**1. 背景**

在Hadoop1.0.0或者CDH3 版本之前， hadoop并不存在安全认证一说。默认集群内所有的节点都是可靠的，值得信赖的。用户与HDFS或者M/R进行交互时并不需要进行验证。导致存在恶意用户伪装成真正的用户或者服务器入侵到hadoop集群上，恶意的提交作业，修改JobTracker状态，篡改HDFS上的数据，伪装成NameNode 或者TaskTracker接受任务等。 尽管在版本0.16以后， HDFS增加了文件和目录的权限，但是并没有强认证的保障，这些权限只能对偶然的数据丢失起保护作用。恶意的用户可以轻易的伪装成其他用户来篡改权限，致使权限设置形同虚设。不能够对Hadoop集群起到安全保障。

在Hadoop1.0.0或者CDH3版本后，加入了Kerberos认证机制。使得集群中的节点就是它们所宣称的，是信赖的。Kerberos可以将认证的密钥在集群部署时事先放到可靠的节点上。集群运行时，集群内的节点使用密钥得到认证。只有被认证过节点才能正常使用。企图冒充的节点由于没有事先得到的密钥信息，无法与集群内部的节点通信。防止了恶意的使用或篡改Hadoop集群的问题，确保了Hadoop集群的可靠安全。

**2. Hadoop 安全问题**

**2.1  用户到服务器的认证问题**

* NameNode，,JobTracker上没有用户认证

用户可以伪装成其他用户入侵到一个HDFS 或者MapReduce集群上。

* DataNode上没有认证

Datanode对读入输出并没有认证。导致如果一些客户端如果知道block的ID，就可以任意的访问DataNode上block的数据

* JobTracker上没有认证

可以任意的杀死或更改用户的jobs，可以更改JobTracker的工作状态

**2.2  服务器到服务器的认证问题**

* 没有DataNode, TaskTracker的认证

用户可以伪装成datanode ,tasktracker，去接受JobTracker, Namenode的任务指派。

**3. Kerberos能解决的Hadoop安全认证问题**

kerberos实现的是机器级别的安全认证，也就是前面提到的服务到服务的认证问题。事先对集群中确定的机器由管理员手动添加到kerberos数据库中，在KDC上分别产生主机与各个节点的keytab(包含了host和对应节点的名字，还有他们之间的密钥)，并将这些keytab分发到对应的节点上。通过这些keytab文件，节点可以从KDC上获得与目标节点通信的密钥，进而被目标节点所认证，提供相应的服务，防止了被冒充的可能性。

* **解决服务器到服务器的认证**

由于kerberos对集群里的所有机器都分发了keytab，相互之间使用密钥进行通信，确保不会冒充服务器的情况。集群中的机器就是它们所宣称的，是可靠的。

防止了用户伪装成Datanode，Tasktracker，去接受JobTracker，Namenode的任务指派。

* **解决client到服务器的认证**

Kerberos对可信任的客户端提供认证，确保他们可以执行作业的相关操作。防止用户恶意冒充client提交作业的情况。

用户无法伪装成其他用户入侵到一个HDFS 或者MapReduce集群上

用户即使知道datanode的相关信息，也无法读取HDFS上的数据

用户无法发送对于作业的操作到JobTracker上

* **对用户级别上的认证并没有实现**

无法控制用户提交作业的操作。不能够实现限制用户提交作业的权限。不能控制哪些用户可以提交该类型的作业，哪些用户不能提交该类型的作业。这些由ACL模块控制（参考）

**4. Kerberos工作原理介绍**

**4.1  基本概念**

Princal(安全个体)：被认证的个体，有一个名字和口令

KDC(key distribution center ) : 是一个网络服务，提供ticket 和临时会话密钥

Ticket：一个记录，客户用它来向服务器证明自己的身份，包括客户标识、会话密钥、时间戳。

AS (Authentication Server)： 认证服务器

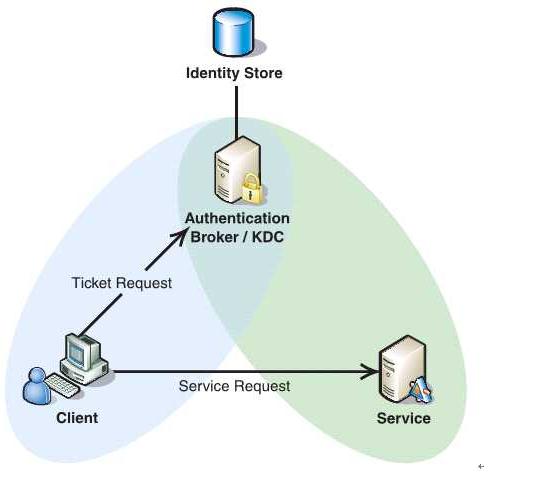
TSG(Ticket Granting Server)： 许可证服务器

**4.2  kerberos 工作原理**

**4.2.1  Kerberos协议**

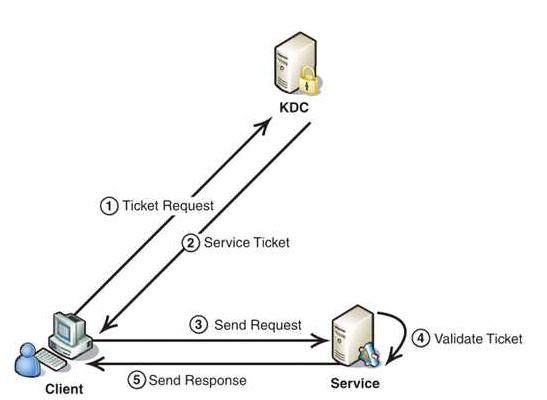
Kerberos可以分为两个部分：

* Client向KDC发送自己的身份信息，KDC从Ticket Granting Service得到TGT(ticket-granting ticket)， 并用协议开始前Client与KDC之间的密钥将TGT加密回复给Client。此时只有真正的Client才能利用它与KDC之间的密钥将加密后的TGT解密，从而获得TGT。（此过程避免了Client直接向KDC发送密码，以求通过验证的不安全方式）
* Client利用之前获得的TGT向KDC请求其他Service的Ticket，从而通过其他Service的身份鉴别



**4.3 Kerberos认证过程**

Kerberos协议的重点在于第二部分（即认证过程）：



（1）Client将之前获得TGT和要请求的服务信息(服务名等)发送给KDC，KDC中的Ticket Granting Service将为Client和Service之间生成一个Session Key用于Service对Client的身份鉴别。然后KDC将这个Session Key和用户名，用户地址（IP），服务名，有效期, 时间戳一起包装成一个Ticket(这些信息最终用于Service对Client的身份鉴别)发送给Service， 不过Kerberos协议并没有直接将Ticket发送给Service，而是通过Client转发给Service，所以有了第二步。

（2）此时KDC将刚才的Ticket转发给Client。由于这个Ticket是要给Service的，不能让Client看到，所以KDC用协议开始前KDC与Service之间的密钥将Ticket加密后再发送给Client。同时为了让Client和Service之间共享那个密钥(KDC在第一步为它们创建的Session Key)，KDC用Client与它之间的密钥将Session Key加密随加密的Ticket一起返回给Client。

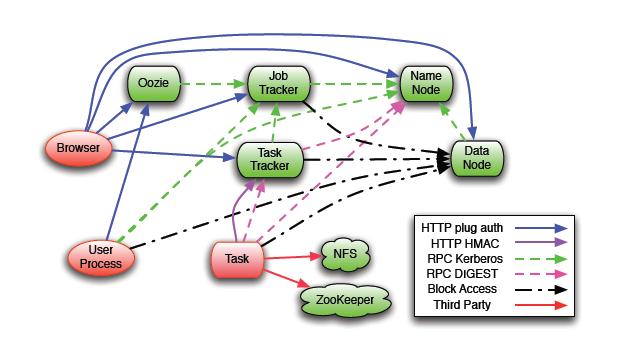
（3）为了完成Ticket的传递，Client将刚才收到的Ticket转发到Service. 由于Client不知道KDC与Service之间的密钥，所以它无法算改Ticket中的信息。同时Client将收到的Session Key解密出来，然后将自己的用户名，用户地址（IP）打包成Authenticator用Session Key加密也发送给Service。

（4）Service 收到Ticket后利用它与KDC之间的密钥将Ticket中的信息解密出来，从而获得Session Key和用户名，用户地址（IP），服务名，有效期。然后再用Session Key将Authenticator解密从而获得用户名，用户地址（IP）将其与之前Ticket中解密出来的用户名，用户地址（IP）做比较从而验证Client的身份。

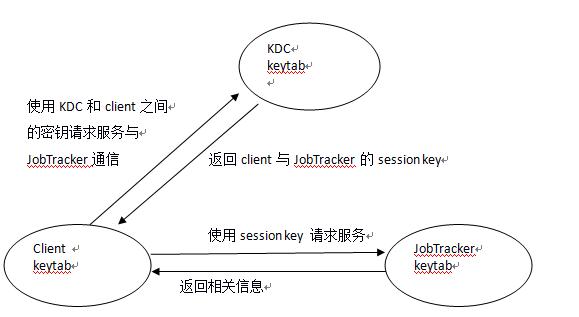
（5）如果Service有返回结果，将其返回给Client。

**4.4  kerberos在Hadoop上的应用**

Hadoop集群内部使用Kerberos进行认证



具体的执行过程可以举例如下：



**4.5  使用kerberos进行验证的原因**

* **可靠** Hadoop 本身并没有认证功能和创建用户组功能，使用依靠外围的认证系统
* **高效** Kerberos使用对称钥匙操作，比SSL的公共密钥快
* **操作简单** 用户可以方便进行操作，不需要很复杂的指令。比如废除一个用户只需要从Kerbores的KDC数据库中删除即可。

**5.  参考资料**

（1）kerberos原理：<http://idior.cnblogs.com/archive/2006/03/20/354027.html>

（2）什么是kerberos：[http://www.logicprobe.org/~octo/pres/pres\_kerberos.pdf](http://www.logicprobe.org/%7Eocto/pres/pres_kerberos.pdf)

（3）“Hadoop Security Design”  Owen O’Malley, Kan Zhang, Sanjay Radia,  Ram Marti, and Christopher Harrell