory /v
ar/log/hadoop/cache/mapred/mapred/local1/taskTracker/atm - No such file or d
irectory
11/08/17 14:44:06 WARN mapred.TaskTracker: Exception while localization jav
a.io.IOException: Job initialization failed (20)
        at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(Linux
TaskController.java:191)
        at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:396)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupI
nformation.java:1127)
        at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.
java:1174)
        at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.j
ava:1089)
        at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.j
ava:2257)
        at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTrack
er.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
        at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
        at org.apache.hadoop.util.Shell.run(Shell.java:182)
        at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.j
ava:375)
        at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(Linux
TaskController.java:184)
        ... 8 more
```

Possible cause: Mismatch of mapred.local.dir values specified in mapred-site.xml and taskcontroller.cfg. These values should be the same.

Steps to resolve: Verify that the setting for mapred.local.dir is the same in both mapred-site.xml and taskcontroller.cfg, and reconfigure if necessary.

Jobs cannot run and TaskTracker cannot create Hadoop logs directory

Symptom: The TaskTracker log contains an error message similar to the following:

```
11/08/17 14:48:23 INFO mapred.TaskController: Failed to create directory /ho
me/atm/src/cloudera/hadoop/build/hadoop-0.23.2-cdh3u1-SNAPSHOT/logs1/userlog
s/job_201108171441_0004 - No such file or directory
11/08/17 14:48:23 WARN mapred.TaskTracker: Exception while localization java
.io.IOException: Job initialization failed (255)
        at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(Linux
TaskController.java:191)
        at org.apache.hadoop.mapred.TaskTracker$4.run(TaskTracker.java:1199)
        at java.security.AccessController.doPrivileged(Native Method)
        at javax.security.auth.Subject.doAs(Subject.java:396)
        at org.apache.hadoop.security.UserGroupInformation.doAs(UserGroupI
nformation.java:1127)
        at org.apache.hadoop.mapred.TaskTracker.initializeJob(TaskTracker.
java:1174)
        at org.apache.hadoop.mapred.TaskTracker.localizeJob(TaskTracker.j
ava:1089)
        at org.apache.hadoop.mapred.TaskTracker.startNewTask(TaskTracker.j
ava:2257)
        at org.apache.hadoop.mapred.TaskTracker$TaskLauncher.run(TaskTrack
er.java:2221)
Caused by: org.apache.hadoop.util.Shell$ExitCodeException:
        at org.apache.hadoop.util.Shell.runCommand(Shell.java:255)
        at org.apache.hadoop.util.Shell.run(Shell.java:182)
        at org.apache.hadoop.util.Shell$ShellCommandExecutor.execute(Shell.j
ava:375)
        at org.apache.hadoop.mapred.LinuxTaskController.initializeJob(Linux
TaskController.java:184)
        ... 8 more
```

Possible cause: Misconfiguration issue.

Steps to resolve: In MRv1, the default value specified for hadoop.log.dir in mapred-site.xml is /var/log/hadoop-0.20-mapreduce. The path must be owned and be writable by the mapred user. If you change the default value specified for hadoop.log.dir, make sure the value is identical in mapred-site.xml and taskcontroller.cfg. If the values are different, the error message above is returned.

Related Information

Getting Started: Operational database cluster
Enabling JCE Policy
Kerberos Authentication Regenerate Principals
JCE 8 download
JCE 7 download
JCE 6 download
MIT Kerberos replay cache

Authentication and Kerberos Issues

Troubleshooting sections for specific issues that involve authentication and Kerberos.

Overview

Clusters that use Kerberos for authentication have several possible sources of potential issues, including:

- Failure of the Key Distribution Center (KDC)
- Missing Kerberos or OS packages or libraries
- Incorrect mapping of Kerberos REALMs for cross-realm authentication

These are just some examples, but they can prevent users and services from authenticating and can interfere with the cluster's ability to run and process workloads. The first step whenever an issue emerges is to try to isolate the source of the actual issue, by answering basic questions such as these:

- Is the issue a local issue or a global issue? That is, are all users failing to authenticate, or is the issue specific to a single user?
- Is the issue specific to a single service, or are all services problematic? and so on.

If all users and multiple services are affected—and if the cluster has not worked at all after integrating with Kerberos for authentication—step through all settings for the Kerberos configuration files, as outlined in the section below, "Auditing the Kerberos Configuration".

Note: All nodes in any given cluster configured for Kerberos must use the same configuration settings in the files that are distributed throughout the cluster.

Auditing the Kerberos Configuration

Cloudera recommends verifying the Kerberos configuration whenever issues arise, especially after initially completing the integration process.

- Verify that all /etc/hosts files conform to Cloudera Manager's installation requirements ("Cloudera Enterprise Requirements and Supported Versions").
- Verify forward and reverse name resolution for all cluster hosts and for the MIT KDC or Active Directory KDC hosts.
- Verify that all required Kerberos server and workstation packages ("Enabling Kerberos Authentication for CDH") have been installed and are the correct versions for the OS running on the host systems.
- Verify that the hadoop.security.auth_to_local property in the core-site.xml has proper mappings for all trusted Kerberos realms, including HDFS trusted realms, for all services on the cluster that use Kerberos.
- Verify your Kerberos configuration by comparing to the "Sample Kerberos Configuration Files" shown below (see "/etc/krb5.conf") and "/var/kerberos/krb5kdc/kdc.conf").
- Review the configuration of all the KDC, REALM, and domain hosts referenced in the krb5.conf and kdc.conf files. The KDC host in particular, is a common point-of-failure and you may have to begin troubleshooting there. Ensure that the REALM set in krb5.conf has the correct hostname listed for the KDC. For cross-realm authentication, see "Reviewing Service Ticket Credentials in Cross-Realm Deployments".
- Use whether the services using Kerberos are running and responding properly with kinit/klist ("User Authentication with and Without Keytab ").
- Attempt to authenticate to Cloudera Manager using cluster service credentials specific to the issue or affected service. Examine the issued credentials if you are able to successfully authenticate with the service keytab.
- Use klist to list the principals present within a service keytab to ensure each service has one.
- Enabling debugging ("Enabling Debugging for the Authentication Process") using either the command line or Cloudera Manager.

Kerberos Command-Line Tools

User Authentication with and Without Keytab

The kinit command line tool is used to authenticate a user, service, system, or device to a KDC. The most basic example is a user authenticating to Kerberos with a username (principal) and password. In the following example, the first attempt uses a wrong password, followed by a second successful attempt.

```
[alice@host1 ~]$ kinit alice@TEST.ORG.LAB
Password for alice@TEST.ORG.LAB: (wrong password)
kinit: Preauthentication failed while getting initial credentials

[alice@host1 ~]$ kinit alice@TEST.ORG.LAB
Password for alice@TEST.ORG.LAB: (correct password)
(note silent return on successful auth)
[alice@host1 ~]$ klist
Ticket cache: FILE:/tmp/krb5cc_10001
Default principal: alice@TEST.ORG.LAB

Valid starting     Expires            Service principal
03/11/14 11:55:39  03/11/14 21:54:55  krbtgt/TEST.ORG.LAB@TEST.ORG.LAB
renew until 03/18/14 11:55:39
```

Another method of authentication is using keytabs with the kinit command. You can verify whether authentication was successful by using the klist command to show the credentials issued by the KDC. The following example attempts to authenticate the hdfs service to the KDC by using the hdfs keytab file.

```
[root@host1 312-hdfs-DATANODE]# kinit -kt hdfs.keytab hdfs/host1.test.lab@TE
ST.LAB
[root@host1 312-hdfs-DATANODE]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs/host1.test.lab@TEST.LAB

Valid starting      Expires      Service principal
03/11/14 11:18:34 03/12/14 11:18:34 krbtgt/TEST.LAB@TEST.LAB
renew until 03/18/14 11:18:34
```

Enabling Debugging for Authentication Issues

Using Cloudera Manager for Debugging

To obtain additional information in the logs and facilitate troubleshooting, administrators can set debug levels for any of the services running on Cloudera Manager Server. Typically, the settings are added using the Advanced Configuration Snippet (Safety Valve) settings for the specific service, the names are specific to the service.

as for HDFS as detailed below:

1. Log in to the Cloudera Manager Admin Console.
2. Select Clusters HDFS-*n* .
3. Click the Configuration tab.
4. Search for properties specific to the different role types for which you want to enable debugging. For example, if you want to enable debugging for the HDFS NameNode, search for the NameNode Logging Threshold property and select at least DEBUG level logging.
5. Enable Kerberos debugging by using the HDFS service's Advanced Configuration Snippet. Once again, this may be different for each specific role type or service. For the HDFS NameNode, add the following properties to the HDFS Service Environment Safety Valve:

   ```
   HADOOP_JAAS_DEBUG=true
   HADOOP_OPTS="-Dsun.security.krb5.debug=true"
   ```

6. Click Save Changes.
7. Restart the HDFS service.

The output will be seen in the process logs: stdout.log and stderr.log. These can be found in the runtime path of the instance:

```
/var/run/cloudera-scm-agent/process/###-service-ROLE
```

After restarting Cloudera Manager Service, the most recent instance of the ###-service-ROLE directory will have debug logs. Use ls -ltr in the /var/run/cloudera-scm-agent/process path to determine the most current path.

Enabling Debugging for the Authentication Process

Set the following properties on the cluster to obtain debugging information from the Kerberos authentication process.

```
# export HADOOP_ROOT_LOGGER=TRACE,console;
# export HADOOP_JAAS_DEBUG=true;
# export HADOOP_OPTS="-Dsun.security.krb5.debug=true"
```

You can then use the following command to copy the console output to the user (with debugging messages), along with all output from STDOUT and STDERR to a file.

```
# hadoop fs -ls / > >(tee fsls-logfile.txt) 2>&1
```

Kerberos Credential-Generation Issues

Cloudera Manager creates accounts needed by CDH services using an internal command (Generate Credentials) that is triggered automatically by the Kerberos configuration wizard or when changes are made to the cluster that require new Kerberos principals.

After configuring the cluster for Kerberos authentication or making changes that require generation of new principals, you can verify that the command ran successfully by using the Cloudera Manager Admin Console, as follows:

1. Log in to the Cloudera Manager Admin Console. Any error messages display on the Home page, in the Status area near the top of the page. The following Status message indicates that the Generate Credentials command failed:

   ```
   Role is missing Kerberos
               keytab
   ```

2. To display the output of the command, go to the Home Status tab and click the All Recent Commands tab.

Active Directory Credential-Generation Errors

Error: ldap_sasl_interactive_bind_s: Can't contact LDAP server (-1)

Possible cause: The Domain Controller specified is incorrect or LDAPS has not been enabled for it.

Steps to resolve: Verify the configuration for Active Directory Active Directory KDC, as follows:

1. Log in to Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Select Kerberos for the Category filter.

Verify all settings. Also check that LDAPS is enabled for Active Directory.

Error: ldap_add: Insufficient access (50)

Possible cause: The Active Directory account you are using for Cloudera Manager does not have permissions to create other accounts.

Steps to resolve: Use the Delegate Control wizard to grant permission to the Cloudera Manager account to create other accounts. You can also login to Active Directory as the Cloudera Manager user to check that it can create other accounts in your Organizational Unit.

MIT Kerberos Credential-Generation Errors

Error: kadmin: Cannot resolve network address for admin server in requested realm while initializing kadmin inter face.

Possible cause: The hostname for the KDC server is incorrect.

Steps to resolve: Check the kdc field for your default realm in krb5.conf and make sure the hostname is correct.

Hadoop commands fail after enabling Kerberos security

Users need to obtain valid Kerberos tickets to interact with a secure cluster, that is, a cluster that has been configured to use Kerberos for authentication. Running any Hadoop command (such as hadoop fs -ls) will fail if you do not have a valid Kerberos ticket in your credentials cache. If you do not have a valid ticket, you will receive an error such as:

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to
 the server : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8
020 failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSExcepti
on: No valid credentials provided (Mechanism level: Failed to find any Kerbe
ros tgt)]
```

Steps to resolve: Examine the Kerberos tickets currently in your credentials cache by running the klist command. You can obtain a ticket by running the kinit command and either specifying a keytab file containing credentials, or entering the password for your principal.

Using the UserGroupInformation class to authenticate Oozie

Secured CDH services mainly use Kerberos to authenticate RPC communication. RPCs are one of the primary means of communication between nodes in a Hadoop cluster. For example, RPCs are used by the YARN NodeManager to communicate with the ResourceManager, or by the HDFS client to communicate with the NameNode.

CDH services handle Kerberos authentication by calling the UserGroupInformation (UGI) login method, loginUse rFromKeytab(), once every time the service starts up. Since Kerberos ticket expiration times are typically short, repeated logins are required to keep the application secure. Long-running CDH applications have to be implemented accordingly to accommodate these repeated logins. If an application is only going to communicate with HDFS, YARN, MRv1, and HBase, then you only need to call the UserGroupInformation.loginUserFromKeytab() method at startup, before any actual API activity occurs. The HDFS, YARN, MRv1 and HBase services' RPC clients have their own built-in mechanisms to automatically re-login when a keytab's Ticket-Granting Ticket (TGT) expires. Therefore, such applications do not need to include calls to the UGI re-login method because their RPC client layer performs the re-login task for them.

However, some applications may include other service clients that do not involve the generic Hadoop RPC framework, such as Hive or Oozie clients. Such applications must explicitly call the UserGroupInformation.getLogi nUser().checkTGTAndReloginFromKeytab() method before every attempt to connect with a Hive or Oozie client. This is because these clients do not have the logic required for automatic re-logins.

This is an example of an infinitely polling Oozie client application:

```
// App startup
UserGroupInformation.loginFromKeytab(KEYTAB_PATH, PRINCIPAL_STRING);
OozieClient client = loginUser.doAs(new PrivilegedAction<OozieClient>() {
    public OozieClient run() {
        try {
```

```
            returnnew OozieClient(OOZIE_SERVER_URI);
        } catch (Exception e) {
            e.printStackTrace();
            returnnull;
        }
    }
});

while (true && client != null) {
    // Application's long-running loop

    // Every time, complete the TGT check first
    UserGroupInformation loginUser = UserGroupInformation.getLoginUser();
    loginUser.checkTGTAndReloginFromKeytab();

    // Perform Oozie client work within the context of the login user object
    loginUser.doAs(new PrivilegedAction<Void>() {
        publicVoid run() {
            try {
                List<WorkflowJob> list = client.getJobsInfo("");
                for (WorkflowJob wfJob : list) {
                    System.out.println(wfJob.getId());
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } // End of function block
    }); // End of doAs
} // End of loop
```

Certain Java versions cannot read credentials cache

Symptom: For MIT Kerberos 1.8.1 (or higher), the following error will occur when you attempt to interact with the Hadoop cluster, even after successfully obtaining a Kerberos ticket using kinit:

```
11/01/04 12:08:12 WARN ipc.Client: Exception encountered while connecting to
 the server : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided
 (Mechanism level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:
8020 failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSExcept
ion: No valid credentials provided (Mechanism level: Failed to find any Kerb
eros tgt)]
```

Possible cause:

At release 1.8.1 of MIT Kerberos, a change ("#6206: new API for storing extra per-principal data in ccache") was made to the credentials cache format that conflicts with Oracle JDK 6 Update 26 (and earlier JDKs) (for details, see "JDK-6979329 : CCacheInputStream fails to read ticket cache files from Kerberos 1.8.1") rendering Java incapable of reading Kerberos credentials cache created by versions of MIT Kerberos 1.8.1 (or higher). Kerberos 1.8.1 is the default in Ubuntu Lucid and higher releases and Debian Squeeze and higher releases. On RHEL and CentOS, an older version of MIT Kerberos which does not have this issue, is the default.

Workaround: Use the -R (renew) option with kinit after initially obtaining credentials with kinit. This sequence causes the ticket to be renewed and credentials are cached using a format that Java can read. However, the initial ticket must be renewable.

For example:

```
$ klist
```

```
klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc_1000)
$ hadoop fs -ls
11/01/04 13:15:51 WARN ipc.Client: Exception encountered while connecting
 to the server : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided (
Mechanism level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.
2:8020 failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSExce
ption: No valid credentials provided (Mechanism level: Failed to find any Ke
rberos tgt)]
$ kinit
Password for username@REALM-NAME.EXAMPLE.COM:
$ klist
Ticket cache: FILE:/tmp/krb5cc_1000
Default principal: username@REALM-NAME.EXAMPLE.COM

Valid starting     Expires            Service principal
01/04/11 13:19:31  01/04/11 23:19:31  krbtgt/REALM-NAME.EXAMPLE.COM@REALM-
NAME.EXAMPLE.COM
 renew until 01/05/11 13:19:30
$ hadoop fs -ls
11/01/04 13:15:59 WARN ipc.Client: Exception encountered while connecting to
 the server : javax.security.sasl.SaslException:
GSS initiate failed [Caused by GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)]
Bad connection to FS. command aborted. exception: Call to nn-host/10.0.0.2:8
020 failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSExcepti
on: No valid credentials provided (Mechanism level: Failed to find any Kerbe
ros tgt)]
$ kinit -R
$ hadoop fs -ls
Found 6 items
drwx------   - user user     0 2011-01-02
          16:16 /user/user/.staging
```

Non-renewable tickets display this error message when the command

```
kinit: Ticket expired while renewing credentials
```

Resolving Cloudera Manager Service keytab Issues

Every service managed by Cloudera Manager has a keytab file that is provided at startup by the Cloudera Manager Agent. The most recent keytab files can be examined by navigating to the path, /var/run/cloudera-scm-agent/process, with an ls -ltr command.

As you can see in the example below, Cloudera Manager service directory names have the form: ###-service-ROLE. Therefore, if you are troubleshooting the HDFS service, the service directory may be called, 326-hdfs-NAMENODE.

```
[root@cehd1 ~]# cd /var/run/cloudera-scm-agent/process/
[root@cehd1 process]# ls -ltr | grep NAMENODE | tail -3
drwxr-x--x 3 hdfs         hdfs         4096 Mar  3 23:43 313-hdfs-NAMENODE
drwxr-x--x 3 hdfs         hdfs         4096 Mar  4 00:07 326-hdfs-NAMENODE
drwxr-x--x 3 hdfs         hdfs         4096 Mar  4 00:07 328-hdfs-NAMENODE-
nnRpcWait

[root@cehd1 process]# cd 326-hdfs-NAMENODE

[root@cehd1 326-hdfs-NAMENODE]# ls
cloudera_manager_agent_fencer.py              dfs_hosts_allow.txt
 hdfs.keytab                  log4j.properties             topology.py
```

```
cloudera_manager_agent_fencer_secret_key.txt  dfs_hosts_exclude.txt
hdfs-site.xml                logs
cloudera-monitor.properties                   event-filter-rules.json     h
ttp-auth-signature-secret  navigator.client.properties
core-site.xml                                 hadoop-metrics2.properties
 krb5cc_494                   topology.map
```

If you have root access to the /var/run/cloudera-scm-agent/process path, you can use any service's keytab file to log in as root or a sudo user to verify whether basic Kerberos authentication is working.

After locating a keytab file, examine its contents ("Examining Kerberos credentials with klist ") using the klist command to view the credentials stored in the file. For example, to list the credentials stored in the hdfs.keytab file:

```
[root@host1 326-hdfs-DATANODE]# klist -kt hdfs.keytab

Keytab name: WRFILE:hdfs.keytab
KVNO Timestamp         Principal
---- ----------------- --------------------------------------------------------
---
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 HTTP/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
  4 02/17/14 19:09:17 hdfs/host1.test.lab@TEST.LAB
```

Now, attempt to authenticate using the keytab file and a principal within it. In this case, we use the hdfs.keytab file with the hdfs/host1.test.lab@TEST.LAB principal. Then use the klist command without any arguments to see the current user session's credentials.

```
root@host1 312-hdfs-DATANODE]# kinit -kt hdfs.keytab hdfs/host1.test.lab@TES
T.LAB
[root@host1 312-hdfs-DATANODE]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: hdfs/host1.test.lab@TEST.LAB

Valid starting     Expires            Service principal
03/11/14 11:18:34  03/12/14 11:18:34  krbtgt/TEST.LAB@TEST.LAB
renew until 03/18/14 11:18:34
```

Note that Kerberos credentials have an expiry date and time. This means, to make sure Kerberos credentials are valid uniformly over a cluster, all hosts and clients within the cluster should be using NTP and must never drift more than 5 minutes apart from each other. Kerberos session tickets have a limited lifespan, but can be renewed (as indicated in the sample krb5.conf and kdc.conf). CDH requires renewable tickets for cluster principals. Check whether renewable tickets have been enabled by using a klist command with the -e (list key encryption types) and -f (list flags set) switches when examining Kerberos sessions and credentials.

Reviewing Service Ticket Credentials in Cross-Realm Deployments

When you examine your cluster configuration, make sure you haven't violated any of following the integration rules:

- When negotiating encryption types, follow the realm with the most specific limitations on supported encryption types.
- All realms should be known to one another through the /etc/krb5.conf file deployed on the cluster.

- When you make configuration decisions for Active Directory environments, you must evaluate the Domain Functional Level or Forrest Functional Level that is present.

Kerberos typically negotiates and uses the strongest form of encryption possible between a client and server for authentication into the realm. However, the encryption types for TGTs may sometimes end up being negotiated downward towards the weaker encryption types, which is not desirable. To investigate such issues, check the kvno of the cross-realm trust principal (krbtgt) as described in the following steps. Replace CLUSTER.REALM and AD.REALM (or MIT.REALM) with the appropriate values for your configured realm. This scenario assumes cross-realm authentication with Active Directory.

- Once trust has been configured (see sample files in previous section), kinit as a system user by authenticating to the AD Kerberos realm.
- From the command line, perform a kvno check of the local and cross-realm krbtgt entry. The local representation of this special REALM service principal is in the form, krbtgt/CLUSTER.REALM@CLUSTER.REALM. The cross-realm principal is named after the trusted realm in the form, krbtgt/AD.REALM.
- Failure of the kvno check indicates incorrect cross-realm trust configuration. Review encryption types again, looking for incompatibilities or unsupported encryption types configured between realms.

Sample Kerberos Configuration Files

This section contains several example Kerberos configuration files.

/etc/krb5.conf

The /etc/krb5.conf file is the configuration a client uses to access a realm through its configured KDC. The krb5.conf maps the realm to the available servers supporting those realms. It also defines the host-specific configuration rules for how tickets are requested and granted.

```
[logging]
 default = FILE:/var/log/krb5libs.log
 kdc = FILE:/var/log/krb5kdc.log
 admin_server = FILE:/var/log/kadmind.log

[libdefaults]
 default_realm = EXAMPLE.COM
 dns_lookup_realm = false
 dns_lookup_kdc = false
 ticket_lifetime = 24h
 renew_lifetime = 7d
 forwardable = true
# udp_preference_limit = 1
# set udp_preference_limit = 1 when TCP only should be
# used. Consider using in complex network environments when
# troubleshooting or when dealing with inconsistent
# client behavior or GSS (63) messages.
# uncomment the following if AD cross realm auth is ONLY providing DES encr
ypted tickets
# allow-weak-crypto = true

[realms]
 AD-REALM.EXAMPLE.COM = {
  kdc = AD1.ad-realm.example.com:88
  kdc = AD2.ad-realm.example.com:88
  admin_server = AD1.ad-realm.example.com:749
  admin_server = AD2.ad-realm.example.com:749
  default_domain = ad-realm.example.com
 }
 EXAMPLE.COM = {
  kdc = kdc1.example.com:88
  admin_server = kdc1.example.com:749
  default_domain = example.com
```

```
 }

# The domain_realm is critical for mapping your host domain names to the k
erberos realms
# that are servicing them. Make sure the lowercase left hand portion indi
cates any domains or subdomains
# that will be related to the kerberos REALM on the right hand side of the e
xpression. REALMs will
# always be UPPERCASE. For example, if your actual DNS domain was test.com b
ut your kerberos REALM is
# EXAMPLE.COM then you would have,

[domain_realm]
test.com = EXAMPLE.COM
#AD domains and realms are usually the same
ad-domain.example.com = AD-REALM.EXAMPLE.COM
ad-realm.example.com = AD-REALM.EXAMPLE.COM
```

Sample Kerberos Configuration Files

/var/kerberos/krb5kdc/kdc.conf

The kdc.conf file only needs to be configured on the actual cluster-dedicated KDC, and should be located at /var/ker beros/krb5kdc. Only primary and secondary KDCs need access to this configuration file. The contents of this file establish the configuration rules which are enforced for all client hosts in the REALM.

```
[kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88

[realms]
  EXAMPLE.COM = {
  #master_key_type = aes256-cts
  max_renewable_life = 7d 0h 0m 0s
  acl_file = /var/kerberos/krb5kdc/kadm5.acl
  dict_file = /usr/share/dict/words
  admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
# note that aes256 is ONLY supported in Active Directory in a domain / fo
rrest operating at a 2008 or greater functional level.
# aes256 requires that you download and deploy the JCE Policy files for y
our JDK release level to provide
# strong java encryption extension levels like AES256. Make sure to match b
ased on the encryption configured within AD for
# cross realm auth, note that RC4 = arcfour when comparing windows and linux
 enctypes
  supported_enctypes = aes256-cts:normal aes128-cts:normal arcfour-hmac:no
rmal
  default_principal_flags = +renewable, +forwardable
 }

[logging]
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmin.log
```

kadm5.acl

```
*/admin@HADOOP.COM *
cloudera-scm@HADOOP.COM * flume/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hbase/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hdfs/*@HADOOP.COM
```

```
cloudera-scm@HADOOP.COM * hive/*@HADOOP.COM
cloudera-scm@HADOOP.COM * httpfs/*@HADOOP.COM
cloudera-scm@HADOOP.COM * HTTP/*@HADOOP.COM
cloudera-scm@HADOOP.COM * hue/*@HADOOP.COM
cloudera-scm@HADOOP.COM * impala/*@HADOOP.COM
cloudera-scm@HADOOP.COM * mapred/*@HADOOP.COM
cloudera-scm@HADOOP.COM * oozie/*@HADOOP.COM
cloudera-scm@HADOOP.COM * solr/*@HADOOP.COM
cloudera-scm@HADOOP.COM * sqoop/*@HADOOP.COM
cloudera-scm@HADOOP.COM * yarn/*@HADOOP.COM
cloudera-scm@HADOOP.COM * zookeeper/*@HADOOP.COM
```

Related Information

Enabling Kerberos Authentication

#6206: new API for storing extra per-principal data in ccache

JDK-6979329 : CCacheInputStream fails to read ticket cache files from Kerberos 1.8.1

HDFS Encryption Issues

The following are possible workarounds for issues that may arise when encrypting HDFS directories and files. HDFS encryption is sometimes referred to in the documentation as HDFS Transparent Encryption or as HDFS Data at Rest Encryption.

KMS server jute buffer exception

Description: You see the following error when the KMS (for example, as a ZooKeeper client) jute buffer size is insufficient to hold all the tokens:

```
2017-01-31 21:23:56,416 WARN org.apache.zookeeper.ClientCnxn: Session 0x259f
5fb3c1000fb for server example.cloudera.com/10.172.0.1:2181, unexpected erro
r, closing socket connection and attempting reconnect
java.io.IOException: Packet len4196356 is out of range!
```

Solution: Increase the jute buffer size and restart the KMS. In Cloudera Manager, go to the KMS Configuration page, and in the Additional Java Configuration Options for KMS (kms_java_opts) field, enter -Djute.maxbuffer=<number_of_bytes>. Restart the KMS.

Retrieval of encryption keys fails

Description: You see the following error when trying to list encryption keys

```
user1@example-sles-4:~> hadoop key list
Cannot list keys for KeyProvider: KMSClientProvider[https: //example-sles-2.
example.com:16000/kms/v1/]: Retrieval of all keys failed.
```

Solution: Make sure your truststore has been updated with the relevant certificate(s), such as the Key Trustee server certificate.

DistCp between unencrypted and encrypted locations fails

Description: By default, DistCp compares checksums provided by the filesystem to verify that data was successfully copied to the destination. However, when copying between unencrypted and encrypted locations, the filesystem checksums will not match since the underlying block data is different.

Solution: Specify the -skipcrccheck and -update distcp flags to avoid verifying checksums.

NameNode - KMS communication fails after long periods of inactivity

Description: Encrypted files and encryption zones cannot be created if a long period of time (by default, 20 hours) has passed since the last time the KMS and NameNode communicated.

Solution: Upgrading your cluster to CDH 6 will fix this problem. For instructions, see "Upgrading the CDH Cluster".

HDFS Trash Behaviour with Transparent Encryption Enabled

The Hadoop trash feature helps prevent accidental deletion of files and directories. When you delete a file in HDFS, the file is not immediately expelled from HDFS. Deleted files are first moved to the /user/<username>/.Trash/Current directory, with their original filesystem path being preserved. After a user-configurable period of time (fs.trash.interval), a process known as trash checkpointing renames the Current directory to the current timestamp, that is, /user/<username>/.Trash/<timestamp>. The checkpointing process also checks the rest of the .Trash directory for any existing timestamp directories and removes them from HDFS permanently. You can restore files and directories in the trash simply by moving them to a location outside the .Trash directory.

Trash Behaviour with HDFS Transparent Encryption Enabled

Starting with CDH 5.7, you can delete files or directories that are part of an HDFS encryption zone. As is evident from the procedure described above, moving and renaming files or directories is an important part of trash handling in HDFS. However, currently HDFS transparent encryption only supports renames within an encryption zone. To accommodate this, HDFS creates a local .Trash directory every time a new encryption zone is created. For example, when you create an encryption zone, enc_zone, HDFS will also create the /enc_zone/.Trash/ subdirectory. Files deleted from enc_zone are moved to /enc_zone/.Trash/<username>/Current/. After the checkpoint, the Current directory is renamed to the current timestamp, /enc_zone/.Trash/<username>/<timestamp>.

If you delete the entire encryption zone, it will be moved to the .Trash directory under the user's home directory, /users/<username>/.Trash/Current/enc_zone. Trash checkpointing will occur only after the entire zone has been moved to /users/<username>/.Trash. However, if the user's home directory is already part of an encryption zone, then attempting to delete an encryption zone will fail because you cannot move or rename directories across encryption zones.

Related Information

Upgrading the CDH Cluster

Key Trustee KMS Encryption Issues

The following errors and conditions are related to the Key Trustee KMS, and includes possible workarounds for issues that may arise when using Key Trustee KMS.

Key Trustee KMS Fails to Restart After Upgrade (HA Only)

Description: You may see the following error after you attempt to restart a Key Trustee KMS HA host after an upgrade:

```
java.io.IOException: Unable to verify private key match between KMS hosts. V
erify private key files have been synced between all KMS hosts. Aborting to
prevent data inconsistency.
```

Solution: If you have failed to synchronize private keys between Key Trustee KMS hosts, they may be in a state where keys are intermittently inaccessible, depending on which Key Trustee KMS host a client interacts with, because cryptographic key material encrypted by one Key Trustee KMS host cannot be decrypted by another. If you are already running multiple Key Trustee KMS hosts with different private keys, immediately back up (see "Backing Up and Restoring Key Trustee Server and Clients") all Key Trustee KMS hosts, and contact Cloudera Support for assistance correcting the issue.

For information about how to check if you are using different private keys, see "Upgrading Key Trustee KMS".

Key Trustee KMS Fails Upon First Run (HA Only)

Description: You may see the following error upon the first run after adding a new Key Trustee KMS HA service:

```
java.io.IOException: Unable to verify private key match between KMS hosts. V
erify private key files have been synced between all KMS hosts. Aborting to
prevent data inconsistency.
```

Solution: See "Upgrading Key Trustee KMS" for guidance on synchronization and validation of private keys.

Cloudera recommends following security best practices and transferring the private key using offline media, such as a removable USB drive. For convenience (for example, in a development or testing environment where maximum security is not required), you can copy the private key over the network by running the rsynccommand on the original Key Trustee KMS host:

```
rsync -zav /var/lib/kms-keytrustee/keytrustee/.keytrustee root@ktkms02.examp
le.com:/var/lib/kms-keytrustee/keytrustee/.
```

Replace ktkms02.example.com with the host name of the Key Trustee KMS host that you are adding.

Note: Execute the rsync command only upon the first run of Key Trustee KMS; do not enter this command if the keys have already been created and data is encrypted.

Key Trustee KMS Fails to Start Because ZooKeeper is Not Running

Description: You may see the following error after you attempt to restart a Key Trustee KMS for the first time:

```
java.lang.Exception: Could not establish connection to ZooKeeper to verify K
MS host private key consistency. Verify private key files have been synced b
etween all KMS hosts. Aborting to prevent data inconsistency.
```

Solution: ZooKeeper is used to communicate with hosts and is also the storage location of private key data, and therefore must be running upon the first restart or running of the GPG validation check, which compares private keys amongst Key Trustee KMS hosts to help prevent a "split brain" scenario (when private keys are not synchronized between hosts). To ensure the GPG validation check can run, start ZooKeeper, and then restart the Key Trustee KMS.

Related Information

Upgrading Key Trustee KMS

Backing Up and Restoring Key Trustee Server and Clients

Troubleshooting TLS/SSL Issues in Cloudera Manager

To diagnose and resolve issues related to TLS/SSL configuration, verify configuration tasks appropriate for the cluster by verifying the steps in "Manually Configuring TLS Encryption for Cloudera Manager".

After checking your settings and finding no obvious misconfiguration, try some of the troubleshooting steps below.

Test Connectivity with OpenSSL

From the host that has connectivity issues, run openssl as shown below. You can also check that the certificate used by the host is recognized by a trusted CA during the TLS/SSL negotiation.

To check the connection:

```
openssl s_client -connect [host.fqdn.name]:[port]
```

For example:

```
openssl s_client -connect test1.sec.cloudera.com:7183
```

A return code 0 means openssl was able to follow trust server chain of trust through its library of trusted public CAs.

For certificates signed by your organization's internal CA or self-signed certificates, you may need to add the certificate to the truststore using the openssl command. Use the -CAfile option to specify the path to the root CA so openssl can verify the self-signed or internal-CA-signed certificate as follows:

```
$ openssl s_client -connect test1.sec.cloudera.com:7183 -CAfile \
/opt/cloudera/security/CAcerts/RootCA.pem
```

Only the Root CA certificate is needed to establish trust for this test. The result from the command is successful when you see the return code 0 as follows:

```
...
 Verify return code: 0 (ok)
---
```

By default, Cloudera Manager Server writes logs to the /etc/cloudera-scm-server/cloudera-scm-server.log file on startup. Successful server start-up using the certificate looks similar to the following log example:

```
2014-10-06 21:33:47,515 INFO WebServerImpl:org.mortbay.log: jetty-6.1.26.clo
udera.2
2014-10-06 21:33:47,572 INFO WebServerImpl:org.mortbay.log: Started SslSelec
tChannelConnector@0.0.0.0:7183
2014-10-06 21:33:47,573 INFO WebServerImpl:org.mortbay.log: Started Selec
tChannelConnector@0.0.0.0:7180
2014-10-06 21:33:47,573 INFO WebServerImpl:com.cloudera.server.cmf.WebSer
verImpl: Started Jetty server.
```

Upload Diagnostic Bundles to Cloudera Support

By default, Cloudera Manager uploads diagnostic bundles over HTTPS to the Cloudera Support server at cops .cloudera.com. However, the upload can fail if the Cloudera Manager truststore cannot verify the authenticity of the Cloudera Support server certificate, and that verification process can fail due to Cloudera Manager truststore configuration issues.

To ensure the Cloudera Manager Server truststore contains the public CAs needed to verify Cloudera Support's certificate, you can explicitly establish trust by importing Cloudera Support's certificate into Cloudera Manager's truststore (see "Importing Cloudera Support's Certificate into the Cloudera Manager Server Truststore".

Note: Cloudera Support servers use certificates signed by a commercial CA, so this step is typically not needed, unless the default truststore has been altered. Before downloading or adding any certificates, test the connection (see "Test Connectivity with OpenSSL") and verify that the certificate is the source of the connection issue.

Importing Cloudera Support's Certificate into the Cloudera Manager Server Truststore

To obtain Cloudera's public key certificate from the Cloudera Support server:

```
openssl s_client -connect cops.cloudera.com:443 | openssl x509 -text -out /p
ath/to/cloudera-cert.pem
```

To import this certificate into the Cloudera Manager truststore (use paths for your own system):

```
keytool -import -keystore /path/to/cm/truststore.jks -file /path/to/cloudera
-cert.pem
```

After importing the certificate, confirm that Cloudera Manager is configured for this truststore file, as detailed in "Configuring Cloudera Manager Truststore Properties".

Note: Alternatively, you can use the default Java truststore for your Cloudera Manager cluster deployment, as described in "Manually Configuring TLS Encryption for Cloudera Manager".

Configuring Cloudera Manager Truststore Properties

After installing the Cloudera Support server certificate into the Cloudera Manager truststore, you must configure Cloudera Manager to use the truststore, as follows:

1. Log into the Cloudera Manager Admin Console.
2. Select Administration Settings .
3. Click the Security category.
4. Enter the path to the truststore and the password (if necessary):

Setting	Description
Cloudera Manager TLS/SSL Certificate Trust Store File	Enter the complete Cloudera Manager Server host filesystem path to the truststore (the trust .jks). Cloudera Manager Server invokes JVM with -Djavax.net.ssl.trustStore to access the specified truststore.
Cloudera Manager TLS/SSL Certificate Trust Store Password	Specify the password (if there is one) for the truststore file. Password is not required to access the truststore, so you can typically leave this field blank. Cloudera Manager Server invokes JVM with -Djavax.net.ssl.trustStore.password if this field has an entry.

5. Click Save Changes to save the settings.

Note: See Oracle's "JSSE Reference Guide" for more information about the JSSE trust mechanism.

Related Information

Auto TLS

JSSE Reference Guide

YARN, MRv1, and Linux OS Security

Several subsystems are fundamental to Hadoop clusters, specifically, the Jsvc, TaskController, and Container Executor Programs, documented below.

MRv1 and YARN: The jsvc Program

The jsvc is part of the Hadoop Apache Commons libraries designed to make Java applications run better on Linux. The jsvcprogram is part of the bigtop-jsvc package and installed in /usr/lib/bigtop-utils/jsvc or /usr/libexec/bigtop-utils/jsvc depending on version of Linux.

In particular, jsvc is used to start the DataNode listening on low port numbers. Its entry point is the SecureDataNodeStarter class, which implements the Daemon interface that jsvc expects. jsvc is run as root, and calls the SecureDataNodeStarter.init(...) method while running as root. Once the SecureDataNodeStarter class has finished initializing, jsvc sets the effective UID to be the hdfs user, and then calls SecureDataNodeStarter.start(...). SecureDataNodeStarter then calls the regular DataNode entry point, passing in a reference to the privileged resources it previously obtained.

MRv1 Only: The Linux TaskController

A setuid binary called task-controller is part of the hadoop-0.20-mapreduce package and is installed in either /usr/lib/hadoop-0.20-mapreduce/sbin/Linux-amd64-64/task-controller or /usr/lib/hadoop-0.20-mapreduce/sbin/Linux-i386-32/task-controller.

This task-controller program, which is used on MRv1 only, allows the TaskTracker to run tasks under the Unix account of the user who submitted the job in the first place. It is a setuid binary that must have a very specific set of permissions and ownership to function correctly. In particular, it must:

1. Be owned by root
2. Be owned by a group that contains only the user running the MapReduce daemons
3. Be setuid
4. Be group readable and executable

This corresponds to the ownership root:mapred and the permissions 4754.

Here is the output of ls on a correctly-configured Task-controller:

```
-rwsr-xr-- 1 root mapred 30888 Mar 18 13:03 task-controller
```

The TaskTracker will check for this configuration on start up, and fail to start if the Task-controller is not configured correctly.

YARN Only: The Linux Container Executor

A setuid binary called container-executor is part of the hadoop-yarn package and is installed in /usr/lib/hadoop-yarn/bin/container-executor.

This container-executor program, which is used on YARN only and supported on GNU/Linux only, runs the containers as the user who submitted the application. It requires all user accounts to be created on the cluster hosts where the containers are launched. It uses a setuid executable that is included in the Hadoop distribution. The NodeManager uses this executable to launch and kill containers. The setuid executable switches to the user who has submitted the application and launches or kills the containers. For maximum security, this executor sets up restricted permissions and user/group ownership of local files and directories used by the containers such as the shared objects, jars, intermediate files, and log files. As a result, only the application owner and NodeManager can access any of the local files/directories including those localized as part of the distributed cache.

Parcel Deployments

In a parcel deployment the container-executor file is located inside the parcel at /opt/cloudera/parcels/CDH/lib/hadoop-yarn/bin/container-executor. For the /usr/lib mount point, setuid should not be a problem. However, the parcel could easily be located on a different mount point. If you are using a parcel, make sure the mount point for the parcel directory is without the nosuid option.

The container-executor program must have a very specific set of permissions and ownership to function correctly. In particular, it must:

1. Be owned by root
2. Be owned by a group that contains only the user running the YARN daemons
3. Be setuid
4. Be group readable and executable. This corresponds to the ownership root:yarn and the permissions 6050.

```
---Sr-s--- 1 root yarn 91886 2012-04-01 19:54 container-executor
```

Important: Configuration changes to the Linux container executor could result in local NodeManager directories (such as usercache) being left with incorrect permissions. To avoid this, when making changes using either Cloudera Manager or the command line, first manually remove the existing NodeManager local directories from all configured local directories (yarn.nodemanager.local-dirs), and let the NodeManager recreate the directory structure.

Troubleshooting

When you set up a secure cluster for the first time and debug problems with it, the task-controller or container-executor may encounter errors. These programs communicate these errors to the TaskTracker or NodeManager daemon via numeric error codes that appear in the TaskTracker or NodeManager logs respectively (/var/log/hadoop-mapreduce

or /var/log/hadoop-yarn). The following sections list the possible numeric error codes with descriptions of what they mean:

- "TaskController Error Codes (MRv1)"
- "ContainerExecutor Error Codes (YARN)"

Related Information

Apache Commons
TaskController Error Codes (MRv1)
ContainerExecutor Error Codes (YARN)

TaskController Error Codes (MRv1)

This table shows some of the error codes and messages generated by TaskController in MapReduce (MRv1).

Numeric Code	Name	Description
1	INVALID_ARGUMENT_NUMBER	• Incorrect number of arguments provided for the given task-controller command • Failure to initialize the job localizer
2	INVALID_USER_NAME	The user passed to the task-controller does not exist.
3	INVALID_COMMAND_PROVIDED	The task-controller does not recognize the command it was asked to execute.
4	SUPER_USER_NOT_ALLOWED_TO_RUN_TASKS	The user passed to the task-controller was the super user.
5	INVALID_TT_ROOT	The passed TaskTracker root does not match the configured TaskTracker root (mapred.local.dir), or does not exist.
6	SETUID_OPER_FAILED	Either could not read the local groups database, or could not set UID or GID
7	UNABLE_TO_EXECUTE_TASK_SCRIPT	The task-controller could not execute the task launcher script.
8	UNABLE_TO_KILL_TASK	The task-controller could not kill the task it was passed.
9	INVALID_TASK_PID	The PID passed to the task-controller was negative or 0.
10	ERROR_RESOLVING_FILE_PATH	The task-controller could not resolve the path of the task launcher script file.
11	RELATIVE_PATH_COMPONENTS_IN_FILE_PATH	The path to the task launcher script file contains relative components (for example, "..").
12	UNABLE_TO_STAT_FILE	The task-controller did not have permission to stat a file it needed to check the ownership of.
13	FILE_NOT_OWNED_BY_TASKTRACKER	A file which the task-controller must change the ownership of has the wrong the ownership.
14	PREPARE_ATTEMPT_DIRECTORIES_FAILED	The mapred.local.dir is not configured, could not be read by the task-controller, or could not have its ownership secured.
15	INITIALIZE_JOB_FAILED	The task-controller could not get, stat, or secure the job directory or job working working directory.
16	PREPARE_TASK_LOGS_FAILED	The task-controller could not find or could not change the ownership of the task log directory to the passed user.
17	INVALID_TT_LOG_DIR	The hadoop.log.dir is not configured.

Numeric Code	Name	Description
18	OUT_OF_MEMORY	The task-controller could not determine the job directory path or the task launcher script path.
19	INITIALIZE_DISTCACHEFILE_FAILED	Could not get a unique value for, stat, or the local distributed cache directory.
20	INITIALIZE_USER_FAILED	Could not get, stat, or secure the per-user task tracker directory.
21	UNABLE_TO_BUILD_PATH	The task-controller could not concatenate two paths, most likely because it ran out of memory.
22	INVALID_TASKCONTROLLER_PERMISSIONS	The task-controller binary does not have the correct permissions set. See "YARN, MRv1, and Linux OS Security".
23	PREPARE_JOB_LOGS_FAILED	The task-controller could not find or could not change the ownership of the job log directory to the passed user.
24	INVALID_CONFIG_FILE	The taskcontroller.cfg file is missing, malformed, or has incorrect permissions.
255	Unknown Error	Several possible causes for this error, including: • User accounts on the cluster with an ID less than the value specified for the min.user.id property in the taskcontroller.cfg file. Default value of 1000 is good for Ubuntu but may not be valid for other OSs. To set the min.user.id in the taskcontroller.cfg file, see "Step 7: Prepare the cluster for each user". • Jobs do not run and the TaskTracker is unable to create a Hadoop logs directory (see "Troubleshooting Error Messages>Jobs cannot run and TaskTracker cannot create Hadoop logs directory"). • May result from previous errors. Check older entries in the log file for possibilities.

Related Information

YARN, MRv1, and Linux OS Security
Error Messages and Various Failures
Prepare Kerberos Cluster

ContainerExecutor Error Codes (YARN)

The codes in the table apply to the container-executor in YARN, but are used by the LinuxContainerExecutor only.

Numeric Code	Name	Description
1	INVALID_ARGUMENT_NUMBER	• Incorrect number of arguments provided for the given container-executor command • Failure to initialize the container localizer
2	INVALID_USER_NAME	The user passed to the container-executor does not exist.
3	INVALID_COMMAND_PROVIDED	The container-executor does not recognize the command it was asked to run.
5	INVALID_NM_ROOT	The passed NodeManager root does not match the configured NodeManager root (yarn.nodemanager.local-dirs), or does not exist.

Numeric Code	Name	Description
6	SETUID_OPER_FAILED	Either could not read the local groups database, or could not set UID or GID
7	UNABLE_TO_EXECUTE_CONTAINER_SCRIPT	The container-executor could not run the container launcher script.
8	UNABLE_TO_SIGNAL_CONTAINER	The container-executor could not signal the container it was passed.
9	INVALID_CONTAINER_PID	The PID passed to the container-executor was negative or 0.
18	OUT_OF_MEMORY	The container-executor couldn't allocate enough memory while reading the container-executor.cfg file, or while getting the paths for the container launcher script or credentials files.
20	INITIALIZE_USER_FAILED	Couldn't get, stat, or secure the per-user NodeManager directory.
21	UNABLE_TO_BUILD_PATH	The container-executor couldn't concatenate two paths, most likely because it ran out of memory.
22	INVALID_CONTAINER_EXEC_PERMISSIONS	The container-executor binary does not have the correct permissions set. See "YARN, MRv1, and Linux OS Security".
24	INVALID_CONFIG_FILE	The container-executor.cfg file is missing, malformed, or has incorrect permissions.
25	SETSID_OPER_FAILED	Could not set the session ID of the forked container.
26	WRITE_PIDFILE_FAILED	Failed to write the value of the PID of the launched container to the PID file of the container.
255	Unknown Error	This error has several possible causes. Some common causes are: • User accounts on your cluster have a user ID less than the value specified for the min.user.id property in the container-executor.cfg file. The default value is 1000 which is appropriate on Ubuntu systems, but may not be valid for your operating system. For information about setting min.user.id in the container-executor.cfg file, see "Step 7: Prepare the cluster for each user". • This error is often caused by previous errors; look earlier in the log file for possible causes.

The following exit status codes apply to all containers in YARN. These exit status codes are part of the YARN framework and are in addition to application specific exit codes that can be set:

Numeric Code	Name	Description
0	SUCCESS	Container has finished succesfully.
-1000	INVALID	Initial value of the container exit code. A container that does not have a COMPLETED state will always return this status.
-100	ABORTED	Containers killed by the framework, either due to being released by the application or being 'lost' due to node failures, for example.

Numeric Code	Name	Description
-101	DISKS_FAILED	Container exited due to local disks issues in the NodeManager node. This occurs when the number of good nodemanager-local-directories or nodemanager-log-directories drops below the health threshold.
-102	PREEMPTED	Containers preempted by the framework. This does not count towards a container failure in most applications.
-103	KILLED_EXCEEDED_VMEM	Container terminated because of exceeding allocated virtual memory limit.
-104	KILLED_EXCEEDED_PMEM	Container terminated because of exceeding allocated physical memory limit.
-105	KILLED_BY_APPMASTER	Container was terminated on request of the application master.
-106	KILLED_BY_RESOURCEMANAGER	Container was terminated by the resource manager.
-107	KILLED_AFTER_APP_COMPLETION	Container was terminated after the application finished.

Related Information

YARN, MRv1, and Linux OS Security
Prepare Kerberos Cluster