# **基本描述**

**Kerberos**使用Needha-Schroeder协议作为它的基础。它使用了一个由两个独立的逻辑部分：认证服务器和票据授权服务器组成的"可信赖的第三方"，术语称为密钥分发中心（KDC）。**Kerberos**工作在用于证明用户身份的"票据"的基础上。

KDC持有一个密钥数据库；每个网络实体——无论是客户还是服务器——共享了一套只有他自己和KDC知道的密钥。密钥的内容用于证明实体的身份。对于两个实体间的通信，KDC产生一个会话密钥，用来加密他们之间的交互信息。

# **协议内容**

协议的安全主要依赖于参加者对时间的松散同步和短周期的叫做Kerberos票据的认证声明。 下面是对这个协议的一个简化描述，将使用以下缩写：

* AS（Authentication Server）= 认证服务器
* TGT（Ticket Granting Ticket）= 票据授权票据，票据的票据
* TGS（Ticket Granting Server）= 票据授权服务器
* SS（Service Server）= 服务器  
  其在网络通讯协定中属于显示层。

简单地说，用户先用共享密钥从某认证服务器得到一个身份证明。随后，用户使用这个身份证明与SS通信，而不使用共享密钥。

# **具体流程**

首先，用户使用客户机（用户自己的机器）上的程序登录：

* 用户输入用户ID和密码到客户机。

客户机程序运行一个单向函数（大多数为杂凑）把密码转换成密钥，这个就是客户机（用户）的"用户密钥"（K\_client）。受信任的AS通过某些安全的途径也获取了与此密钥相同的密钥。

* 随后，客户机认证（客户机从AS获取票据的票据（TGT））：
* 客户机向AS发送1条消息（注意：用户不向AS发送密钥（K\_client），也不发送密码）：
  + 包含用户ID的明文消息，例如"用户Sunny想请求服务"（Sunny是用户ID）
* AS检查用户ID有效性，而后返回2条消息：
  + 消息A：用户密钥（K\_client）加密后的"客户机-TGS会话密钥"(K\_TGS-session)（会话密钥用在将来客户机与TGS的通信（会话）上）
  + 消息B：TGS密钥（K\_TGS）加密后的"票据授权票据"(TGT)（TGT包括：客户机-TGS会话密钥（K\_TGS-session），用户ID，用户网址，TGT有效期）

客户机用自己的密钥（K\_client）解密A，得到客户机-TGS会话密钥（K\_TGS-session）。（注意：客户机不能解密消息B，因为B是用TGS密钥（K\_TGS）加密的）。

* 然后，服务授权（客户机从TGS获取票据（T））：
  + 客户机向TGS发送以下2条消息：
    - 消息c：即消息B（K\_TGS加密后的TGT），和想获取的服务的服务ID（注意：不是用户ID）
    - 消息d：客户机-TGS会话密钥（K\_TGS-session）加密后的"认证符"（认证符包括：用户ID，时间戳）
  + TGS用自己的密钥（K\_TGS）解密c中的B得到TGT，从而得到AS提供的客户机-TGS会话密钥（K\_TGS-session）。再用这个会话密钥解密d得到用户ID（认证），而后返回2条消息：
    - 消息E：服务器密钥（K\_SS）加密后的"客户机-服务器票据"(T)（T包括：客户机-SS会话密钥（K\_SS-session），用户ID，用户网址，T有效期）
    - 消息F：客户机-TGS会话密钥（K\_TGS-session）加密后的"客户机-SS会话密钥"(K\_SS\_session)
  + 客户机用客户机-TGS会话密钥（K\_TGS-session）解密F，得到客户机-SS会话密钥（K\_SS\_session）。（注意：客户机不能解密消息E，因为E是用SS密钥（K\_SS）加密的）。

最后，服务请求（客户机从SS获取服务）：

* + 客户机向SS发出2条消息：
  + 消息e：即消息E
  + 消息g：客户机-服务器会话密钥（K\_SS\_session）加密后的"新认证符"（新认证符包括：用户ID，时间戳）
  + SS用自己的密钥（K\_SS）解密e/E得到T，从而得到TGS提供的客户机-服务器会话密钥（K\_SS\_session）。再用这个会话密钥解密g得到用户ID（认证），而后返回1条消息（确认函：确证身份真实，乐于提供服务）：
    - 消息H：客户机-服务器会话密钥（K\_SS\_session）加密后的"新时间戳"（新时间戳是：客户机发送的时间戳加1）
  + 客户机用客户机-服务器会话密钥（K\_SS\_session）解密H，得到新时间戳。
  + 客户机检查时间戳被正确地更新，则客户机可以信赖服务器，并向服务器（SS）发送服务请求。

服务器（SS）提供服务。

# **缺陷**

* 失败于单点：它需要中心服务器的持续响应。当Kerberos服务结束前，没有人可以连接到服务器。这个缺陷可以通过使用复合Kerberos服务器和缺陷认证机制弥补。
* **Kerberos**要求参与通信的主机的时钟同步。票据具有一定有效期，因此，如果主机的时钟与Kerberos服务器的时钟不同步，认证会失败。默认设置要求时钟的时间相差不超过10分钟。在实践中，通常用网络时间协议后台程序来保持主机时钟同步。
* 管理协议并没有标准化，在服务器实现工具中有一些差别。RFC 3244描述了密码更改。
* 因为所有用户使用的密钥都存储于中心服务器中，危及服务器的安全的行为将危及所有用户的密钥。

一个危险客户机将危及用户密码。

# **常见问题**

相关链接：<https://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH4-Security-Guide/cdh4sg_topic_17.html>  
**Problem 1: Running any Hadoop command fails after enabling security.**  
**Description:**  
A user must have a valid Kerberos ticket in order to interact with a secure Hadoop cluster. 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/10.0.0.2:8020 failed on local exception: java.io.IOException:

javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]

**Solution:**  
You can 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.

**Problem 2: Java is unable to read the Kerberos credentials cache created by versions of MIT Kerberos 1.8.1 or higher.**

**Description:**  
If you are running 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 GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]

Because of a change 1 in the format in which MIT Kerberos writes its credentials cache, there is a bug 2 in the Oracle JDK 6 Update 26 and earlier that causes Java to be unable to read the 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 later releases and Debian Squeeze and later releases. (On RHEL and CentOS, an older version of MIT Kerberos which does not have this issue, is the default.)  
**Footnotes:** 1 MIT Kerberos change: <http://krbdev.mit.edu/rt/Ticket/Display.html?id=6206> 2 Report of bug in Oracle JDK 6 Update 26 and earlier: <http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6979329>

**Solution:**  
If you encounter this problem, you can work around it by running kinit -R after running kinit initially to obtain credentials. Doing so will cause the ticket to be renewed, and the credentials cache rewritten in a format which Java can read. To illustrate this:

$ klist

klist: No credentials cache found (ticket cache FILE:/tmp/krb5cc\_1000)

$ hadoop fs -ls11/01/04 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 GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]

$ kinitPassword for atm@YOUR-REALM.COM:

$ klist

Ticket cache: FILE:/tmp/krb5cc\_1000Default principal: atm@YOUR-REALM.COM

Valid starting Expires Service principal01/04/11 13:19:31 01/04/11 23:19:31 krbtgt/YOUR-REALM.COM@YOUR-REALM.COM

renew until 01/05/11 13:19:30

$ hadoop fs -ls11/01/04 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/10.0.0.2:8020 failed on local exception: java.io.IOException:

javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]

$ kinit -R

$ hadoop fs -lsFound 6 items

drwx------ - atm atm 0 2011-01-02 16:16 /user/atm/.staging

**Note:**  
This workaround for Problem 2 requires the initial ticket to be renewable. Note that whether or not you can obtain renewable tickets is dependent upon a KDC-wide setting, as well as a per-principal setting for both the principal in question and the Ticket Granting Ticket (TGT) service principal for the realm. A non-renewable ticket will have the same values for its "valid starting" and "renew until" times. If the initial ticket is not renewable, the following error message is displayed when attempting to renew the ticket:

kinit: Ticket expired while renewing credentials

**Problem 3: java.io.IOException: Incorrect permission  
Description:**  
An error such as the following example is displayed if the user running one of the Hadoop daemons has a umask of 0002, instead of 0022:

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.mkdirsWithExistsAndPermissionCheck(DiskChecker.java:144)

at org.apache.hadoop.util.DiskChecker.checkDir(DiskChecker.java:160)

at org.apache.hadoop.hdfs.server.datanode.DataNode.makeInstance(DataNode.java:1484)

at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1432)

at org.apache.hadoop.hdfs.server.datanode.DataNode.instantiateDataNode(DataNode.java:1408)

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

at org.apache.hadoop.hdfs.MiniDFSCluster.(MiniDFSCluster.java:279)

at org.apache.hadoop.hdfs.MiniDFSCluster.(MiniDFSCluster.java:203)

at org.apache.hadoop.test.MiniHadoopClusterManager.start(MiniHadoopClusterManager.java:152)

at org.apache.hadoop.test.MiniHadoopClusterManager.run(MiniHadoopClusterManager.java:129)

at org.apache.hadoop.test.MiniHadoopClusterManager.main(MiniHadoopClusterManager.java:308)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at org.apache.hadoop.util.ProgramDriver$ProgramDescription.invoke(ProgramDriver.java:68)

at org.apache.hadoop.util.ProgramDriver.driver(ProgramDriver.java:139)

at org.apache.hadoop.test.AllTestDriver.main(AllTestDriver.java:83)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

at java.lang.reflect.Method.invoke(Method.java:597)

at org.apache.hadoop.util.RunJar.main(RunJar.java:186)

**Solution:**  
Make sure that the umask for hdfs and mapred is 0022.

**Problem 4: A cluster fails to run jobs after security is enabled.  
Description:**  
A cluster that was previously configured to not use security may fail to run jobs for certain users on certain TaskTrackers (MRv1) or NodeManagers (YARN) after security is enabled:

1. A cluster is at some point in time configured without security enabled. 2. A user X runs some jobs on the cluster, which creates a local user directory on each TaskTracker or NodeManager. 3. Security is enabled on the cluster. 4. User X tries to run jobs on the cluster, and the local user directory on (potentially a subset of) the TaskTrackers or NodeManagers is owned by the wrong user or has overly-permissive permissions.

The bug is that after step 2, the local user directory on the TaskTracker or NodeManager should be cleaned up, but isn't.

If you're encountering this problem, you may see errors in the TaskTracker or NodeManager logs. The following example is for a 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(LinuxTaskController.java:212)

at org.apache.hadoop.mapred.LinuxTaskController.initializeUser(LinuxTaskController.java:442)

at org.apache.hadoop.mapreduce.server.tasktracker.Localizer.initializeUserDirs(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:2174)

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:370)

at org.apache.hadoop.mapred.LinuxTaskController.runCommand(LinuxTaskController.java:203)

... 5 more

**Solution:**  
Delete the mapred.local.dir or yarn.nodemanager.local-dirs directories for that user across the cluster.

**Problem 5: The NameNode does not start and KrbException Messages (906) and (31) are displayed.  
Description:**  
When you attempt to start the NameNode, a login failure occurs. This failure prevents the NameNode from starting and the following KrbException messages are displayed:

Caused by: KrbException: Integrity check on decrypted field failed (31) - PREAUTH\_FAILED}}

and

Caused by: KrbException: Identifier doesn't match expected value (906)

**Note:**  
These KrbException error messages are displayed only if you enable debugging output. See Appendix D - Enabling Debugging Output for the Sun Kerberos Classes.

**Solution:**  
Although there are several possible problems that can cause these two KrbException error messages to display, here are some actions you can take to solve the most likely problems:

* If you are using CentOS/Red Hat Enterprise Linux 5.6 or later, or Ubuntu, which use AES-256 encryption by default for tickets, you must install the Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File on all cluster and Hadoop user machines. For information about how to verify the type of encryption used in your cluster, see Step 3: If you are Using AES-256 Encryption, install the JCE Policy File. Alternatively, you can change your kdc.conf or krb5.conf to not use AES-256 by removing aes256-cts:normal from the supported\_enctypes field of the kdc.conf or krb5.conf file. Note that after changing the kdc.conf file, you'll need to restart both the KDC and the kadmin server for those changes to take affect. You may also need to recreate or change the password of the relevant principals, including potentially the TGT principal ([krbtgt/REALM@REALM](mailto:krbtgt/REALM@REALM)).
* Recreate the hdfs keytab file and mapred keytab file using the -norandkey option in the xst command (for details, see Step 4: Create and Deploy the Kerberos Principals and Keytab Files).

kadmin.local: xst -norandkey -k hdfs.keytab hdfs/fully.qualified.domain.name HTTP/fully.qualified.domain.name

kadmin.local: xst -norandkey -k mapred.keytab mapred/fully.qualified.domain.name HTTP/fully.qualified.domain.name

**Problem 6: The NameNode starts but clients cannot connect to it and error message contains enctype code 18.  
Description:**  
The NameNode keytab file does not have an AES256 entry, but client tickets do contain an AES256 entry. The NameNode starts but clients cannot connect to it. The error message doesn't refer to "AES256", but does contain an enctype code "18".

**Solution:**  
Make sure the "Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy File" is installed or remove aes256-cts:normal from the supported\_enctypes field of the kdc.conf or krb5.conf file. For more information, see the first suggested solution above for Problem 5.

For more information about the Kerberos encryption types, see <http://www.iana.org/assignments/kerberos-parameters/kerberos-parameters.xml>.

**Problem 9: After you enable cross-realm trust, you can run Hadoop commands in the local realm but not in the remote realm.  
Description:**  
After you enable cross-realm trust, authenticating as a principal in the local realm will allow you to successfully run Hadoop commands, but authenticating as a principal in the remote realm will not allow you to run Hadoop commands. The most common cause of this problem is that the principals in the two realms either don't have the same encryption type, or the cross-realm principals in the two realms don't have the same password. This issue manifests itself because you are able to get Ticket Granting Tickets (TGTs) from both the local and remote realms, but you are unable to get a service ticket to allow the principals in the local and remote realms to communicate with each other.

**Solution:**  
On the local MIT KDC server host, type the following command in the kadmin.local or kadmin shell to add the cross-realm krbtgt principal:

kadmin: addprinc -e "<enc\_type\_list>" krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM

where the parameter specifies the types of encryption this cross-realm krbtgt principal will support: AES, DES, or RC4 encryption. You can specify multiple encryption types using the parameter in the command above, what's important is that at least one of the encryption types parameters corresponds to the encryption type found in the tickets granted by the KDC in the remote realm. For example:

kadmin: addprinc -e "aes256-cts:normal rc4-hmac:normal des3-hmac-sha1:normal" krbtgt/YOUR-LOCAL-REALM.COMPANY.COM@AD-REALM.COMPANY.COM

**Problem 11: Users are unable to obtain credentials when running Hadoop jobs or commands.**  
**Description:**  
This error occurs because the ticket message is too large for the default UDP protocol. An error message similar to the following may be displayed:

13/01/15 17:44:48 DEBUG 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: Fail to create credential.

(63) - No service creds)]

**Solution:**  
Force Kerberos to use TCP instead of UDP by adding the following parameter to libdefaults in the krb5.conf file on the client(s) where the problem is occurring.

[libdefaults]udp\_preference\_limit = 1

**More Info About the udp\_preference\_limit Property**

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.

**Problem 12: Request is a replay exceptions in the logs.  
Description:**  
**Symptom:** 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 Server listener on 8020: readAndProcess threw exception javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: Failure unspecified at GSS-API level (Mechanism l

javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException: Failure unspecified at GSS-API level (Mechanism level: Request is a replay (34))]

at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.java:159)

at org.apache.hadoop.ipc.Server$Connection.saslReadAndProcess(Server.java:1040)

at org.apache.hadoop.ipc.Server$Connection.readAndProcess(Server.java: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(GSSContextImpl.java:323)

at sun.security.jgss.GSSContextImpl.acceptSecContext(GSSContextImpl.java:267)

at com.sun.security.sasl.gsskerb.GssKrb5Server.evaluateResponse(GssKrb5Server.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>(InitSecContextToken.java:79)

at sun.security.jgss.krb5.Krb5Context.acceptSecContext(Krb5Context.java:724)

... 7 more

In addition, this problem can manifest itself as performance issues for all clients in the cluster, including dropped connections, timeouts attempting to make RPC calls, and so on.  
**Likely causes:**

* **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](mailto:myservice@EXAMPLE.COM), services should have per-host principals such as [myservice/host123.example.com@EXAMPLE.COM](mailto:myservice/host123.example.com@EXAMPLE.COM).
* **Clocks not in synch:** All hosts should run NTP so that clocks are kept in synch between clients and servers.