设置Hadoop用户以便访问任何HDFS文件

Hadoop分布式文件系统实现了一个和POSIX系统类似的文件和目录的权限模型。每个文件和目录有一个所有者（owner）和一个组（group）。文件或目录对其所有者、同组的其他用户以及所有其他用户分别有着不同的权限。对文件而言，当读取这个文件时需要有r权限，当写入或者追加到文件时需要有w权限。对目录而言，当列出目录内容时需要具有r权限，当新建或删除子文件或子目录时需要有w权限，当访问目录的子节点时需要有x权限。每个访问HDFS的用户进程的标识分为两个部分，分别是用户名和组名列表。每次用户进程访问一个文件或目录foo，HDFS都要对其进行权限检查：

1、如果用户即foo的所有者，则检查所有者的访问权限；
2、如果foo关联的组在组名列表中出现，则检查组用户的访问权限；
3、否则检查foo其他用户的访问权限。

如果权限检查失败，则客户的操作会失败。所以如果我们以某个用户ls HDFS上某个文件夹时可能会出现以下的错误信息：

```
Caused by: org.apache.hadoop.ipc.RemoteException
(org.apache.hadoop.security.AccessControlException):
Permission denied: user=iteblog, access=EXECUTE,
 inode="/user/hadoop":iteblog:iteblog:drwx------
```

上面的iteblog用户试图访问HDFS上/user/hadoop目录下的文件，但是iteblog是属于iteblog组的，而/user/hadoop目录的权限是drwx------，也就是只有该文件夹所有者才能访问这个目录。仔细研究，我们可以找到这个用户获取模块是在UserGroupInformation类中获取的，部分代码如下：

```
Principal user = null;
// if we are using kerberos, try it out
if (isAuthenticationMethodEnabled(AuthenticationMethod.KERBEROS)) {
  user = getCanonicalUser(KerberosPrincipal.class);
  if (LOG.isDebugEnabled()) {
    LOG.debug("using kerberos user:"+user);
  }
}
//If we don't have a kerberos user and security is disabled, check
//if user is specified in the environment or properties
if (!isSecurityEnabled() && (user == null)) {
  String envUser = System.getenv(HADOOP_USER_NAME);
  if (envUser == null) {
    envUser = System.getProperty(HADOOP_USER_NAME);
  }
```

```
  user = envUser == null ? null : new User(envUser);
}
// use the OS user
if (user == null) {
  user = getCanonicalUser(OS_PRINCIPAL_CLASS);
  if (LOG.isDebugEnabled()) {
    LOG.debug("using local user:"+user);
  }
}
// if we found the user, add our principal
if (user != null) {
  subject.getPrincipals().add(new User(user.getName()));
  return true;
}
LOG.error("Can't find user in " + subject);
throw new LoginException("Can't find user name");
```

从上面代码片段可以知道，Hadoop先判断集群是否启用了Kerberos授权，如果是，则直接从配置中获取用户（可以为空）；如果不是，则往下走。然后判断是否启用了安全认证，安全认证是通过下面参数配置的：

```
<property>
      <name>hadoop.security.authentication</name>
      <value>simple</value>
</property>
```

默认是不启用的。所以如果没有启用安全认证或者从Kerberos获取的用户为null，那么获取HADOOP_USER_NAME环境变量，并将它的值作为Hadoop执行用户。如果我们没有设置HADOOP_USER_NAME环境变量，那么程序将调用whoami来获取当前用户，并用groups来获取用户所在组。

根据上面的分析，我们有两个办法来解决用户访问HDFS失败的问题（这两个方法能够使用的前提都是没有启用安全认证）：

1、我们可以在启动程序之前设置HADOOP_USER_NAME环境变量，比如：

```
[iteblog@www.iteblog.com ~]$ export HADOOP_USER_NAME=iteblog
```

2、我们可以自定义whoami和groups命令，返回咱们需要的用户或者组。

只要Hadoop没有启用安全认证，我们想咋修改用户和用户所在组就咋修改，这样对存在HDFS上的文件来说是非常不利的。

```
val ugi = UserGroupInformation.createRemoteUser("iteblog")
ugi.doAs(new PrivilegedExceptionAction[Void]() {
  def  run():Void= {
    //something todo
    return null
  }
})
```

当然，这个前提也是没有启用安全认证。 -->