

大数据生态安全框架的实现原理 与最佳实践



【扫码关注】
微信公众号：明哥的IT随笔

Contents

1. 大数据生态安全框架概述

2. HDFS 认证详解

3. HDFS 授权详解

4. HIVE 认证详解

5. HIVE 授权详解

6. 金融行业大数据安全最佳实践

01

大数据生态安全框架概述



大数据安全概述-AAA

数据安全/网络安全领域，有个AAA框架，同样适用于大数据：

- Authentication 认证: 事中: who is trying to access
- Authorization 授权/鉴权: 事前和事中: what resources are allowed to access
- Accounting/Audit/ 审计: 事后: what is being accessed by whom at what time



大数据安全概述-AAA

- Authentication 认证：认证是 AAA 中的第一步，是安全的基础，用户必须证明其身份: The system needs to make sure the person accessing a system is who they say they are.
- 常见的认证机制有：
 - What They Know: 密码或安全问题 (password, security questions) ;
 - Who They Are: 指纹或其他生物特征;
 - What They Have: Access cards, token (jwt)
- 在实际应用中，上述认证方法经常被结合起来使用;



大数据安全概述-AAA

- Authorization 授权/鉴权：包括事前的授权，事中的鉴权；
- 授权一般需要遵循最小化原则；
- 鉴权确保用户没有访问该用户没有被授权访问的资源；
- 经常使用基于角色的授权与鉴权机制 Role-Based Access Control (RBAC)；



大数据安全概述-AAA

Accounting/Audit: 审计: 记录什么用户在什么时间对什么资源进行了什么访问 (what is being accessed by whom at what time for how long)

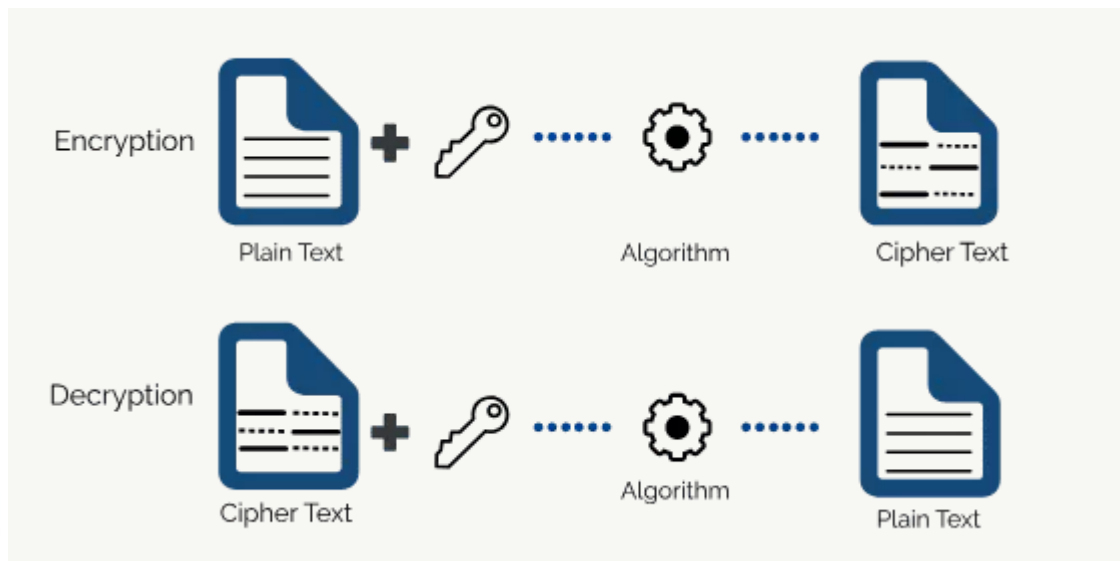
审计是事后的安全监督措施, 可以辅助判断当前的 authenticating 和 authorization 策略是否恰当, 是否需要调整, 从而形成安全闭环;





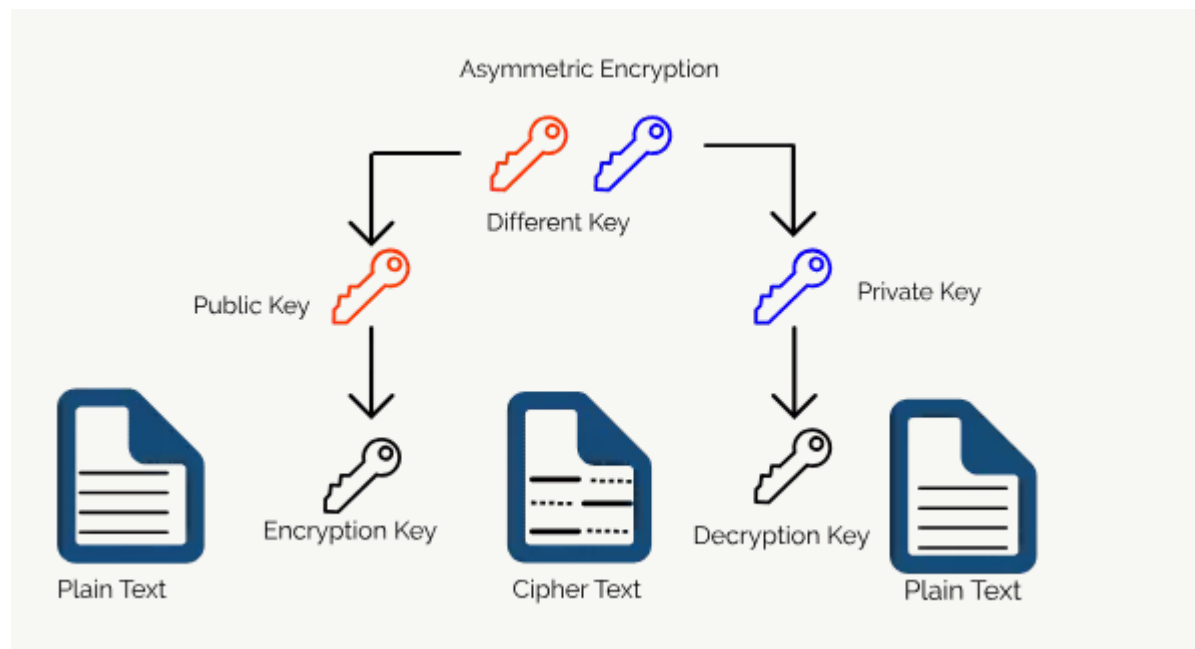
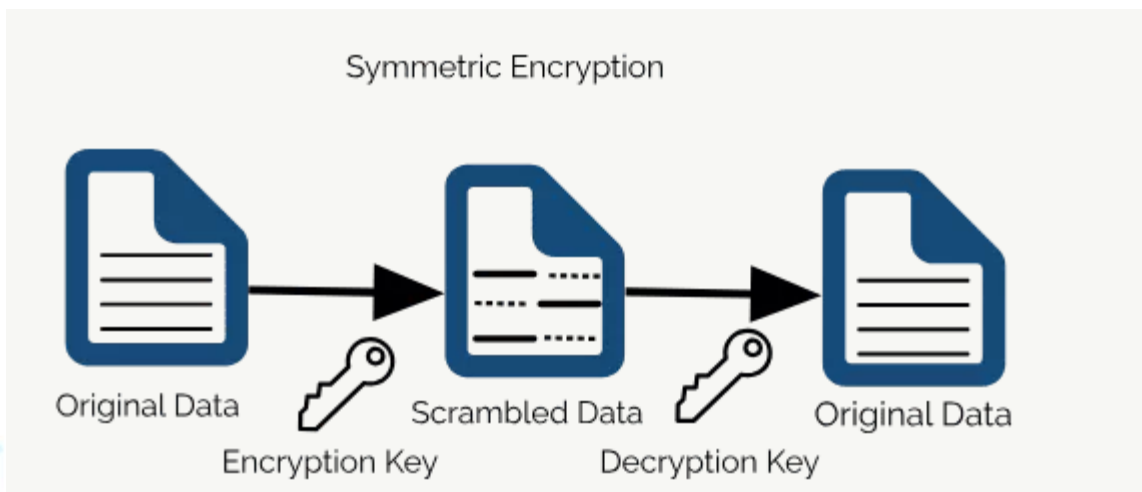
大数据安全概述-Encryption

- 数据安全/网络安全领域，还涉及到 Encryption 加解密；
- 经常接触到 https, ssh 都用到了加解密算法；



大数据安全概述- Encryption

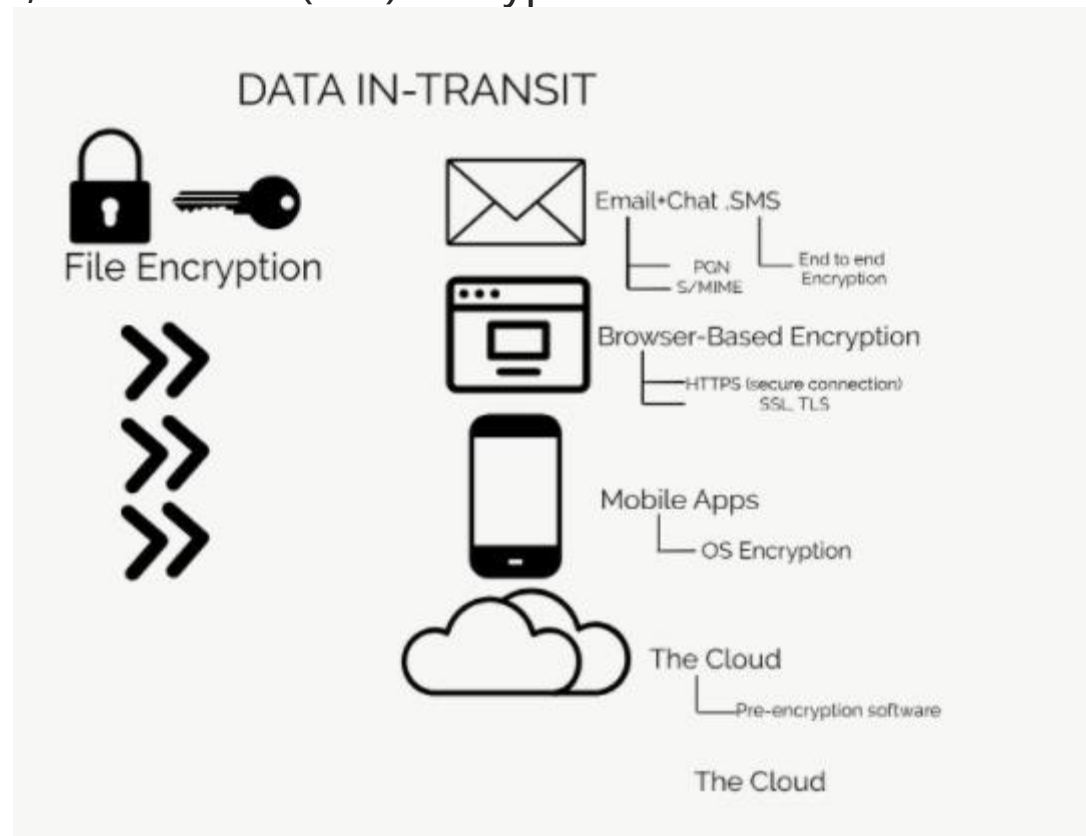
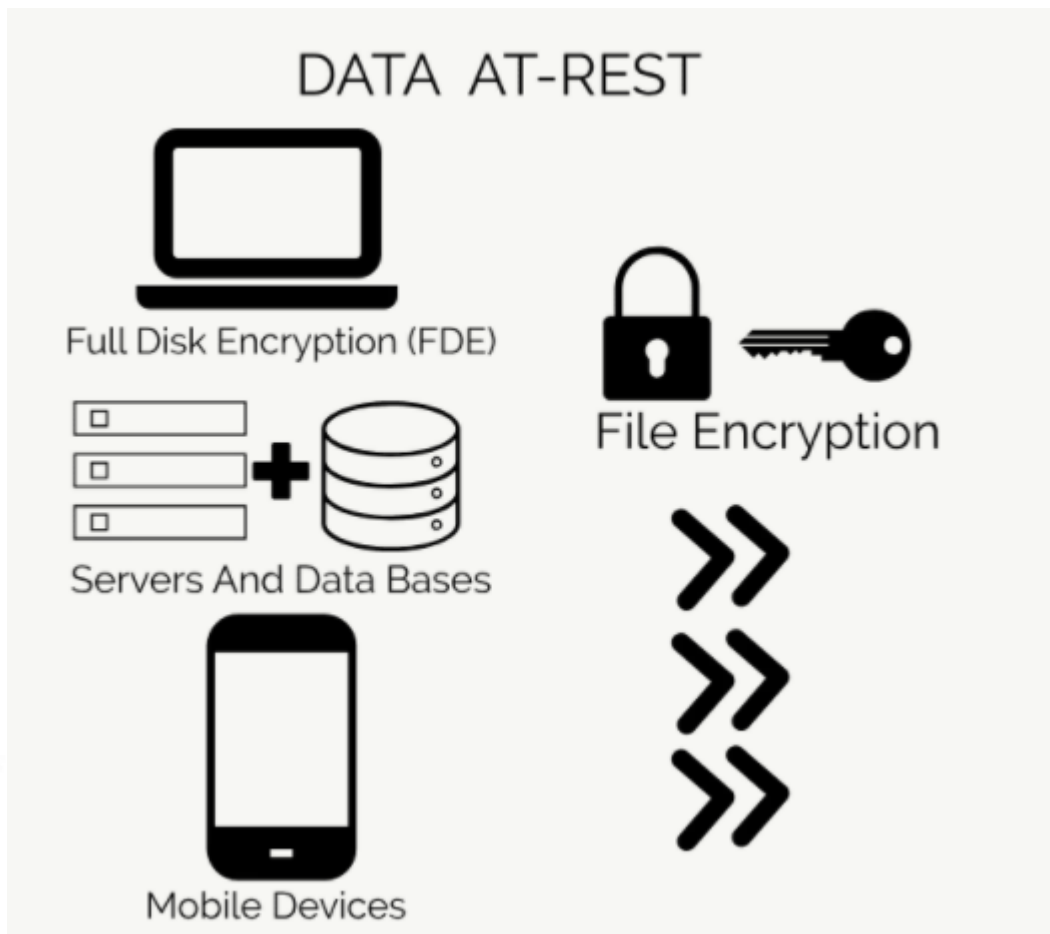
- 加解密的算法，包括对称加密算法和非对称加密算法：DES, AES, RSA(ssh)
- Ssh 底层使用的 RSA加密算法是非对称加密算法；
- 大数据集群中，出于运维方便，经常需要配置 Ssh 免密码登录，此时拷贝给其它机器的是自己的公钥
/root/.ssh/id_rsa.pub, /root/.ssh/authorized_keys





大数据安全概述- Encryption

- 加解密包括对静态数据的加解密，和对传输过程中的数据的加解密：data at rest, data in motion/in transit
- 常见的术语有：FDE: full disk encryption, file encryption, End-to-End (e2e) encryption





大数据安全概述-audit

- HDFS 在 Audit 上，默认通过log4j在/var/log/Hadoop-hdfs目录下打印了audit日志；

```
-rw-r--r-- 1 hdfs hdfs 268435564 11月 15 02:49 hdfs-audit.log.7
-rw-r--r-- 1 hdfs hdfs 268435510 11月 5 05:10 hdfs-audit.log.8
-rw-r--r-- 1 hdfs hdfs 268435546 8月 20 2021 hdfs-audit.log.9
-rw-r--r-- 1 hdfs hdfs 13636229 3月 30 19:29 SecurityAuth-hdfs.audit
-rw-r--r-- 1 hdfs hdfs 158507322 3月 17 18:53 SecurityAuth-hdfs.audit.1
-rw-r--r-- 1 hdfs hdfs 294764436 3月 17 13:55 SecurityAuth-hdfs.audit.2
-rw-r--r-- 1 hdfs hdfs 312081493 2月 19 05:21 SecurityAuth-hdfs.audit.3
drwxr-xr-x 2 hdfs hdfs 4096 4月 16 2019 stacks
[root@node3 hadoop-hdfs]# pwd
/var/log/hadoop-hdfs
[root@node3 hadoop-hdfs]#
```



大数据安全概述-encryption

- HDFS 在 Encryption上, 实现了 end-to-end Transparent Encryption, 包含了 *at-rest encryption* 和 *in-transit encryption*, 底层通过 KMS(Key Management Server) 服务支持了多个 encryption zone;

Overview

HDFS implements *transparent, end-to-end* encryption. Once configured, data read from and written to special HDFS directories is *transparently* encrypted and decrypted without requiring changes to user application code. This encryption is also *end-to-end*, which means the data can only be encrypted and decrypted by the client. HDFS never stores or has access to unencrypted data or unencrypted data encryption keys. This satisfies two typical requirements for encryption: *at-rest encryption* (meaning data on persistent media, such as a disk) as well as *in-transit encryption* (e.g. when data is travelling over the network).

Background

Encryption can be done at different layers in a traditional data management software/hardware stack. Choosing to encrypt at a given layer comes with different advantages and disadvantages.

- **Application-level encryption.** This is the most secure and most flexible approach. The application has ultimate control over what is encrypted and can precisely reflect the requirements of the user. However, writing applications to do this is hard. This is also not an option for customers of existing applications that do not support encryption.
- **Database-level encryption.** Similar to application-level encryption in terms of its properties. Most database vendors offer some form of encryption. However, there can be performance issues. One example is that indexes cannot be encrypted.
- **Filesystem-level encryption.** This option offers high performance, application transparency, and is typically easy to deploy. However, it is unable to model some application-level policies. For instance, multi-tenant applications might want to encrypt based on the end user. A database might want different encryption settings for each column stored within a single file.
- **Disk-level encryption.** Easy to deploy and high performance, but also quite inflexible. Only really protects against physical theft.

HDFS-level encryption fits between database-level and filesystem-level encryption in this stack. This has a lot of positive effects. HDFS encryption is able to provide good performance and existing Hadoop applications are able to run transparently on encrypted data. HDFS also has more context than traditional filesystems when it comes to making policy decisions.

HDFS-level encryption also prevents attacks at the filesystem-level and below (so-called "OS-level attacks"). The operating system and disk only interact with encrypted bytes, since the data is already encrypted by HDFS.



大数据安全概述

- AAA + Encryption 的安全框架，同样适用于大数据；
- 本次分享，我们侧重于Authentication 和 authorization, 不会过多讲述 audit 和 encryption;
- 本此分享，我们侧重于大数据存储框架的安全，主要是 HDFS 和 HIVE;
- 大数据生态中的 authentication 认证，事实上的标准是使用 Kerberos 协议，我们后文会有详细讲述；
- 大数据生态中的各种存储系统，如HDFS/hive/hbase/zookeeper/kafka等，都支持开启Kerberos安全认证；
- 当大数据集群中的存储系统如HDFS/hive/hbase/zookeeper/kafka等开启了kerberos安全认证后，访问这些存储系统的客户端，包含各种计算引擎如 hive/hbase/spark/flink 的系统服务，和用户编写的各种应用如 spark/hive/flink等，都需要经过 kerberos 认证后才能访问对应的服务（当然还需要 authentication 鉴权！）

02

HDFS认证详解



HDFS认证详解

- HDFS在认证上，支持两种方式，通过参数 `hadoop.security.authentication` 来进行控制，可选的参数有：`simple (no authentication)` 和 `kerberos`；
- 该参数是服务端参数，不能在客户端覆盖！
- Kerberos 是一种计算机网络授权协议（network authentication protocol），用来在非安全网络中，对个人通信以安全的手段进行身份认证；
- Kerberos 协议常见的实现有 MIT kerberos（麻省理工学院开发），ApacheDS (embedded kerberos), freeIPA 等。

KERBEROS





HDFS认证详解 `hadoop.security.authentication = simple`

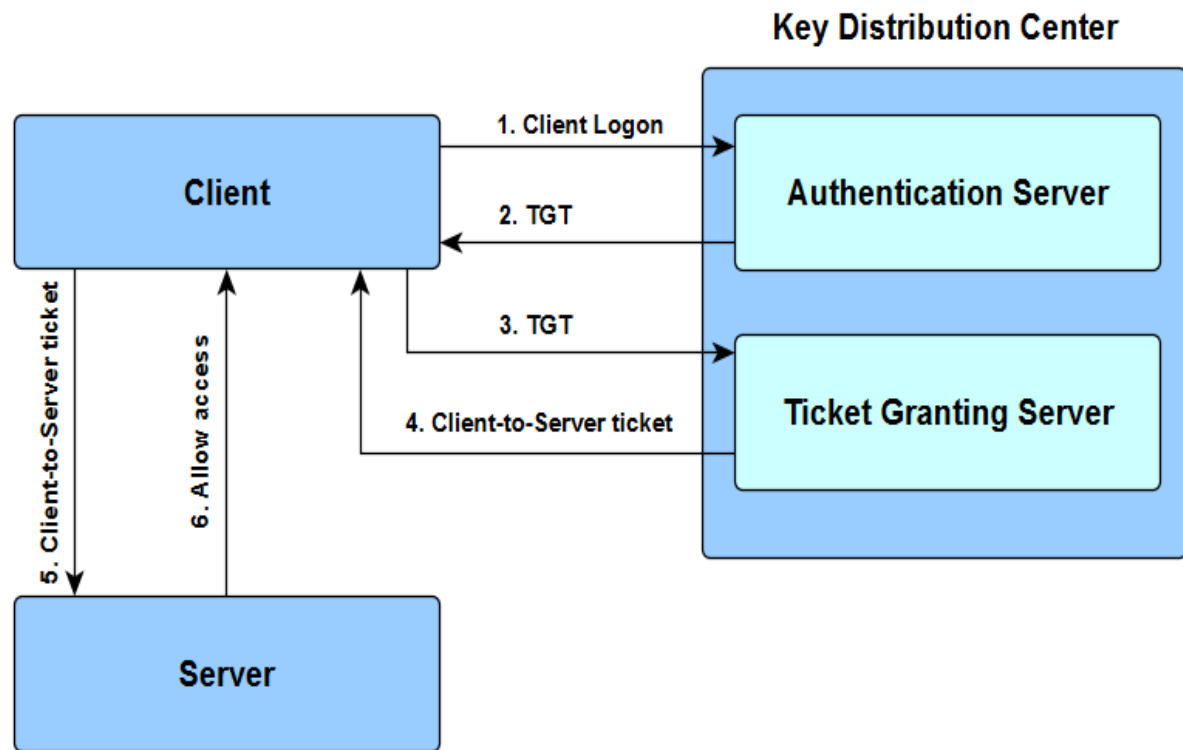
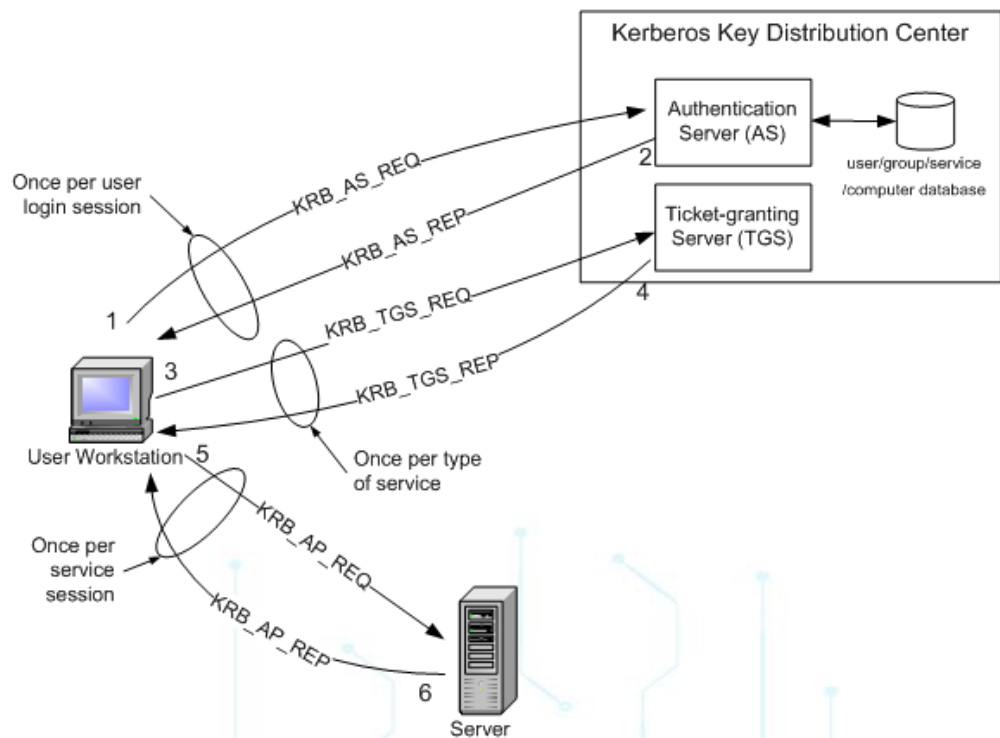
- 此时用户身份由环境变量或系统参数 `HADOOP_USER_NAME` 决定，当该环境变量/系统参数不存在时，由当前Linux登录用户身份决定；
- 所以在没有开启KERBEROS时，我们可以切换到业务用户身份下提交命令，或通过以下方式指定登录用户：
 - `System.setProperty(UserGroupInformation.HADOOP_USER_NAME, "randomUser");`
 - `Export HADOOP_USER_NAME = randomUser;`

```
189 @Override
190 public boolean commit() throws LoginException {
191     LOG.debug("hadoop login commit");
192     // if we already have a user, we are done.
193     if (!subject.getPrincipals(User.class).isEmpty()) {
194         LOG.debug("Using existing subject: {}", subject.getPrincipals());
195         return true;
196     }
197     Principal user = getCanonicalUser(KerberosPrincipal.class);
198     if (user != null) {
199         LOG.debug("Using kerberos user: {}", user);
200     }
201     //If we don't have a kerberos user and security is disabled, check
202     //if user is specified in the environment or properties
203     if (!isSecurityEnabled() && (user == null)) {
204         String envUser = System.getenv(HADOOP_USER_NAME);
205         if (envUser == null) {
206             envUser = System.getProperty(HADOOP_USER_NAME);
207         }
208         user = envUser == null ? null : new User(envUser);
209     }
210     // use the OS user
211     if (user == null) {
212         user = getCanonicalUser(OS_PRINCIPAL_CLASS);
213         LOG.debug("Using local user: {}", user);
214     }
```

HDFS认证详解

hadoop.security.authentication=kerberos

Kerberos: 3 headed dog: client, server, kdc





HDFS认证详解

hadoop.security.authentication=kerberos

- 查看kdc服务状态与日志: `systemctl status krb5kdc/systemctl status kadmind/journalctl -u krb5kdc`
- 查看kerberos配置: `/etc/krb5.conf`
- Kerberos 底层使用了 tcp与udp协议, 主要包括88与749 端口;
- Kerberos 要求所有节点的时钟是同步的, 以避免重放攻击 replay attack;

```
[root@uf30-1 ~]# less /etc/services |grep kerberos
kerberos      88/tcp        krb5           # Kerberos v5
kerberos      88/udp        krb5           # Kerberos v5
kerberos-adm  749/tcp      `kadmin'      (v5)
kerberos-adm  749/udp      # kerberos administration
kerberos-iv   750/udp      krb4sec kdc loadav
kerberos-iv   750/tcp      krb4sec kdc rfile
kerberos_master 751/udp      pump          # Kerberos authentication
kerberos_master 751/tcp      pump          # Kerberos authentication
[root@uf30-1 ~]#
```



HDFS认证详解

hadoop.security.authentication=kerberos

常用的 Kerberos 命令有:

Klist/klist -kt xx

Kinit/kinit -kt xxx

Kdestroy

```
[root@uf30tdh1 ~]# kdestroy
[root@uf30tdh1 ~]# klist -kt /home/hundsun/workspace/kerberos/dap/dap.keytab
Keytab name: FILE:/home/hundsun/workspace/kerberos/dap/dap.keytab
KVNO Timestamp          Principal
-----
0 2020-04-07T10:17:28 dap@TDH
0 2020-04-07T10:17:28 dap@TDH
0 2020-04-07T10:17:28 dap@TDH
0 2020-04-07T10:17:28 dap@TDH
0 2020-04-07T10:17:28 dap@TDH
[root@uf30tdh1 ~]# kinit dap -kt /home/hundsun/workspace/kerberos/dap/dap.keytab
[root@uf30tdh1 ~]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: dap@TDH

Valid starting    Expires          Service principal
2022-03-30T14:34:17 2022-03-31T14:34:17 krbtgt/TDH@TDH
renew until 2022-04-06T14:34:17
[root@uf30tdh1 ~]# ls -al /tmp/krb5cc_0
-rw----- 1 root root 416 3月 30 14:34 /tmp/krb5cc_0
[root@uf30tdh1 ~]#
```




HDFS认证详解

hadoop.security.authentication=kerberos

- 管理principal与keytab, 经常使用 Kadmin.local;
- addprinc -randkey xx/delprinc xx/modprinc/getprinc/listprincs
- kadmin.local -q "getprinc liming@TEST.COM" | grep -i life
- xst -norandkey -k dap.keytab [dap/uf30-1@CDH.COM](#)
- Cpw/change_password,
- Ktadd/xst/ktremove

```
[root@uf30-1 ~]# kadmin.local
Authenticating as principal root/admin@CDH.COM with password.
kadmin.local: ? 
Available kadmin.local requests:

add_principal, addprinc, ank          Add principal
delete_principal, delprinc            Delete principal
modify_principal, modprinc            Modify principal
rename_principal, renprinc            Rename principal
change_password, cpw                  Change password
get_principal, getprinc               Get principal
list_principals, listprincs, get_principals, getprincs  List principals
add_policy, addpol                    Add policy
modify_policy, modpol                 Modify policy
delete_policy, delpol                 Delete policy
get_policy, getpol                    Get policy
list_policies, listpols, get_policies, getpols          List policies
```



HDFS认证详解

hadoop.security.authentication=kerberos

- Keytab vs KRB5CCNAME/Ccache:
- KRB5CCNAME/Ccache: by default is /tmp/krb5cc_0 in linux;
- Kinit -R: you can use kinit -R to renew your tgt when the ticket has not expired and is still within the renewable life, after successful renew, the new KRB5CCNAME,, will be modified;

The screenshot shows the Apache Spark 3.2.1 documentation page. The navigation bar includes links for Overview, Programming Guides, API Docs, Deploying, and More, along with a search bar. The main content area is titled "Long-Running Applications" and discusses issues with delegation token lifetime. It mentions that Spark supports automatically creating new tokens and provides two ways to enable this functionality: "Using a Keytab" and "Using a ticket cache". Red arrows point to these two sections. The "Using a Keytab" section explains that Spark can maintain a valid Kerberos login by providing a principal and keytab. The "Using a ticket cache" section explains that Spark can use a local Kerberos ticket cache for authentication. The page also notes that the ticket cache can be customized by setting the KRB5CCNAME environment variable.

Users can exclude Kerberos delegation token renewal at resource scheduler. Currently it is only supported on YARN. The configuration is covered in the [Running Spark on YARN](#) page.

Long-Running Applications

Long-running applications may run into issues if their run time exceeds the maximum delegation token lifetime configured in services it needs to access.

This feature is not available everywhere. In particular, it's only implemented on YARN and Kubernetes (both client and cluster modes), and on Mesos when using client mode.

Spark supports automatically creating new tokens for these applications. There are two ways to enable this functionality.

Using a Keytab

By providing Spark with a principal and keytab (e.g. using `spark-submit` with `--principal` and `--keytab` parameters), the application will maintain a valid Kerberos login that can be used to retrieve delegation tokens indefinitely.

Note that when using a keytab in cluster mode, it will be copied over to the machine running the Spark driver. In the case of YARN, this means using HDFS as a staging area for the keytab, so it's strongly recommended that both YARN and HDFS be secured with encryption, at least.

Using a ticket cache

By setting `spark.kerberos.renewal.credentials` to `ccache` in Spark's configuration, the local Kerberos ticket cache will be used for authentication. Spark will keep the ticket renewed during its renewable life, but after it expires a new ticket needs to be acquired (e.g. by running `kinit`).

It's up to the user to maintain an updated ticket cache that Spark can use.

The location of the ticket cache can be customized by setting the `KRB5CCNAME` environment variable.



HDFS认证详解

`hadoop.security.authentication=kerberos`

- 问题现象：客户端执行命令 `kinit -R` 报错：“KDC can't fulfill requested option while renewing credentials”
- 问题原因：This error occurs when the KDC fails to generate a renewable ticket-granting ticket (TGT);
- 问题解决1：服务端更改或确认配置文件 `/etc/krb5.conf` 中对应的 `realms` 段中需要配置参数允许刷新：
`max_renewable_life`
- 问题解决2：服务端更改或确认上述两个 `principal` 的参数 `maxrenewlife`，示例命令如下
- `modprinc -maxrenewlife "1 week" +allow_renewable liming@TEST.COM;`
- `modprinc -maxrenewlife "1 week" +allow_renewable krbtgt/TEST.COM@TEST.COM`



HDFS认证详解-groups 如何确定

- 确定了user后, 如何确定其 groups 是由参数 `hadoop.security.group.mapping` 决定的;
- 该参数的默认值是 `org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback`;
- Hadoop also supports special group mapping mechanisms through LDAP and composition of LDAP and operating system group name resolution, which require additional configurations
- 在默认配置下, 确定的groups的值, 等同于Linux 下 `bash -c groups/id -gn/id -Gn`

```
mappingServiceProvider.java x JniBasedUnixGroupsMappingWithFallback.java x ShellBasedUnixGroupsMapping.java x Shell.java x JniBasedUnixGroup
A command to get a given user's groups list. If the OS is not WINDOWS, the command will get the user's
primary group first and finally get the groups list which includes the primary group. i.e. the user's primary
group will be included twice.
221 @ public static String[] getGroupsForUserCommand(final String user) {
222 // 'groups username' command return is inconsistent across different unixes
223 if (WINDOWS) {
224 return new String[]
225 {getWinUtilsPath(), "groups", "-F", "\"" + user + "\""};
226 } else {
227 String quotedUser = bashQuote(user);
228 return new String[] {"bash", "-c", "id -gn " + quotedUser +
229 " " + id -Gn " + quotedUser};
230 }
231 }
232
A command to get a given user's group id list. The command will get the user's primary group first and
finally get the groups list which includes the primary group. i.e. the user's primary group will be included
twice. This command does not support Windows and will only return group names.
240 @ public static String[] getGroupsIDForUserCommand(final String user) {
241 // 'groups username' command return is inconsistent across different unixes
242 if (WINDOWS) {
243 return new String[] {getWinUtilsPath(), "groups", "-F", "\"" + user +
244 "\""};
245 } else {
```

```
JniBasedUnixGroupsMappingWithFallback.java x ShellBasedUnixGroupsMapping.java x Shell.java x JniBasedUnixGroupsMapping
Set up our JNI resources.
Throws: RuntimeException - if setup fails.
native static void anchorNative();

Get the set of groups associated with a user.
Params: username - The user name
Returns: The set of groups associated with a user.
native static String[] getGroupsForUser(String username);

Log an error message about a group. Used from JNI.
static private void logError(int groupId, String error) {
LOG.error("error looking up the name of group " + groupId + ": " + error);
}

@Override
public List<String> getGroups(String user) throws IOException {
return Arrays.asList(getGroupsInternal(user));
}
```



HDFS认证详解-相关参数

- `hadoop.security.authentication`
- `hadoop.security.group.mapping`
- `hadoop.proxyuser.xxx.groups`
- `hadoop.proxyuser.xxx.hosts`

03

HDFS授权详解



HDFS授权详解-posix mode

HDFS在授权上，大体上遵循了类 linux文件系统的 posix 模型机制：

- 每个文件和目录都有对应的 owner/group, 并可以针对 owner/group/others 分别进行授权;
- 对文件来说, 读文件需要 R 权限, 写或追加写需要 W 权限, X 权限没有意义 (因为hdfs文件没有所谓的 executables files 可执行文件) ;
- 对目录来说, 查看目录下内容需要R权限, 在该目录下创建和删除文件或目录需要 W 权限, 访问该目录下的文件或目录需要x权限;
- Hdfs 可以对目录设置 sticky bit, 防止目录下文件被误删除: The sticky bit can be set on directories, preventing anyone except the superuser, directory owner or file owner from deleting or moving the files within the directory.
- HDFS 对文件设置 sticky bit 没有任何效果: Setting the sticky bit for a file has no effect.
- 不同于标准 POSIX 模型的一点是, hdfs 的目录和文件都没有 setuid/setgid bits; (hdfs文件没有所谓的 executables files 可执行文件) ;

```
[root@uf30-1 ~]# hdfs dfs -getfacl /tmp
# file: /tmp
# owner: hdfs
# group: supergroup
# flags: --t
user::rwx
group::rwx
other::rwx

[root@uf30-1 ~]# hdfs dfs -ls /
Found 6 items
drwxr-xr-x - flink flink          0 2021-04-13 15:00 /flink
drwxr-xr-x - hbase hbase         0 2021-03-01 16:45 /hbase
drwxrwxr-x - solr solr           0 2020-11-05 09:34 /solr
drwxr-xr-x - spark supergroup    0 2021-09-02 10:31 /spark-jars
drwxrwxrwt - hdfs supergroup     0 2021-07-23 19:24 /tmp
drwxr-xr-x - hdfs supergroup     0 2021-09-01 16:09 /user
[root@uf30-1 ~]#
```



HDFS授权详解-posix mode

- 创建目录或文件时，其默认的 owner/group 是这样决定的：its owner is the user identity of the client process, and its group is the group of the parent directory (the BSD rule);
- 创建文件或目录时，其默认的 MODE 是由 umask 参数决定的：fs.permissions.umask-mode
- Umask 默认的配置是 022，此时默认所有用户对所有目录和文件都有读权限：this is typically 755 for new directories and 644 for new files: 022 is octal for u=rwx,g=r-x,o=r-x in symbolic;
- Umask 常见的配置还有007和077，其中后者默认阻止了非owner用户对所有目录和文件的访问权限，是最为严苛的授权；（007 is octal for “u=rwx,g=rwx,o=” in symbolic;）



HDFS授权详解-superuser

- HDFS 有超级用户super-user 的概念: 启动 namenode 进程的用户即是超级用户, HDFS的超级用户拥有所有权限, 类似 LINUX 的超级用户 ROOT;(超级用户是动态的, 不需要通过参数配置, 谁启动 namenode 谁就是超级用户, 该用户一般是 hdfs)
- HDFS 有超级用户组 superusergroup 的概念: 超级用户组下的所有用户都是超级用户, 超级用户组只能是一个而不能是多个, 通过参数 dfs.permissions.superusergroup 进行配置, 默认是 supergroup;
- HDFS Web Server 用户的身份: 需要通过参数进行配置, 比如: dfs.web.ugi = webuser,webgroup (新版本是 hadoop.http.staticuser.user), 当该用户不是超级用户时, 通过 web server 可能不能查看到文件系统的所有内容

```
[root@uf30-1 ~]# hdfs getconf -confKey dfs.web.ugi
Configuration dfs.web.ugi is missing.
[root@uf30-1 ~]# hdfs getconf -confKey hadoop.http.staticuser.user
dr.who
[root@uf30-1 ~]# hdfs getconf -confKey dfs.permissions.superusergroup
supergroup
[root@uf30-1 ~]#
```



HDFS授权详解-posix acl

- 除了传统的 POSIX 权限模型，HDFS 也支持 POSIX ACLs (Access Control Lists);
- 通过 ACL，可以对目录或文件的 owner/group 之外的特定用户或用户组，授予不同于 others 的特定的权限，相比 owner/group/others 的粗粒度的授权，粒度更细也更为灵活；
- ACL 分为两种：access acl 和 default acl，其中前者用来在访问文件或目录内容时进行权限校验，后者用来确定在配置了 ACL 的目录下创建子目录或子文件时，这些子目录子文件的 access acl 和 default acl; (只有目录才有default ACL)
- 对特定的目录或文件，可以声明的 ACL entry 的个数是有上限的,最大为64: The maximum number is 32 for access and 32 for default entries which is 64 in total.
- ACL 对 namenode 造成了额外的压力: A file with an ACL incurs an additional cost in memory in the NameNode compared to a file that has only permission bits.
- 最佳实践是: Best practice is to rely on traditional permission bits to implement most permission requirements, and define a smaller number of ACLs to augment the permission bits with a few exceptional rules.



HDFS授权详解-posix acl

ACL 常见命令:

- 查看 acl: `hdfs dfs -ls <args>`: 针对设置了ACL的文件或目录, 注意输出中有个 '+' character
- 查看 acl: `hdfs dfs -getfacl [-R] <path>`
- 设置 acl: `hdfs dfs -setfacl [-R] [-b |-k -m |-x <acl_spec> <path>] [--set <acl_spec> <path>]`

对HIVE外表配置ACL, 示例命令如下: (access acl+default acl)

- `hdfs dfs -getfacl /user/hundsun/dap/hive/liming2`
- `hdfs dfs -setfacl -R -m group:hive:rwX /user/hundsun/dap/hive/liming2`
- `hdfs dfs -setfacl -m default:group:hive:rwX /user/hundsun/dap/hive/liming2`

setfacl

Usage: `hadoop fs -setfacl [-R] [-b |-k -m |-x <acl_spec> <path>] [--set <acl_spec> <path>]`

Sets Access Control Lists (ACLs) of files and directories.

Options:

- `-b`: Remove all but the base ACL entries. The entries for user, group and others are retained for compatibility with permission bits.
- `-k`: Remove the default ACL.
- `-R`: Apply operations to all files and directories recursively.
- `-m`: Modify ACL. New entries are added to the ACL, and existing entries are retained.
- `-x`: Remove specified ACL entries. Other ACL entries are retained.
- `--set`: Fully replace the ACL, discarding all existing entries. The `acl_spec` must include entries for user, group, and others for compatibility with permission bits. If the ACL spec contains only access entries, then the existing default entries are retained. If the ACL spec contains only default entries, then the existing access entries are retained. If the ACL spec contains both access and default entries, then both are replaced.
- `acl_spec`: Comma separated list of ACL entries.
- `path`: File or directory to modify.

Examples:

- `hadoop fs -setfacl -m user:hadoop:rw- /file`
- `hadoop fs -setfacl -x user:hadoop /file`
- `hadoop fs -setfacl -b /file`
- `hadoop fs -setfacl -k /dir`
- `hadoop fs -setfacl --set user::rw-,user:hadoop:rw-,group::r--,other::r-- /file`
- `hadoop fs -setfacl -R -m user:hadoop:r-x /dir`



HDFS授权详解-posix acl

ACL 示例输出如下：（Each ACL entry names a specific user or group and grants or denies read, write and execute permissions for that specific user or group）

- user::rwx
- group::r-x
- other::r-x
- default:user::rwx
- default:user:bruce:rwx
- default:group::r-x
- default:group:sales:rwx
- default:mask::r-x
- default:other::r-x

```
[root@uf30-1 1818-hdfs-NAMENODE-nnRpcWait]# hdfs dfs -ls /user/hundsun/dap/hive/
Found 8 items
drwxrwxrwx - hive hive      0 2021-12-07 17:56 /user/hundsun/dap/hive/HS_ODS
drwxrwxrwx - dap  hive      0 2021-09-01 16:19 /user/hundsun/dap/hive/hs_custom
drwxrwxrwx - dap  hive      0 2022-03-29 13:36 /user/hundsun/dap/hive/hs_mid
drwxrwxrwx - dap  hive      0 2022-03-29 16:29 /user/hundsun/dap/hive/hs_ods
drwxrwxrwx - hive hive      0 2022-03-29 16:42 /user/hundsun/dap/hive/hs_sr
drwxrwxrwx - dap  hive      0 2022-02-20 01:14 /user/hundsun/dap/hive/hs_st
drwxrwxrwx - dap  hive      0 2022-03-14 13:27 /user/hundsun/dap/hive/hs_tmp
drwxrwxr-x+ - dap  dap      0 2022-03-30 17:08 /user/hundsun/dap/hive/liming2
[root@uf30-1 1818-hdfs-NAMENODE-nnRpcWait]# hdfs dfs -getfacl /user/hundsun/dap/hive/liming2
# file: /user/hundsun/dap/hive/liming2
# owner: dap
# group: dap
user::rwx
group::r-x
group:hive:rwx
mask::rwx
other::r-x
default:user::rwx
default:group::r-x
default:group:hive:rwx
```



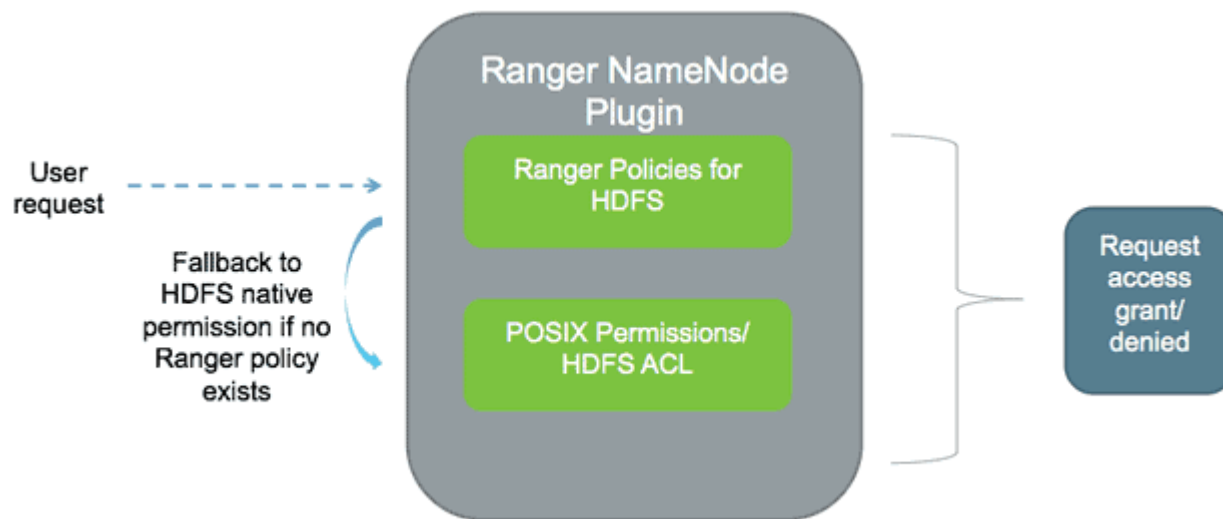
HDFS授权详解-鉴权-hdfs native

- HDFS在鉴权上，通过参数 `dfs.permissions.enabled` 来进行控制，可已配置鉴权也可以配置不鉴权 `false/true`，该参数默认是 `true`，当该参数配置为 `false` 时，将不会进行任何 `permission checking`；
- 更改该参数的值，不会改变文件和目录的 `mode/owner/group/acl`等；
- 该参数是服务端参数，不能在客户端覆盖！



HDFS授权详解-鉴权-ranger

- 除了使用 hdfs 的 native 鉴权机制，还可以结合使用 ranger 的权限机制；
- Apache Ranger 基于插件机制，提供了对 Hadoop 生态众多组件的集中的权限管控机制：a centralized security administration solution for Hadoop；
- Ranger hdfs 插件是对 hdfs native permission model 的补充：Ranger plugin for HDFS checks for Ranger policies and if a policy exists, access is granted to user. If a policy doesn't exist in Ranger, then Ranger would default to native permissions model in HDFS (POSIX or HDFS ACL).





HDFS授权详解-proxy代理机制

- Hadoop 还有代理用户 proxyuser 的概念，一般 hue/hive/sqoop/flume 等服务的系统用户都支持代理机制；
- 所谓代理机制，是由当前的系统用户/真实用户realuser/超级用户，如 hive/sqoop 等服务的进程对应的用户，代理最终的业务用户比如dap/cic等，对底层的 hdfs进行访问；
- 经过代理后，hdfs 进行权限校验时，是针对最终业务用户比如 hundsun/dap/cic,进行权限校验；
- 没有使用代理时，hdfs 进行权限校验时，是针对系统用户比如 hive/hue/sqoop,进行权限校验；

Use Case

The code example described in the next section is applicable for the following use case.

A superuser with username 'super' wants to submit job and access hdfs on behalf of a user joe. The superuser has kerberos credentials but user joe doesn't have any. The tasks are required to run as user joe and any file accesses on namenode are required to be done as user joe. It is required that user joe can connect to the namenode or job tracker on a connection authenticated with super's kerberos credentials. In other words super is impersonating the user joe.

Some products such as Apache Oozie need this.

Code example

In this example super's credentials are used for login and a proxy user ugi object is created for joe. The operations are performed within the doAs method of this proxy user ugi object.

```
...
//Create ugi for joe. The login user is 'super'.
UserGroupInformation ugi =
    UserGroupInformation.createProxyUser("joe", UserGroupInformation.getLoginUser());
ugi.doAs(new PrivilegedExceptionAction<Void>() {
    public Void run() throws Exception {
        //Submit a job
        JobClient jc = new JobClient(conf);
        jc.submitJob(conf);
        //OR access hdfs
        FileSystem fs = FileSystem.get(conf);
        fs.mkdir(someFilePath);
    }
}
```


HDFS授权详解-proxy代理机制

当指定了环境变量或系统参数 HADOOP_PROXY_USER 时, HADOOP 会自动使用代理机制, 且被代理用户就是该环境变量/系统参数只当的值, 所以可以通过以下方式指定被代理的业务用户:

- `System.setProperty(UserGroupInformation.HADOOP_PROXY_USER, "randomUser");`
- `Export HADOOP_PROXY_USER = randomUser;`

```
UserGroupInformation.java x TestProxyUserFromEnv.java x
29 public class TestProxyUserFromEnv {
    Test HADOOP_PROXY_USER for impersonation
31
32 @Test
33 public void testProxyUserFromEnvironment() throws IOException {
34     String proxyUser = "foo.bar";
35     System.setProperty(UserGroupInformation.HADOOP_PROXY_USER, proxyUser);
36     UserGroupInformation ugi = UserGroupInformation.getLoginUser();
37     assertEquals(proxyUser, ugi.getUserName());
38
39     UserGroupInformation realUgi = ugi.getRealUser();
40     assertNotNull(realUgi);
41     // get the expected real user name
42     Process pp = Runtime.getRuntime().exec("command: 'whoami'");
43     BufferedReader br = new BufferedReader
44         (new InputStreamReader(pp.getInputStream()));
45     String realUser = br.readLine().trim();
46
47     // On Windows domain joined machine, whoami returns the username
48     // in the DOMAIN\username format, so we trim the domain part before
49     // the comparison. We don't have to special case for Windows
50     // given that Unix systems do not allow slashes in usernames.
51     int backslashIndex = realUser.indexOf('\\');
52     if (backslashIndex != -1) {
53         realUser = realUser.substring(backslashIndex + 1);
54     }
55     assertEquals(realUser, realUgi.getUserName());
56 }
```

```
UserGroupInformation.java x TestProxyUserFromEnv.java x User.java x
HADOOP_PROXY_USER x Cc W * 3/5 ↑ ↓ □ + - × ÷ = > <
712 public static void loginUserFromSubject(Subject subject) throws IOException {
713     setLoginUser(createLoginUser(subject));
714 }
715
716 private static
717 UserGroupInformation createLoginUser(Subject subject) throws IOException {
718     UserGroupInformation realUser = doSubjectLogin(subject, params: null);
719     UserGroupInformation loginUser = null;
720     try {
721         // If the HADOOP_PROXY_USER environment variable or property
722         // is specified, create a proxy user as the logged in user.
723         String proxyUser = System.getenv(HADOOP_PROXY_USER);
724         if (proxyUser == null) {
725             proxyUser = System.getProperty(HADOOP_PROXY_USER);
726         }
727         loginUser = proxyUser == null ? realUser : createProxyUser(proxyUser, realUser);
728     }
```

HDFS授权详解-proxy代理机制

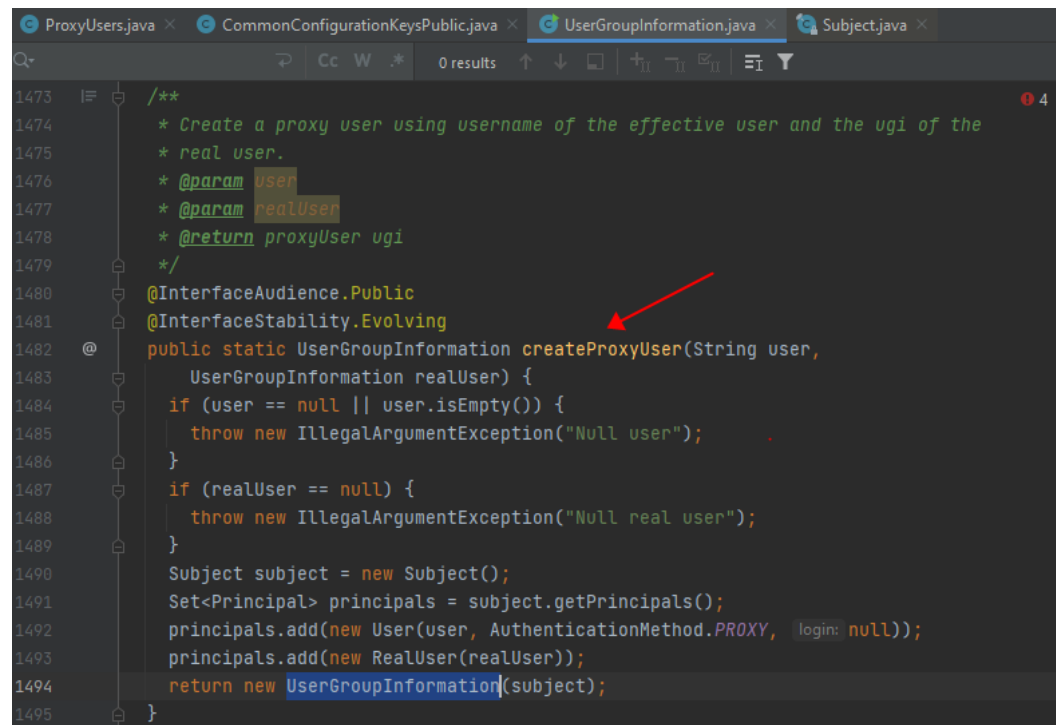
当然也可以在代码中，显示指定代理用户;

- `UserGroupInformation.createProxyUser("randomUser", UserGroupInformation.getLoginUser());`

Code example

In this example super's credentials are used for login and a proxy user ugi object is created for joe. The operations

```
...
//Create ugi for joe. The login user is 'super'.
UserGroupInformation ugi =
    UserGroupInformation.createProxyUser("joe", UserGroupInformation.getLoginUser());
ugi.doAs(new PrivilegedExceptionAction<Void>() {
    public Void run() throws Exception {
        //Submit a job
        JobClient jc = new JobClient(conf);
        jc.submitJob(conf);
        //OR access hdfs
        FileSystem fs = FileSystem.get(conf);
        fs.mkdir(someFilePath);
    }
}
```



```
ProxyUsers.java x CommonConfigurationKeysPublic.java x UserGroupInformation.java x Subject.java x
0 results
1473 /**
1474  * Create a proxy user using username of the effective user and the ugi of the
1475  * real user.
1476  * @param user
1477  * @param realUser
1478  * @return proxyUser ugi
1479  */
1480 @InterfaceAudience.Public
1481 @InterfaceStability.Evolving
1482 @
1483 public static UserGroupInformation createProxyUser(String user,
1484     UserGroupInformation realUser) {
1485     if (user == null || user.isEmpty()) {
1486         throw new IllegalArgumentException("Null user");
1487     }
1488     if (realUser == null) {
1489         throw new IllegalArgumentException("Null real user");
1490     }
1491     Subject subject = new Subject();
1492     Set<Principal> principals = subject.getPrincipals();
1493     principals.add(new User(user, AuthenticationMethod.PROXY, login: null));
1494     principals.add(new RealUser(realUser));
1495     return new UserGroupInformation(subject);
1496 }
```




HDFS授权详解-proxyuser

- hue/hive/sqoop/flume等服务的系统用户，使用代理机制时，需要进行配置；
- 相关的配置参数如下：
- `hadoop.proxyuser.xxx.groups`
- `hadoop.proxyuser.xxx.hosts`

Hive Proxy User Hosts

`hadoop.proxyuser.hive.hosts`

HDFS (Service-Wide)

*



Comma-delimited list of hosts where you want to allow the Hive user to impersonate other users. The default '*' allows all hosts. To disable entirely, use a string that doesn't correspond to a host name, such as '_no_host'.



Hive Proxy User Groups

`hadoop.proxyuser.hive.groups`

HDFS (Service-Wide)

*



Comma-delimited list of groups that you want to allow the Hive user to impersonate. The default '*' allows all groups. To disable entirely, use a string that doesn't correspond to a group name, such as '_no_group_'.



Hue Proxy User Hosts

`hadoop.proxyuser.hue.hosts`

HDFS (Service-Wide)

*



Comma-delimited list of hosts where you want to allow the Hue user to impersonate other users. The default '*' allows all hosts. To disable entirely, use a string that doesn't correspond to a host name, such as '_no_host'.



Hue Proxy User Groups

`hadoop.proxyuser.hue.groups`

HDFS (Service-Wide)

*



Comma-delimited list of groups that you want to allow the Hue user to impersonate. The default '*' allows all groups. To disable entirely, use a string that doesn't correspond to a group name, such as '_no_group_'.





大数据平台 CDH/HDP/CDP 与 TDH, HDFS 授权与鉴权的差异

- TDH 中，通过安全组件 Guardian 来管理各个组件的安全；
- Guardian 底层整合了 kerberos 和 ApacheDS来做认证；
- 同时Guardian 还通过 hdfs 插件机制，支持了对 hdfs 的目录和文件进行授权和鉴权，有点类似于 ranger 的插件机制；

TRANSWARP
GUARDIAN

[首页](#) [策略](#) [权限](#) [租户](#) [审计](#) [设置](#)

admin ▾

[< 返回](#)

下载keytab 重置密码 [锁定用户](#)

hive ● ACTIVE [编辑用户](#)

Email: 全名: Predefined user for inceptor

描述: Superuser for inceptor service

我的权限

hdfs1

☒ 显示继承权限 搜索

+ 添加权限

服务	路径	权限	操作
hdfs1	-	ACCESS	✎ ✕
hdfs1	/user/hundsun	READ, WRITE, EXECUTE, ADMIN	✎ ✕
hdfs1	/	READ, WRITE, EXECUTE, ADMIN	✎ ✕
hdfs1	/inceptor1/user/hive/warehouse	READ, WRITE, EXECUTE, ADMIN	✎ ✕
hdfs1	/inceptor1/tmp/hive	READ, WRITE, EXECUTE, ADMIN	✎ ✕



大数据平台 CDH/HDP/CDP 与 TDH, HDFS 授权与鉴权的差异

通过访问HDFS目录时的底层日志，我们可以大胆推测，guardian hdfs plug，和 ranger hdfs plugin 的实现细节，还是有所区别的：

Guardian是先进行hdfs posix 的权限校验，如果校验成功就允许执行对应的hdfs操作，只有在hdfs posix 权限校验失败时，才会查看 guardian 配置的权限；

如果guardian权限校验也是失败的，就阻止对应的hdfs 操作；如果guardian权限校验成功，就允许执行对应的hdfs操作；（底层是切换使用了 hdfs 超级用户来进行实际的 hdfs 操作）

```
...跳过
2021-10-14 17:04:09,391 INFO SecurityLogger.org.apache.hadoop.ipc.Server: Auth successful for hive/uf30-tdh2-regression@TDH (auth:KERBEROS)
2021-10-14 17:04:09,393 INFO SecurityLogger.org.apache.hadoop.security.authorize.ServiceAuthorizationManager: Authorization successful for hive/uf30-tdh3-regression@TDH (auth:PROXY) via hive/uf30-tdh2-regression@TDH (auth:KERBEROS) for protocol=interface org.apache.hadoop.hdfs.protocol.ClientProtocol
2021-10-14 17:04:09,415 INFO io.transwarp.guardian.plugins.hdfs.GuardianHdfsAuthorizer: Access denied during checking hdfs posix acl, trying Guardian permissions.
2021-10-14 17:04:16,480 INFO SecurityLogger.org.apache.hadoop.ipc.Server: Auth successful for hdfs/uf30-tdh1-regression@TDH (auth:KERBEROS)
2021-10-14 17:04:16,482 INFO SecurityLogger.org.apache.hadoop.security.authorize.ServiceAuthorizationManager: Authorization successful for hdfs/uf30-tdh1-regression@TDH (auth:KERBEROS) for protocol=interface org.apache.hadoop.hdfs.protocol.ClientProtocol
2021-10-14 17:04:26,535 INFO SecurityLogger.org.apache.hadoop.ipc.Server: Auth successful for hdfs/uf30-tdh1-regression@TDH (auth:KERBEROS)
```

```
2021-10-14 16:13:24,422 INFO SecurityLogger.org.apache.hadoop.ipc.Server: Auth successful for hive/uf30-tdh2-regression@TDH (auth:KERBEROS)
2021-10-14 16:13:24,423 INFO SecurityLogger.org.apache.hadoop.security.authorize.ServiceAuthorizationManager: Authorization successful for hive/uf30-tdh3-regression@TDH (auth:PROXY) via hive/uf30-tdh2-regression@TDH (auth:KERBEROS) for protocol=interface org.apache.hadoop.hdfs.protocol.ClientProtocol
2021-10-14 16:13:24,427 INFO io.transwarp.guardian.plugins.hdfs.GuardianHdfsAuthorizer: Access denied during checking hdfs posix acl, trying Guardian permissions.
2021-10-14 16:13:24,427 INFO org.apache.hadoop.ipc.Server: IPC Server handler 5 on 8020, call org.apache.hadoop.hdfs.protocol.ClientProtocol.rename2 from 10.20.159.114:60864 Call#3620 Retry#0: org.apache.hadoop.security.AccessControlException: Permission denied: user=hive, access=WRITE, inode="/user/hundsun/dap/hive/hs_tmp":dap:hadoop:drwxr-xr-x in the default POSIX ACLs, nor is granted permission in Guardian service
2021-10-14 16:13:24,429 INFO io.transwarp.guardian.plugins.hdfs.GuardianHdfsAuthorizer: Access denied during checking hdfs posix acl, trying Guardian permissions.
2021-10-14 16:13:24,429 INFO org.apache.hadoop.ipc.Server: IPC Server handler 8 on 8020, call org.apache.hadoop.hdfs.protocol.ClientProtocol.rename2 from 10.20.159.114:60864 Call#3623 Retry#0: org.apache.hadoop.security.AccessControlException: Permission denied: user=hive, access=WRITE, inode="/user/hundsun/dap/hive/hs_tmp":dap:hadoop:drwxr-xr-x in the default POSIX ACLs, nor is granted permission in Guardian service
2021-10-14 16:13:24,999 INFO SecurityLogger.org.apache.hadoop.ipc.Server: Auth successful for hdfs/uf30-tdh1-regression@TDH (auth:KERBEROS)
2021-10-14 16:13:25,000 INFO SecurityLogger.org.apache.hadoop.security.authorize.ServiceAuthorizationManager: Authorization successful for hdfs/uf30-tdh1-regression@TDH (auth:KERBEROS) for protocol=interface org.apache.hadoop.hdfs.protocol.ClientProtocol
```



大数据平台 CDH/HDP/CDP 与 TDH, HDFS 授权与鉴权的差异

通过查看反编译的 Guardian hdfs plugin 的源码，印证了我们上述推测：其底层使用了java 的 try-catch 捕获了 HDFS native 权限校验抛出的异常，并进一步通过guardian进行权限校验：

```
GuardianHdfsAuthorizer.class - Java Decompiler
File Edit Navigation Search Help

guardian-client-3.0-transwarp-6.0.2.jar guardian-common-3.0-transwarp-6.0.2.jar hdfs-plugin-3.0-transwarp-6.0.0-SNAPSHOT.jar

HdfsQuotaProps.class HdfsPermUtil.class GuardianHdfsAuthorizer.class

public void checkPermission(String fsOwner, String supergroup, UserGroupInformation callerUgi, INodeAttributes[] inodeAttrs, INode[] inodes, byte[][] pathByNameArr, int snapshotId, String path, int ancestorIndex, boolean doCheckOwner, boolean ancestorAccess, boolean parentAccess, boolean access, boolean subAccess) throws AccessControlException {
    62 AccessControlException posixException = null;
    64 String username = callerUgi.getShortUserName();
    67 if (isSuperAdmin(username)) {
        68 return;
    }
    72 if (!PermUtil.checkGlobalAccess(this.guardianClient, username)) {
    73 String errorMsg = "Access denied by Guardian Service, user " + username + " has no ACCESS permission to HDFS service " + this.component;
    75 throw new AccessControlException(errorMsg);
    }
    78 if (this.guardianConf.getBoolean(GuardianVars.GUARDIAN_HDFS_DEFAULT_AUTHORIZER_ENABLED.varname, GuardianVars.GUARDIAN_HDFS_DEFAULT_AUTHORIZER_ENABLED.defaultBoolVal)) {
    82 try {
        83 this.posixACLEnforcer.checkPermission(fsOwner, supergroup, callerUgi, inodeAttrs, inodes, pathByNameArr, snapshotId, path, ancestorIndex, doCheckOwner, ancestorAccess, parentAccess, access, subAccess);
        84 return;
    } catch (AccessControlException e) {
    85 posixException = e;
    86 LOG.info("Access denied during checking hdfs posix acl, trying Guardian permissions.");
    87 LOG.debug("Access denied during checking hdfs posix acl: ", (Throwable)e);
    }
    94 if (inodes == null) {
        95 return;
    }
    101 boolean grant = (check(username, inodes, ancestorIndex, ancestorAccess) && check(username, inodes, inodes.length - 2, parentAccess) && check(username, inodes, access) && check(username, inodes, subAccess));
    103 if (doCheckOwner) {
    104 checkOwner(username, inodes, inodes.length - 1);
    }
    107 if (!grant) {
    110 String errorMsg = (posixException == null) ? "Access denied by Guardian Service, permission should be granted in Guardian Service" : (posixException.getMessage() + " in the default POSIX ACLs, nor is granted by Guardian Service");
    111 throw new AccessControlException(errorMsg);
    }
}

private boolean check(String username, INode[] iNodes, FsAction access) {
    116 return check(username, iNodes, iNodes.length - 1, access);
}

private boolean check(String username, INode[] iNodes, int end, FsAction access) {
    120 if (access == null || end < 0) {
    121 return true;
    }
    122 for (FsAction action : fsActions) {
    123 if (access.implies(action)) {
    124 boolean grant = check(username, iNodes, end, action.name());
    125 if (!grant && !checkGlobalPerm(username, action.name())) {
        126 return false;
    }
    }
    }
}
```


HDFS授权详解-Best practices for HDFS authorization

权限管理最佳实践:

- 通过配置 umask 控制所有新创建的目录和文件的 posix mode, umask 默认022, 常用的还有 007, 077;
- 如果需要更改特定目录的权限, 应该遵循权限最小化原则, 按照 posix mode 对目录和文件的 owner/group/other 进行赋权;
- 如果需要对特定目录赋予特定用户的权限, 应该遵循权限最小化原则, 通过 posix acl 对 posix mode 进行补充;

当结合ranger hdfs 插件对hdfs进行权限管理时, ranger 推荐的最佳实践:

- Change HDFS umask to 077
- Identify directory which can be managed by Ranger policies
- Identify directories which need to be managed by HDFS native permissions
- Enable Ranger policy to audit all records



HDFS授权详解-相关参数

- `hadoop.security.authorization`: true/false: Is service-level authorization enabled?
- `dfs.permissions.enabled/dfs.permissions(deprecated)` = true: If yes use the permissions system as described here. If no, permission checking is turned off, but all other behavior is unchanged.
- `dfs.web.ugi` = webuser,webgroup
- `dfs.permissions.superusergroup` = supergroup: The name of the group of super-users.
- `fs.permissions.umask-mode` = 0022
- `dfs.cluster.administrators` = ACL-for-admins
- `dfs.namenode.acls.enabled` = true: Set to true to enable support for HDFS ACLs (Access Control Lists). By default, ACLs are enabled. When ACLs are disabled, the NameNode rejects all attempts to set an ACL.
- `dfs.namenode.posix.acl.inheritance.enabled`: Set to true to enable POSIX style ACL inheritance. Enabled by default. When it is enabled and the create request comes from a compatible client, the NameNode will apply default ACLs from the parent directory to the create mode and ignore the client umask. If no default ACL is found, it will apply the client umask.
- `hadoop.proxyuser.xxx.groups/hosts`

04

HIVE认证详解



HIVE认证详解

- HIVE 的认证方式，通过参数 `hive.server2.authentication` 在服务端进行统一配置；
- 该参数可选的值主要有三种：`hive.server2.authentication=none/kerberos/ldap`
- 客户端，不管是 beeline 等专用 cli 客户端，还是 dbeaver 等通用 jdbc gui 客户端，抑或 JAVA 应用（基于 jdbc），都需要根据服务端配置的认证方式，使用对应的方式，进行认证后才能成功连上 hiveserver2，进而提交查询命令。
- 视乎大数据集群中是否开启了 kerberos，实际的认证方式，分为以下四种：
 - 无认证模式/
 - 只开启LDAP认证模式/
 - 只开启Kerberos认证模式/
 - 开启Kerberos认证和LDAP认证模式，



HIVE认证详解-hive.server2.authentication = none

- 当不需要对用户身份进行校验，可以配置 `hive.server2.authentication = none`, 这种境况经常用在测试环境，生产环境一般不推荐;
- 此时用户通过各种客户端如 `cli/gui/java` 登录时，可以不配置用户名和密码, 在服务端 Hive 会认为登录的是匿名用户 `anonymous`, (这点不同于 `hdfs`, 当没有开启`kerberos`安全时，如果没有配置 环境变量或系统参数 `Hadoop_user_name`, `hdfs` 的默认用户是提交作业的`LINUX`系统用户) 如: `beeline -u jdbc:hive2://xx.xx.xx.xx:10000/default`
- 此时用户通过各种客户端如 `cli/gui/java` 登录时，也可以配置为任意用户名和任意密码, 在服务端 Hive 会认为登录的是用户声明的任意用户 (用户名可以是任意用户名，甚至是不存在的用户名；密码可以是任意密码，或不配置密码)，如: `beeline -u jdbc:hive2://xx.xx.xx.xx:10000/default -n xyz`; `beeline -u jdbc:hive2://xx.xx.xx.xx:10000/default -n xyz -p xxxx`
- 可以通过 `hiveserver2 webui`，验证登录的用户身份;

Active Sessions

User Name	IP Address	Operation Count	Active Time (s)
xyz	10.20.39.41	0	2728
anonymous	10.23.10.20	0	78
anonymous	10.23.10.20	0	78
Total number of sessions: 3			



HIVE认证详解-hive.server2.authentication = ldap

- 中大型企业中一般都会有用户身份的统一认证平台，其底层一般都使用 ldap 协议，其具体实现有微软的 ActiveDirectory，也有 openLdap, ApacheDS等开源实现；
- Hive 提供了基于 Ldap 的认证机制，可以使用企业的统一认证平台，来验证登录hive的用户身份,其配置方式：`hive.server2.authentication = ldap`;
- 具体的 ldap 工具的 url，需要通过参数指定：`hive.server2.authentication.ldap.url`;
- 除了集成商业版的 ActiveDirectory，大数据集群中也可以使用独立安装的开源的ldap工具，此类工具常见的有 openLdap 和 ApacheDS，其中前者在大部分linux发行版中都自带了package安装包，更容易安装，不过主要通过命令行cli进行管理；而后者则自带了gui客户端 Apache Directory Studio，功能更为丰富；以 openLdap 为例，其安装命令如下：`sudo yum -y install openldap-clients`;`sudo yum -y install openldap`;
- 客户端登录 ldap 认证的 hiveserver2 时，需要提供用户名和密码，hiveserver2 会到ldap中验证用户名和密码，只有验证通过后才能正常登录；
- 以 beeline 登录为例，其命令格式如下：`beeline -u jdbc:hive2://xx.xx.xx.xx:10000/default -n hs_cic -p xyzabc`;



HIVE认证详解- kerberos 环境下的认证

- 大数据生态中的各种存储系统，如HDFS/hive/hbase/zookeeper/kafka等，都支持开启Kerberos安全认证；
- 当大数据集群中的存储系统如HDFS/hive/hbase/zookeeper/kafka等开启了kerberos安全认证后，访问这些存储系统的客户端，包含各种计算引擎如 hive/hbase/spark/flink 的系统服务，和用户编写的各种应用如 spark/hive/flink等，都需要经过 kerberos kdc 的认证获得了 ticket 凭证后，才能与这些存储系统进行正常交互；
- 具体到 hiveserver2，其在跟开启了 kerberos 安全认证的 hdfs/yarn/hbase 等交互时，同样需要配置使用相应的 kerberos principal(一般配置为hive)，且只有在经过 kdc 验证获得 ticket 后，才能与 hdfs/yarn/zk 进行交互,hive-site.xml中，相关配置项截图如下：

```
<property>
  <name>hive.server2.authentication</name>
  <value>kerberos</value>
</property>
<property>
  <name>hive.metastore.kerberos.principal</name>
  <value>hive/_HOST@CDH.COM</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.principal</name>
  <value>hive/_HOST@CDH.COM</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.keytab</name>
  <value>hive.keytab</value>
</property>
<!-- 'hive.server2.webui.use.spnego', originally set to 'true' (non-final), is overridden below by a safety valve-->
<property>
  <name>hive.server2.webui.spnego.keytab</name>
  <value>hive.keytab</value>
</property>
```



HIVE认证详解- kerberos 环境下的认证

- 在开启了 kerberos 安全认证的大数据集群环境中，HIVE既可以配置使用 kerberos 认证机制，也可以配置使用 LDAP 认证机制：hive.server2.authentication = kerberos/ldap，分别对应上文所说的，只开启Kerberos 认证模式 和 开启Kerberos认证和LDAP认证模式
- 配置hive.server2.authentication = kerberos，即要求 hiveserver2 的各种客户端如 cli/gui/java jdbc，只有在通过 kerberos 认证获得ticket 后，才能正常登陆 hiveserver2 进而提交 sql；
- 配置hive.server2.authentication = ldap，即要求 hiveserver2 的各种客户端如 cli/gui/java jdbc，需要提供用户名和密码，且hiveserver2 会到ldap中验证用户名和密码，只有验证通过后，才能正常登陆 hiveserver2 进而提交 sql,其配置方式是；（当然因为整个大数据环境开启了 kerberos, 所以在登录hiveserver2之前，一样要经过 kerberos kdc 的认证）
- 在强调数据安全的金融行业，我们推荐使用这两种方式



HIVE认证详解- kerberos 环境下, hive.server2.authentication = kerberos

- 由于是在kerberos环境下，所以客户端在登录前，需要首先从 kdc 获取 ticket 并维护在 ticket cache中： a valid Kerberos ticket in the ticket cache before connecting;
- 如果是 cli/beeline 等客户端，一般会通过命令 kinit，基于手工输入的密码或keytab 文件，来获取特定业务用户的 ticket，并存储在客户端的 ticket cache中；（如果缓存的 ticket 过期了，需要重新获取）；
- 如果是程序代码，则一般通过 `org.apache.hadoop.security.UserGroupInformation.loginUserFromKeytab(String user, String path)` 的方式，基于keytab文件来获取特定业务用户的 ticket，并存储在客户端的 ticket cache中；（UserGroupInformation 在后台会自动基于keytab 文件来定时刷新ticket，确保不会过期）；
- 客户端在获取业务用户的 ticket 成功后，才可以通过 jdbc连接，登录到指定的 hiveserver2;

HIVE认证详解- kerberos 环境下, hive.server2.authentication = kerberos

客户端在获取业务用户的 ticket 成功后, 通过 jdbc连接登录到指定的 hiveserver2时, 需要特别注意下 hiveserver2 的url的格式, 其格式推荐使用:

jdbc:hive2://xx.xx.xx.xx:10000/default;principal=hive/_HOST@CDH.COM:

- 这里的principal部分, 推荐使用三段式来指定, 包含principal, host 和 realm;
- principal 必须指定为系统用户hive, 而不能是业务用户如 dap,xyz等 (本质上是因为, hive-site.xml 中配置的hive系统用户是hive) ;
- host部分, 推荐指定为 HOST, 此时在底层使用时会替换为 hiveserver2 节点的hostname (当然也可以直接指定为 hiveserver2 节点的具体的 hostname) ;
- realm 部分, 需要根据实际配置情况进行指定 (可以查看配置文件 /etc/krb5.conf) ;



HIVE认证详解- kerberos 环境下, hive.server2.authentication = ldap

- 由于是在kerberos环境下，所以客户端在登录前，需要首先从 kdc 获取 ticket 并维护在 ticket cache中，这一点跟 kerberos 环境下，hive 的 kerberos 认证方式时一直的：a valid Kerberos ticket in the ticket cache before connecting;
- 如果是 cli/beeline 等客户端，一般会通过命令 kinit，基于手工输入的密码或keytab 文件，来获取特定业务用户的 ticket，并存储在客户端的 ticket cache中；（如果缓存的 ticket 过期了，需要重新获取）；
- 如果是程序代码，则一般通过 `org.apache.hadoop.security.UserGroupInformation.loginUserFromKeytab(String user, String path)` 的方式，基于keytab文件来获取特定业务用户的 ticket，并存储在客户端的 ticket cache中；（UserGroupInformation 在后台会自动基于keytab 文件来定时刷新ticket，确保不会过期）；
- 客户端在获取业务用户的 ticket 成功后，才可以通过 jdbc连接，登录到指定的 hiveserver2，此时登录格式，跟非 kerberos 环境下，hive 的 ldap认证方式，是一样的；



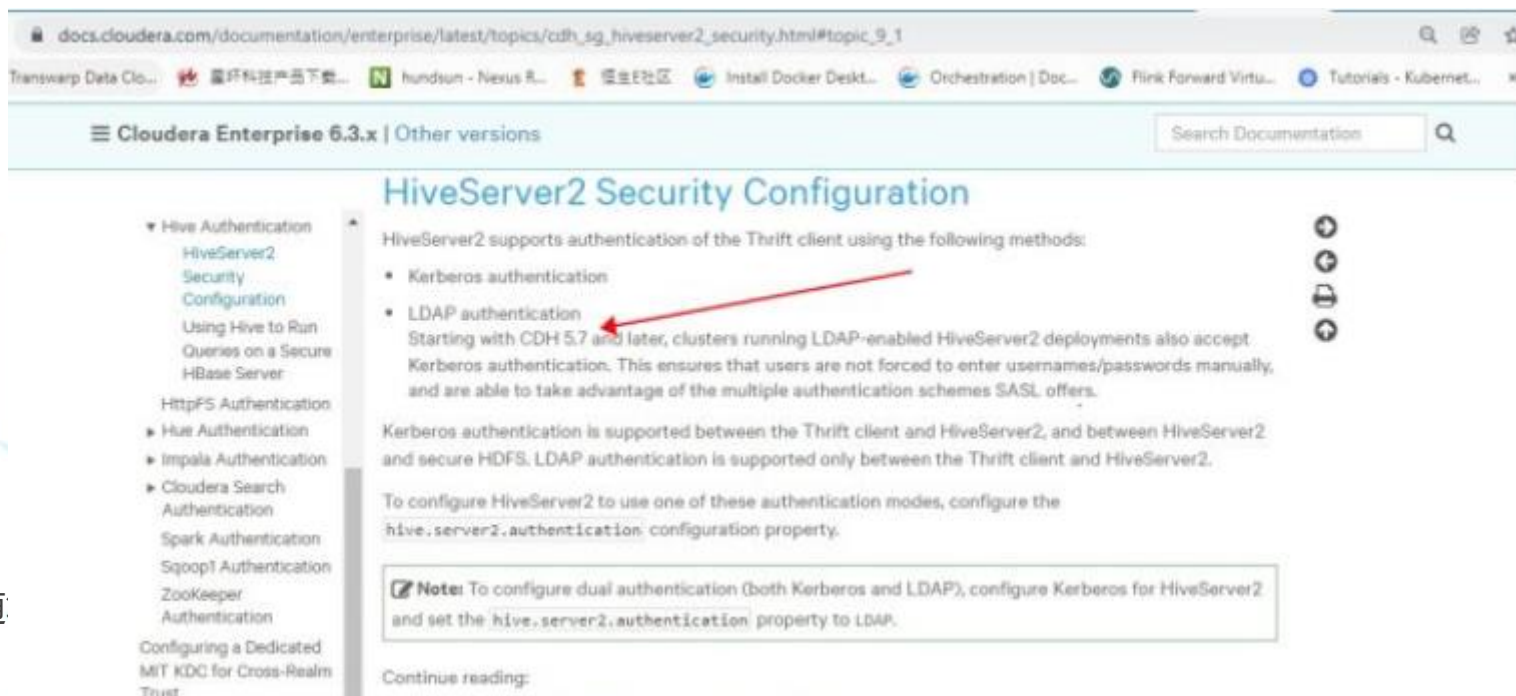
HIVE认证详解- kerberos 环境下, hive.server2.authentication = ldap

- 客户端在获取业务用户的 ticket 成功后, 通过 jdbc连接登录到指定的 hiveserver2时, 跟非 kerberos 环境下的 ldap认证方式的登录格式, 是一样的:
- 此时需要提供用户名和密码, hiveserver2 会到ldap服务器中验证用户名和密码, 只有验证通过后才能正常登录;
- 以 beeline 登录为例, 其命令格式如下: `beeline -u jdbc:hive2://xx.xx.xx.xx:10000/default -n hs_cic -p xyzabc;`



大数据平台 CDH/TDH/CDP 与 TDH 中, hive 认证方式的差异

- 客户端的登录方式, 取决于服务端的具体配置;
- CDH/TDH/CDP中, 在 CDH 5.7 及以后的版本中, Cloudera 对hive的安全认证进行了增强: 在大数据集群开启了 kerberos 安全认证的环境下, 即使 hive 服务端配置了使用ldap (hive.server2.authentication = ldap), 客户端也可以通过url指定使用 KERBEROS 认证方式来登录; 此时实际的业务用户, 是登录前, 通过 kinit指定的业务用户!!! 此时需要注意, url中需要指定principal=HIVE/_HOST@CDH.COM,以示默认的 ldap认证方式下, 实际使用的是kerberos认证方式!!!
- 在TDH环境下, 在大数据集群开启了 kerberos 安全认证的环境下, 如果 hive 服务端配置了使用ldap (hive.server2.authentication = ldap), 则必须通过kerberos和ldap的双重认证后, 才能登陆 hiveserver2;





大数据平台 CDH/TDH/CDP 与 TDH 中, hive 认证方式的差异

- TDH 中, 通过安全组件 Guardian 来管理各个组件的安全, Guardian 底层整合了 kerberos 和 ApacheDS;
- TDH中, 同样支持以上四种认证方式, 其推荐的hive认证方式, 其实等同于 “kerberos环境下, hive 的 LDAP 认证方式: hive.server2.authentication = ldap” ;
- 在TDH 中配置 inceptor 使用 “开启Kerberos认证和LDAP认证模式”, 并不需要额外安装 OpenLdap/ApacheDS/microsoft AD 等 ldap的具体实现, 也不需要跟企业内部统一的 LDAP 服务器打通, 也不存在泄露企业域账号用户名和密码的风险, 因为底层实际使用的是 Guardian 底层自带的一个 ldap实现 (本质是ApacheDS), 创建用户更改密码等操作都是在 Guardian 中操作的, 更企业域账户是独立的;

Inceptor使用手册

8. Inceptor外部工具连接手册

9. Inceptor运维手册

9.1. Inceptor的配置

9.1.1. 安装时配置LDAP认证

9.1.2. 安装时配置Kerberos认证

9.1.3. 认证模式切换

9.2. 多Inceptor的安装配置

9.3. Inceptor 资源调度器

附录 A: Inceptor字段规范

集群未安装Guardian时, 只支持简单用户认证模式 (不需要安装和Kerberos认证; 另外对Inceptor还支持LDAP用户认证。

在开启LDAP或Kerberos认证之前, 您必须先为集群安装Guardian。

Inceptor服务的认证包括两部分: 用户连接Inceptor和Inceptor连接其他组件 (如 Hive、HBase、HDFS 等) 需要密码)、LDAP认证和Kerberos认证三种认证方式; Inceptor服务安装时配置, 也可以选择安装后再切换认证模式, 共有以下四种认证模式:

1. 无认证模式: 用户连接Inceptor服务时及Inceptor与其他组件交互时, 均不需要认证。

2. 只开启LDAP认证模式: 用户连接Inceptor服务时使用LDAP认证, Inceptor连接其他组件时, 需使用LDAP认证的CLI登录方式。

3. 只开启Kerberos认证模式(推荐, 更加安全): 用户连接Inceptor服务时, 需使用Kerberos认证的CLI登录方式; Inceptor连接其他组件时, 需使用Kerberos认证的CLI登录方式。

4. 开启Kerberos认证和LDAP认证模式: 用户连接Inceptor服务时, 需使用Kerberos认证; 您通过命令行连接Inceptor服务时, 需使用LDAP认证。

9.1.1. 安装时配置LDAP认证

TRANSWARP

服务 管理 应用市场

Guardian HEALTHY

home > Guardian

角色: 1-10 / 12

角色名称	节点名称
<input type="checkbox"/> TxSQL Server (Guardian,tdh-01)	tdh-01
<input type="checkbox"/> TxSQL Server (Guardian,tdh-02)	tdh-02
<input type="checkbox"/> TxSQL Server (Guardian,tdh-03)	tdh-03
<input type="checkbox"/> Guardian Federation Service (Guardian,tdh-01)	tdh-01
<input type="checkbox"/> Guardian Federation Service (Guardian,tdh-02)	tdh-02
<input type="checkbox"/> Guardian ApacheDS (Guardian,tdh-01)	tdh-01
<input type="checkbox"/> Guardian ApacheDS (Guardian,tdh-02)	tdh-02



HIVE认证详解-相关参数

hive-site.xml中，相关参数有：

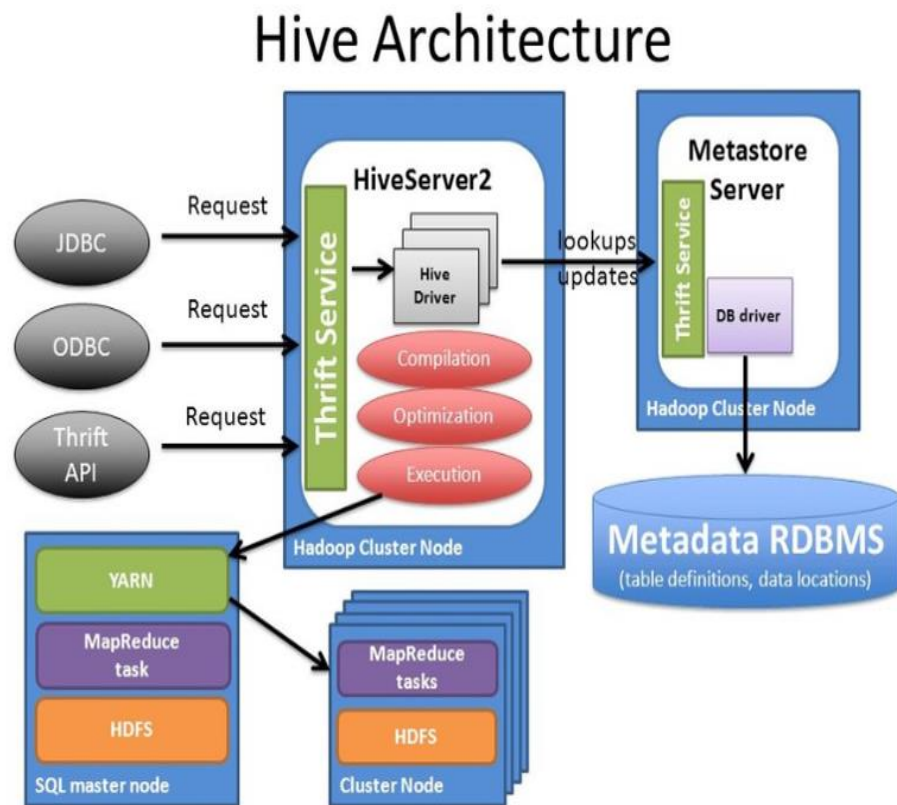
- hive.server2.authentication
- hive.server2.authentication.kerberos.keytab
- hive.server2.authentication.kerberos.principal
- hive.server2.authentication.spnego.keytab
- hive.server2.authentication.spnego.principal
- hive.server2.authentication.ldap.url
- hive.server2.authentication.ldap.baseDN
- hive.server2.authentication.ldap.Domain
- hive.metastore.kerberos.keytab.file
- hive.metastore.kerberos.principal
- hive.server2.enable.doAs/hive.server2.enable.impersnation

05

HIVE授权详解

HIVE授权详解-HIVE架构和使用方式回顾

先来回顾下 hive 的整体架构，分为客户端和服务端两部分；
服务端又分为 hiveserver2 和 hms，以及底层的元数据持久存储 metastore db；
Hive 又作为 hadoop的客户端，访问底层的 hdfs 和 yarn；





HIVE授权详解-HIVE架构和使用方式回顾

当前市面上HIVE的使用方式分两种：

- 只使用 HMS，将hive 看做是单独的 table format layer，比如 spark/flink on hive: Hive as a table storage layer: These users have direct access to HDFS and the metastore server (which provides an API for metadata access);
- 完整地使用 hive，包括 hiveserver2+ hms, 比如 hive on mr/tez/spark: Hive as a SQL query engine: These users have all data/metadata access happening through HiveServer2. They don't have direct access to HDFS or the metastore.
- 第一种只使用 hms 的方式下，spark 等应用会直接访问 hdfs, 所以授权依赖于 hdfs 的 authentication 机制: HDFS access is authorized through the use of HDFS permissions. (Metadata access needs to be authorized using Hive configuration.)
- 第二种使用 hiveserver2 的方式，用户通过 hiveserver2 进而访问 hms 和 hdfs/s3，hive 更容易统一管控用户权限，也更容易组织底层文件的layout和存储格式，比如支持 orc acid事务表，从而提供高效的数据服务，是Hive 社区期望的方式；



HIVE授权详解-四种authentication机制

HIVE 共有四种 authentication 机制:

- Storage Based Authorization in the Metastore Server
- SQL Standards Based Authorization in HiveServer2
- Authorization using Apache Ranger & Sentry
- Old default Hive Authorization (Legacy Mode)

HIVE授权详解-四种authentication机制

之所以有这么多种 authentication 机制，从根本上来说，还是因为 hive 历史发展的原因：

- 早期的 hive 仅仅是一个简单的存储引擎之上的SQL解析与执行层，对底层存储引擎中的数据没有完整的掌控力，允许外部应用如 spark/presto/flink 等，直接访问 hms获取元数据，并进而读写底层的数据（schema on read）；
- 后续hive 逐渐迭代，加强了对底层存储引擎中的数据的管控，包括文件的目录结构文件名称存储格式等，并进而支持了 orc acid 事务表等高效的数据存取格式，越来越像一款类似 mysql/oracle 等的数据库，推荐用户使用 hiveserver2 来访问底层数据；
- 所以 hive 需要支持外部计算引擎直接使用 hms 与使用 hiveserver2两种方式，需要支持 orc acid 事务表，也需要支持其它格式的各种内部表和外部表

This RDBMS style of authorization is not very suitable for the typical use cases in Hadoop because of the following differences in implementation:

1. Unlike a traditional RDBMS, Hive is not in complete control of all data underneath it. The data is stored in a number of files, and the file system has an independent authorization system.
2. Also unlike a traditional RDBMS which doesn't allow other programs to access the data directly, people tend to use other applications that read or write directly into files or directories that get used with Hive.

This creates problem scenarios like:

1. You grant permissions to a user, but the user can't access the database or file system because they don't have file system permissions.
2. You remove permissions for a user, but the user can still access the data directly through the file system, because they have file system permissions.



• HIVE授权详解-Old default Hive Authorization (Legacy Mode)

Hive Old Default Authorization 是 hive 2.0 之前默认的 authorization model, 支持类似 RDBMS 中对 user/group/roles 赋予 database/table 各种权限的机制: authorization based on users, groups and roles and granting them permissions to do operations on database or table

但是 Hive Old Default Authorization 并不是一个完整的权限控制模型: leaving many security gaps unaddressed, for example, the permissions needed to grant privileges for a user are not defined, and any user can grant themselves access to a table or database.

hive 2.0 之后默认的 authorization model 已经被切换为SQL standards based authorization mode (HIVE-12429);

Status of Hive Authorization before Hive 0.13

The default authorization in Hive is not designed with the intent to protect against malicious users accessing data they should not be accessing. It only helps in preventing users from accidentally doing operations they are not supposed to do. It is also incomplete because it does not have authorization checks for many operations including the grant statement. The authorization checks happen during Hive query compilation. But as the user is allowed to execute dfs commands, user-defined functions and shell commands, it is possible to bypass the client security checks.

Hive also has support for storage based authorization, which is commonly used to add authorization to metastore server API calls (see [Storage Based Authorization in the Metastore Server](#)). As of Hive 0.12.0 it can be used on the client side as well. While it can protect the metastore against changes by malicious users, it does not support fine grained access control (column or row level).

The default authorization model in Hive can be used to provide fine grained access control by creating views and granting access to views instead of the underlying tables.



HIVE授权详解- Storage Based Authorization in the Metastore Server-必要性

- HMS 提供了对 hive metastore db 中的元数据的访问，为保护这些元数据被各种 hms 客户端，如 spark/presto/flink，错误地访问和修改，HMS 不能完全依赖这些客户端自身的认证和授权等安全机制，为此 Hive 0.10 通过 HIVE-3705 增加了对 HMS 的 authorization 能力，即 Storage Based Authorization;
- Storage Based Authorization 底层依赖 hdfs permissions 作为 source of truth: it uses the file system permissions for folders corresponding to the different metadata objects as the source of truth for the authorization policy.

Storage-System Based Authorization Model

The Hive community realizes that there might not be a one-size-fits-all authorization model, so it has support for alternative authorization models to be plugged in.

In the HCatalog package, we have introduced implementation of an authorization interface that uses the permissions of the underlying file system (or in general, the storage backend) as the basis of permissions on each database, table or partition.

✓ Note

This feature is also available in Hive on the metastore-side, starting with release 0.10.0 (see [Storage Based Authorization in the Metastore Server](#) in the Hive documentation). Starting in Hive 0.12.0 it also runs on the client side ([HIVE-5048](#) and [HIVE-5402](#)).

In Hive, when a file system is used for storage, there is a directory corresponding to a database or a table. With this authorization model, the read/write permissions a user or group has for this directory determine the permissions a user has on the database or table. In the case of other storage systems such as HBase, the authorization of equivalent entities in the system will be done using the system's authorization mechanism to determine the permissions in Hive.

For example, an alter table operation would check if the user has permissions on the table directory before allowing the operation, even if it might not change anything on the file system.

A user would need write access to the corresponding entity on the storage system to do any type of action that can modify the state of the database or table. The user needs read access to be able to do any non-modifying action on the database or table.

When the database or table is backed by a file system that has a Unix/POSIX-style permissions model (like HDFS), there are read(r) and write(w) permissions you can set for the owner user, group and 'other'.

Details of HDFS permissions are given at http://hadoop.apache.org/docs/rx.x.x/hdfs_permissions_guide.html, for example:

- [HDFS Permissions Guide \(release 1.0.4\)](#)
- [HDFS Permissions Guide \(release 1.2.1\)](#)

Note: Support for HDFS ACL (introduced in Apache Hadoop 2.4) is not available in the released versions of Hive. Which means, that it checks only the traditional rwx style permissions to determine if a user can write to the file system. The support for ACL is available in Hive trunk HIVE-7583, which will be available in Hive 0.14.



HIVE授权详解- Storage Based Authorization in the Metastore Server-实现细节

- Storage Based Authorization 提供了客户端直接访问 HMS 时，对底层元数据的保护：To control metadata access on the metadata objects such as Databases, Tables and Partitions, it checks if you have permission on corresponding directories on the file system;
- Storage Based Authorization 通过代理机制 (hive.server2.enable.doAs =true) ，也可以提供对客户端通过 hiveserver2 访问HIVE数据时，对底层元数据和数据的保护：You can also protect access through HiveServer2 by ensuring that the queries run as the end user;
- Storage Based Authorization 可以通过 hdfs acl 提供权限的灵活性：Through the use of HDF ACL, you have a lot of flexibility in controlling access to the file system, which in turn provides more flexibility with Storage Based Authorization;



HIVE授权详解- Storage Based Authorization in the Metastore Server-局限性

由于Storage based authorization 的底层原理是依赖用户对底层存储系统中数据的访问权限，且该用户在 hiveserver2 开启代理与不开启代理机制下身份不同，所以其主要用来在用户直接访问 hms 时，提供对底层元数据的保护；

对于用户使用 hiveserver2 的情况，需要限制 HiveServer2 中可以执行的操作，此时不能单纯依靠 Storage based authorization，还需要配合 “SQL Standards Based Authorization” 或 “Authorization using Apache Ranger & Sentry，或者配置使用 FallbackHiveAuthorizer：

Fall Back Authorizer

You need to use Hive 2.3.4 or 3.1.1 or later to use Fall Back Authorizer.

Admin needs to specify the following entries in hiveserver2-site.xml:

```
<property>
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hive.security.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.plugin.fallback.FallbackHiveAuthorizerFactory</value>
</property>
```

FallbackHiveAuthorizerFactory will do the following to mitigate above mentioned threat:

1. Disallow local file location in sql statements except for admin
2. Allow "set" only selected whitelist parameters
3. Disallow dfs commands except for admin
4. Disallow "ADD JAR" statement
5. Disallow "COMPILE" statement
6. Disallow "TRANSFORM" statement



HIVE授权详解-SQL Standards Based Authorization in HiveServer2

- Storage Based Authorization 只能基于文件系统的目录/文件的权限管理机制提供database/table/partition粒度的权限管控，为提供更细粒度的权限管控，比如行级别，列级别，视图级别的权限管控，Hive 0.13.0 通过 HIVE-5837 引入了SQL Standards Based Authorization；
- SQL Standards Based Authorization，其作用域是 HiveServer2，可以跟/需要跟 hms 的 storage based authorization 结合使用，以提供对HIVE元数据和数据的全面的安全管控；

SQL Standards Based Hive Authorization (New in Hive 0.13)

The SQL standards based authorization option (introduced in Hive 0.13) provides a third option for authorization in Hive. This is recommended because it allows Hive to be fully SQL compliant in its authorization model without causing backward compatibility issues for current users. As users migrate to this more secure model, the current default authorization could be deprecated.

For an overview of this authorization option, see [SQL Standards Based Authorization in HiveServer2](#).

This authorization mode can be used in conjunction with storage based authorization on the metastore server. Like the current default authorization in Hive, this will also be enforced at query compilation time. To provide security through this option, the client will have to be secured. This can be done by allowing users access only through Hive Server2, and by restricting the user code and non-SQL commands that can be run. The checks will happen against the user who submits the request, but the query will run as the Hive server user. The directories and files for input data would have read access for this Hive server user. For users who don't have the need to protect against malicious users, this could potentially be supported through the Hive command line as well.

The goal of this work has been to comply with the SQL standard as far as possible, but there are deviations from the standard in the implementation. Some deviations were made to make it easier for existing Hive users to migrate to this authorization model, and some were made considering ease of use (in such cases we also looked at what many widely used databases do).

Under this authorization model, users who have access to the Hive CLI, HDFS commands, Pig command line, 'hadoop jar' command, etc., are considered privileged users. In an organization, it is typically only the teams that work on ETL workloads that need such access. **These tools don't access the data through HiveServer2, and as a result their access is not authorized through this model. For Hive CLI, Pig, and MapReduce users access to Hive tables can be controlled using storage based authorization enabled on the metastore server.**

Most users such as business analysts tend to use SQL and ODBC/JDBC through HiveServer2 and their access can be controlled using this authorization model.



HIVE授权详解-SQL Standards Based Authorization in HiveServer2

- SQL Standards Based Authorization, 会在 hiveserver2 解析编译用户的 sql query 语句时, 校验提交该 sql query 语句的业务用户的权限, 但是在底层执行该 query 时, 使用的是 Hive 系统用户, 所以 hive 系统用户需要有对底层数据对应的目录/文件的访问权限;
- 由于 SQL Standards Based Authorization 的作用域是 HiveServer2, 所以对于直接访问底层hdfs数据的用户, 比如 spark on hive/Hive CLI/hadoop jar命令等, 需要依赖 hms 的 storage based authorization 来进行安全管控;

SQL Standards Based Hive Authorization (New in Hive 0.13)

The SQL standards based authorization option (introduced in Hive 0.13) provides a third option for authorization in Hive. This is recommended because it allows Hive to be fully SQL compliant in its authorization model without causing backward compatibility issues for current users. As users migrate to this more secure model, the current default authorization could be deprecated.

For an overview of this authorization option, see [SQL Standards Based Authorization in HiveServer2](#).

This authorization mode can be used in conjunction with storage based authorization on the metastore server. Like the current default authorization in Hive, this will also be enforced at query compilation time. To provide security through this option, the client will have to be secured. This can be done by allowing users access only through Hive Server2, and by restricting the user code and non-SQL commands that can be run. The checks will happen against the user who submits the request, but the query will run as the Hive server user. The directories and files for input data would have read access for this Hive server user. For users who don't have the need to protect against malicious users, this could potentially be supported through the Hive command line as well.

The goal of this work has been to comply with the SQL standard as far as possible, but there are deviations from the standard in the implementation. Some deviations were made to make it easier for existing Hive users to migrate to this authorization model, and some were made considering ease of use (in such cases we also looked at what many widely used databases do).

Under this authorization model, users who have access to the Hive CLI, HDFS commands, Pig command line, 'hadoop jar' command, etc., are considered privileged users. In an organization, it is typically only the teams that work on ETL workloads that need such access. These tools don't access the data through HiveServer2, and as a result their access is not authorized through this model. For Hive CLI, Pig, and MapReduce users access to Hive tables can be controlled using storage based authorization enabled on the metastore server.

Most users such as business analysts tend to use SQL and ODBC/JDBC through HiveServer2 and their access can be controlled using this authorization model.



HIVE授权详解-SQL Standards Based Authorization in HiveServer2

- SQL Standards Based Authorization, 对 hiveserver2 中能提交的命令, 做了限制;
- 启用该 authorization 机制时, Dfs/add/delete/compile/reset 等命令时被禁用的;
- 使用白名单机制, 通过参数 `hive.security.authorization.sqlstd.confwhitelist`, 限制了可以通过 set 覆盖哪些参数配置;

Restrictions on Hive Commands and Statements



Commands such as `dfs`, `add`, `delete`, `compile`, and `reset` are disabled when this authorization is enabled.

The `set` commands used to change Hive configuration are restricted to a smaller safe set. This is controlled using the `hive.security.authorization.sqlstd.confwhitelist` configuration parameter. If this set needs to be customized, the HiveServer2 administrator can set a value for this configuration parameter in its `hive-site.xml`.

Privileges to add or drop functions and macros are restricted to the **admin** role.

To enable users to use functions, the ability to create `permanent functions` has been added. A user in the **admin** role can run commands to create these functions, which all users can then use.

The Hive `transform clause` is also disabled when this authorization is enabled.



HIVE授权详解-Authorization using Apache Ranger & Sentry

- Apache Ranger 和 Apache Sentry 通过插件机制，提供了对 hive authorization的支持，Sentry和Ranger这两个框架非常类似，都是基于角色的访问控制模型（role-based access control, RBAC），基于角色的访问控制在需要对大量用户进行授权时能大大减轻所需的ACL配置工作；
- Sentry最初是由Cloudera公司内部开发而来的，初衷是为了让用户能够细粒度的控制Hadoop系统中的数据（主要指HDFS，Hive的数据），所以Sentry对HDFS，Hive以及同样由Cloudera开发的Impala有着很好的支持性；
- 而Ranger最初是由另一家公司Hortonworks（已经被Cloudera收购）主导开发的，它同样是做细粒度的权限控制，但相比较于Sentry而言，ranger有着更加丰富的策略控制，以及更加通用性的大数据组件支持，包括于HDFS, Hive, HBase, Yarn, Storm, Knox, Kafka, Solr 和 NiFi等在Cloudera公司新推出的大数据平台cdp中，不在提供对sentry的支持而是统一使用了Ranger；
- CDH中默认使用的是 Sentry， HDP 中默认使用的是 Ranger, 由于 sentry 项目已经从 asf 中退役了，目前业界推荐的，CDP中默认的，都是 Ranger；

授权

- 集中式平台，可跨Hadoop生态系统一致地定义和管理安全策略
- HDFS, Hive, HBase, YARN, Kafka, Solr, Storm, Knox, NiFi, Atlas, Impala, HMS
- 具有自定义策略条件和上下文丰富程序的可扩展体系结构
- 轻松添加新的组件类型以进行授权

密钥管理

- 存储和管理加密密钥策略和生命周期
- 支持HDFS透明数据加密
- 与HSM集成
- Safenet (LUNA, KeySecure)

审计

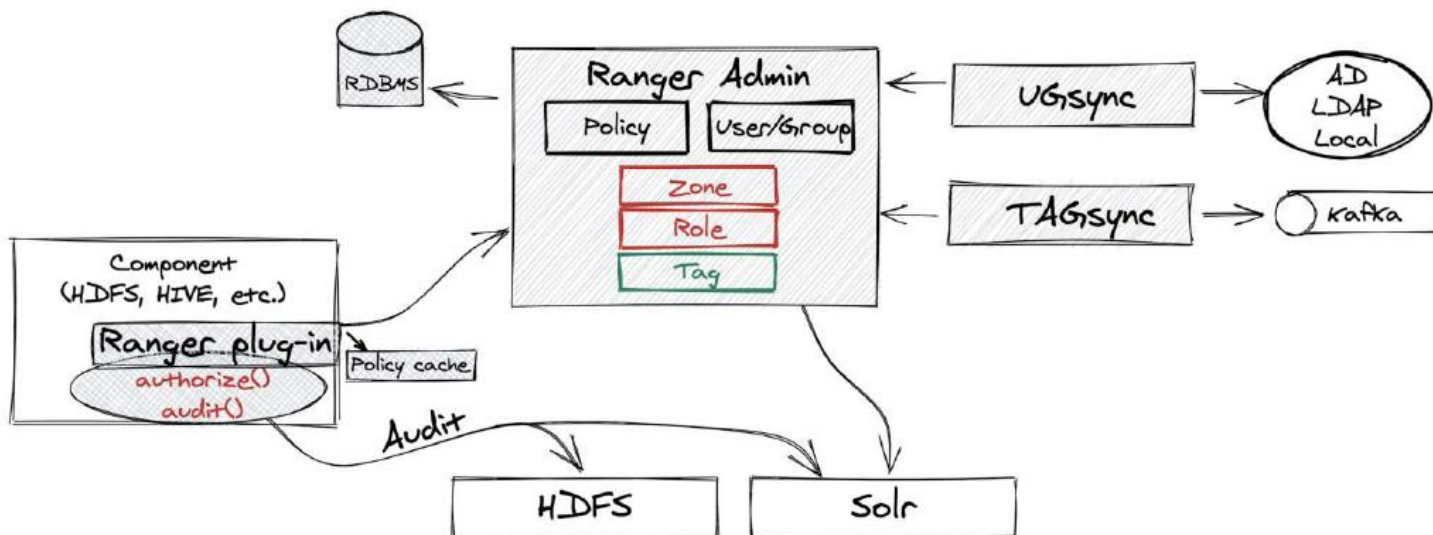
- 所有访问请求的中央审计位置
- 支持多个目标源（HDFS，Solr等）
- 实时视觉查询界面



HIVE授权详解-Authorization using Apache Ranger & Sentry

- Ranger 作为集中式平台，对 Hadoop 生态中的大多数组件，包括hdfs/hive/hbase/yarn/kafka等，综合提供了授权，密钥管理和审计功能；
- Ranger 采用了 ABAC 模型（Attribute based access control），基于标签策略，根据分类（标记）控制对资源的访问；
- Ranger 提供了很多高级特性，如 web ui直观和细粒度的策略查看和管理，全面的可扩展的审核日志记录，动态的行和列级别的访问控制，动态数据掩码可实时保护敏感数据等。

RANGER ADMIN 架构





HIVE授权详解-总结

- HIVE的权限管控机制，最常用的是结合 Storage Based Authorization in the Metastore Server 和 SQL Standards Based Authorization in HiveServer2 或 sentry/ranger 插件；
- 其中前者基于底层文件系统的权限管理机制，提供对 hms API 的权限管控，比如查看/新增/删除表或分区等；
- 后者提供通过 hiveserver2 访问hive数据时，更细粒度的权限管控，比如行级别，列级别，视图级别的权限管控；
- 当使用 spark/flink/presto on hive 方案时，应用直接访问 hms 和 hdfs，只有 Storage Based Authorization起作用，主要依赖的是 hdfs 的权限管控机制 (posix mode + posix acl)
- 当使用 hive on mr/tez/spark 方案时，应用通过 hiveserver2 访问 hms 和 hdfs，Storage Based Authorization 和 sentry/ranger 都起作用（需要关闭代理功能 hive.server.enable.doAs=false）



HIVE授权详解-Hive.server2.enableDoas

- 终端业务用户比如 xyz 提交给 HIVESERVER2 的 SQL作业，经过 HIVESERVER2 的解析编译和优化后，一般会生成 MR/TEZ/SPARK 任务（之所以说一般，是因为有的 SQL 是直接HIVESERVER2中执行的，不会生成分布式的 MR/TEZ/SPARK 任务），这些 MR/TEZ/SPARK 任务最终访问底层的基础设施 HDFS 和 YARN 时，一样要经过这些基础设施 hdfs/yarn的权限校验；
- 那么这些底层基础设施 hdfs/yarn 进行权限校验时，是针对 hive 系统用户进行校验（hiveserver2 这个服务的系统用户一般是linux操作系统上的用户 hive），还是针对终端业务用户比如 hundsun进行校验呢？
- 这点可以通过参数 hive.server2.enable.doAs进行控制（老版本参数是hive.server2.enable.impersonation）
- hive.server2.enable.doAs=false/TRUE: “Setting this property to true will have HiveServer2 execute Hive operations as the user making the calls to it.”；

```
58 public class SecurityUtils {
59     private static final Logger LOG = LoggerFactory.getLogger(SecurityUtils.class);
60
61     public static UserGroupInformation getUGI() throws LoginException, IOException {
62         String doAs = System.getenv( name: "HADOOP_USER_NAME");
63         if (doAs != null && doAs.length() > 0) {
64             /*
65              * this allows doAs (proxy user) to be passed along across process boundary where
66              * delegation tokens are not supported. For example, a DDL stmt via WebHCat with
67              * a doAs parameter, forks to 'hcat' which needs to start a Session that
68              * proxies the end user
69              */
70             return UserGroupInformation.createProxyUser(doAs, UserGroupInformation.getLoginUser());
71         }
72         return UserGroupInformation.getCurrentUser();
73     }
74 }
```




HIVE授权详解-Hive.server2.enableDoas

- 当启用了 HIVE 的代理机制时 (hive.server.enable.doAs=true)，业务终端用户如 xyz 提交的 HIVE SQL 作业底层的 MR/TEZ/SPARK 任务访问 HDFS/YARN 时，HDFS/YARN 验证的是业务终端用户 xyz 的身份 (后续 HDFS/YARN 的权限校验，校验的也是 xyz 用户的权限)；
- 当没有启用 HIVE 的代理机制时 (hive.server.enable.doAs=false)，业务终端用户提交的 HIVE SQL 作业底层的 MR/TEZ/SPARK 任务访问 HDFS/YARN 时，需要验证的是 hiveserver2 服务对应的用户，即 hive 的身份 (后续 HDFS/YARN 的权限校验，校验的也是 hive 用户的权限)；

```
AbstractSparkClient.java x SparkSubmitSparkClient.java x SparkLauncherSparkClient.java x
isDoAsEnabled x Cc W .* 2 results 29 1 18 29
121 }
122
123 @Override
124 protected void addKeytabAndPrincipal(boolean isDoAsEnabled, String keyTabFile, String principal)
125 {
126     if (isDoAsEnabled) {
127         List<String> kinitArgv = Lists.newLinkedList();
128         kinitArgv.add("kinit");
129         kinitArgv.add(principal);
130         kinitArgv.add("-k");
131         kinitArgv.add("-t");
132         kinitArgv.add(keyTabFile + ";");
133         kinitArgv.addAll(argv);
134     } else {
135         // if doAs is not enabled, we pass the principal/keypad to spark-submit in order to
136         // support the possible delegation token renewal in Spark
137         argv.add("--principal");
138         argv.add(principal);
139         argv.add("--keytab");
140         argv.add(keyTabFile);
141     }
142 }
343 // The options --principal/--keypad do not work with --proxy-user in spark-submit.sh
344 // (see HIVE-15485, SPARK-5493, SPARK-19143), so Hive could only support doAs or
345 // delegation token renewal, but not both. Since doAs is a more common case, if both
346 // are needed, we choose to favor doAs. So when doAs is enabled, we use kinit command,
347 // otherwise, we pass the principal/keypad to spark to support the token renewal for
348 // long-running application.
349 if ("kerberos".equals(hiveConf.get(HADOOP_SECURITY_AUTHENTICATION))) {
350     String principal = SecurityUtil.getServerPrincipal(hiveConf.getVar(ConfVars.HIVE_SERVER2_KERBEROS_PRINCIPAL),
351         hostname: "0.0.0.0");
352     String keyTabFile = hiveConf.getVar(ConfVars.HIVE_SERVER2_KERBEROS_KEYTAB);
353     boolean isDoAsEnabled = hiveConf.getBoolVar(HiveConf.ConfVars.HIVE_SERVER2_ENABLE_DOAS);
354     if (StringUtils.isNotBlank(principal) && StringUtils.isNotBlank(keyTabFile)) {
355         addKeytabAndPrincipal(isDoAsEnabled, keyTabFile, principal);
356     }
357 }
358 if (hiveConf.getBoolVar(HiveConf.ConfVars.HIVE_SERVER2_ENABLE_DOAS)) {
359     try {
360         String currentUser = Utils.getUGI().getShortUserName();
361         // do not do impersonation in CLI mode
362         if (!currentUser.equals(System.getProperty("user.name"))) {
363             LOG.info("Attempting impersonation of " + currentUser);
364             addProxyUser(currentUser);
365         }
366     } catch (Exception e) {
367         String msg = "Cannot obtain username: " + e;
368         throw new IllegalStateException(msg, e);
369     }
370 }
```




HIVE授权详解-相关参数

- `hive.security.authorization.enabled`: Enable or disable the Hive client authorization
- `hive.security.authorization.manager`:
`org.apache.hadoop.hive.ql.security.authorization.plugin.fallback.FallbackHiveAuthorizerFactory/org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationProvider/org.apache.hadoop.hive.ql.security.authorization.DefaultHiveMetastoreAuthorizationProvider/org.apache.sentry.binding.hive.authz.SentryHiveAuthorizerFactory`;
- `hive.metastore.pre.event.listeners`: The pre-event listener classes to be loaded on the metastore side to run code whenever databases, tables, and partitions are created, altered, or dropped. Set this configuration property to `org.apache.hadoop.hive.ql.security.authorization.AuthorizationPreEventListener` in `hive-site.xml` to turn on Hive metastore-side security;
- `hive.security.metastore.authorization.manager`: This tells Hive which metastore-side authorization provider to use. Defaults to `org.apache.hadoop.hive.ql.security.authorization.StorageBasedAuthorizationProvider`.
- `hive.security.authorization.createtable.owner.grants`
- `hive.security.authorization.sqlstd.confwhitelist`
- `hive.warehouse.subdir.inherit.perms`
- `hive.server2.enable.doAs/hive.server2.enable.impersonation`
- `hive.security.metastore.authenticator.manager`: The authenticator manager class name to be used in the metastore for authentication, defaults to `org.apache.hadoop.hive.ql.security.HadoopDefaultMetastoreAuthenticator`
- `hive.security.authenticator.manager`: Hive client authenticator manager class name. The user-defined authenticator should implement interface `org.apache.hadoop.hive.ql.security.HiveAuthenticationProvider`. Default to `org.apache.hadoop.hive.ql.security.HadoopDefaultAuthenticator`.

06

金融行业大数据安全最佳实践



金融行业大数据安全最佳实践

- 在金融行业强调安全的背景下，我们推荐对大数据集群开启kerberos 安全认证, 开启用户权限校验：
`hadoop.security.authentication=kerberos, dfs.permissions.enabled=true`
- HDFS 文件的授权，要遵循最小化原则，不能对目标目录笼统设置777！
- HDFS文件的授权，建议主要使用 POSIX MODE 来进行设置，必要时通过 POSIX ACL 进行补充：
`dfs.namenode.acls.enabled=true`
- 业务用户对HDFS文件的使用，建议将所有文件，比如 jar/配置文件/数据字典等，存放在统一的目录下，比如 `/user/hundsun/dap`



金融行业大数据安全最佳实践

- 针对HIVE的认证，我们推荐两种方式：只开启Kerberos认证模式/开启Kerberos认证和LDAP认证模式，其中后者的实际含义是“大数据集群（hdfs/yarn/zookeeper/hive等）开启了kerberos安全认证，同时 hive 组件配置了使用 LDAP认证”；
- 具体的LDAP，如果要跟企业域账户打通，则需要使用企业内部统一的 LDAP 服务器；否则建议使用一个单独的ldap 服务器（比如 OpenLdap/ApacheDS/microsoft AD 等 ldap的某个具体实现），比如在 TDH中，底层实际使用的是 Guardienn 底层自带的一个 ldap实现（本质是ApacheDS），创建用户更改密码等操作都是在 Guardienn 中操作的，更企业域账户是独立的；
- 针对HIVE的授权，我们推荐使用 sentry/ranger等第三方插件统一进行授权和鉴权，此时需要关闭hive的代理功能 `Hive.server2.enableDoas=false`；
- 我们推荐各个业务系统使用该系统对应的业务用户的身份提交业务SQL，而不是各个业务系统统一都使用 HIVE 系统用户；
- 业务用户对HIVE表的使用，如果都是通过HIVESERVER2来访问，建议使用 ORC 事务内表；如果通过 SPARK 等计算引擎做 ETL，建议使用 orc/parquet格式的HIVE外表，并存放在统一的目录下，比如 `/user/hundsun/dap`，针对外表目录的授权，需要结合 HDFS POSIX ACL 来进行；



Thank you