1.1 What is Kerberos

1.1.1 简单介绍  
　　Kerberos是一个用于鉴定身份（authentication）的协议， 它采取对称密钥加密（symmetric-key cryptography），这意味着密钥不会在网络上传输。在Kerberos中，未加密的密码（unencrypted password）不会在网络上传输，因此攻击者无法通过嗅探网络来偷取用户的密码。

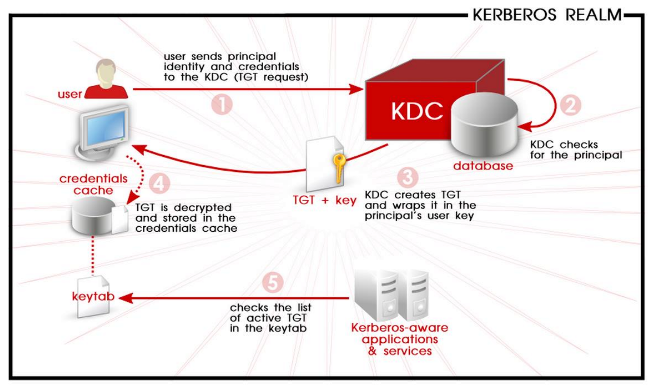
　　Kerberos利用对称加密和受信任的第三方（即KDC, key distribution center）来鉴别要求使用网络服务的用户的身份。所有有KDC和secondary KDC管理的主机构成了一个域（realm）。

　　当一个用户的身份通过了KDC的验证后，KDC会向该用户发送一个证书/票据（该证书/票据是与这次会话绑定的），其他kerberized services会在该用户的主机上搜索该ticket，而不是要求用户使用密码来验证他的身份。  
1.1.2 关于Kerberos的一些概念  
Principal  
一个用户会以一个独一无二的身份来被KDC认证，该身份被称为principal。一个Principal由三个部分组成：primary, instance以及realm，其组成形式为primary/instance@realm。  
primary : 可以是OS中的username，也可以是service name；  
instance : 用于区分属于同一个user或者service的多个principals，该项为optional；  
realm : 类似于DNS中的domain，定义了一组principals  
1.1.3 认证过程  
　　1、当用户在一个kerberos-aware网络中登录到他的workstation之后，authentication server将向KDC发送一个TGT请求（a request for a ticket-granting ticket），而他的principal将作为其中的组成部分。该请求可以由登录程序来负责发送（这样该过程对用户透明），也可以在用户登录后通过执行kinit来发送。

　　2、KDC在其数据库中检查该pricipal。

　　3、如果找到了该principal，则KDC将创建一个TGT，并用user key进行加密，然后将加密后的TGT发送给该用户。这里，user key并不是用户的密码，而是由用户密码计算而来（例如hash）。

　　4、用户所在主机的Kerberos client的登录程序或者kinit程序在收到加密的TGT后，用该用户的user key进行解密。user key只会在client主机上被使用，绝不会在网络上传输。KDC发送的ticket将被保存在一个本地文件中（credentials cache），它可以被kerberized services查验。

　　5、用户的身份验证完成后，servers（运行着kerberized applciations & services）可以查验被识别的principals及其keys（这将被保存在keytab中），而不必用kinit来查验。  


　　TGT有一个可设定的失效期（通常为10~24小时），这会被保存在client所在主机的credential cache中。在ticket过期之前，用户在请求kerberized services的服务时不需要再次输入用户密码，除非他们登出后再登录。

　　每当用户需要访问一个network service时，client会利用TGT向TGS（ticket-granting server）请求获得针对该特定网络服务的一个新的ticket。之后，用户可以利用该service ticket来向该network service实现authentication。  
1.1.4 How Kerberos works  
　　KDC就是受信任的第三方（trusted third party arbitrator），KDC上运行着2个重要的Kerberos daemons，即 kadmind 和 krb5kdc。  
　　Kadmind: 这是管理Kerberos server的进程，一个名为kadmin 的程序使用 kadmind 来维护principal database和policy configuration。  
　　Krb5kdc: 在Kerberos authentication的过程中担负trusted third party arbitrator的职责。

1.2 认证原理  
1.2.1 知识准备  
　　为了使读者更加容易理解，在这里我们先给出两个重要的概念：  
　　▪ Long-term Key/Master Key：在Security的领域中，有的Key可能长期内保持不  
变，比如你在密码，可能几年都不曾改变，这样的Key、以及由此派生的Key被称为Long-term Key。对于Long-term Key的使用有这样的原则：被Long-term Key加密的数据不应该在网络上传输。原因很简单，一旦这些被Long-term Key加密的数据包被恶意的网络监听者截获，在原则上，只要有充足的时间，他是可以通过计算获得你用于加密的Long-term Key的——任何加密算法都不可能做到绝对保密。

　　在一般情况下，对于一个Account来说，密码往往仅仅限于该Account的所有者知晓，甚至对于任何Domain的Administrator，密码仍然应该是保密的。但是密码却又是证明身份的凭据，所以必须通过基于你密码的派生的信息来证明用户的真实身份，在这种情况下，一般将你的密码进行Hash运算得到一个Hash code, 我们一般管这样的Hash Code叫做Master Key。由于Hash Algorithm是不可逆的，同时保证密码和Master Key是一一对应的，这样既保证了你密码的保密性，又同时保证你的Master Key和密码本身在证明你身份的时候具有相同的效力。  
　　▪ Short-term Key/Session Key：由于被Long-term Key加密的数据包不能用于网络传送，所以我们使用另一种Short-term Key来加密需要进行网络传输的数据。由于这种Key只在一段时间内有效，即使被加密的数据包被黑客截获，等他把Key计算出来的时候，这个Key早就已经过期了。  
1.2.2 认证详细原理  
　　Kerberos实际上一个基于Ticket的认证方式。Client想要获取Server端的资源，先得通过Server的认证；而认证的先决条件是Client向Server提供从KDC获得的一个有Server的Master Key进行加密的Session Ticket（Session Key + Client Info）。可以这么说，Session Ticket是Client进入Server领域的一张门票。而这张门票必须从一个合法的Ticket颁发机构获得，这个颁发机构就是Client和Server双方信任的KDC，同时这张Ticket具有超强的防伪标识：它是被Server的Master Key加密的。对Client来说，获得Session Ticket是整个认证过程中最为关键的部分。  
　　为了更好的说明整个Ticket Distribution的过程，我在这里做一个类比。现在的股事很火爆，上海基本上是全民炒股，我就举一个认股权证的例子。有的上市公司在股票配股、增发、基金扩募、股份减持等情况会向公众发行认股权证，认股权证的持有人可以凭借这个权证认购一定数量的该公司股票，认股权证是一种具有看涨期权的金融衍生产品。  
而我们今天所讲的Client获得Ticket的过程也和通过认股权证购买股票的过程类似。如果我们把Client提供给Server进行认证的Ticket比作股票的话，那么Client在从KDC那边获得Ticket之前，需要先获得这个Ticket的认购权证，这个认购权证在Kerberos中被称为TGT：Ticket Granting Ticket，TGT的分发方仍然是KDC。  
我们现在来看看Client是如何从KDC处获得TGT的：首先Client向KDC发起对TGT的申请，申请的内容大致可以这样表示：“我需要一张TGT用以申请获取用以访问所有Server的Ticket”。KDC在收到该申请请求后，生成一个用于该Client和KDC进行安全通信的Session Key（SKDC-Client）。为了保证该Session Key仅供该Client和自己使用，KDC使用Client的Master Key和自己的Master Key对生成的Session Key进行加密，从而获得两个加密的SKDC-Client的Copy。对于后者，随SKDC-Client一起被加密的还包含以后用于鉴定Client身份的关于Client的一些信息。最后KDC将这两份Copy一并发送给Client。这里有一点需要注意的是：为了免去KDC对于基于不同Client的Session Key进行维护的麻烦，就像Server不会保存Session Key（SServer-Client）一样，KDC也不会去保存这个Session Key（SKDC-Client），而选择完全靠Client自己提供的方式。



　　通过上面的过程，Client实际上获得了两组信息：一个通过自己Master Key加密的Session Key，另一个被Sever的Master Key加密的数据包，包含Session Key和关于自己的一些确认信息。

Client通过自己的Master Key对KDC加密的Session Key进行解密从而获得Session Key，随后创建Authenticator（Client Info + Timestamp）并用Session Key对其加密。最后连同从KDC获得的、被Server的Master Key加密过的数据包（Client Info + Session Key）一并发送到Server端。我们把通过Server的Master Key加密过的数据包称为Session Ticket。  
　　当Server接收到这两组数据后，先使用他自己的Master Key对Session Ticket进行解密，从而获得Session Key。随后使用该Session Key解密Authenticator，通过比较Authenticator中的Client Info和Session Ticket中的Client Info从而实现对Client的认证。



1.2.3kafka认证过程  
　　总结起来也很简单，Broker启动时，需要使用配置文件中的身份和密钥文件向KDC（Kerberos服务器）认证，认证通过则加入Kafka集群，否则报错退出。  
Producer（或Consumer）启动后需要经过如下步骤与Broker建立安全的Socket连接：  
　　1.Producer向KDC认证身份，通过则得到TGT（票证请求票证），否则报错退出  
　　2.Producer使用TGT向KDC请求Kafka服务，KDC验证TGT并向Producer返回SessionKey（会话密钥）和ServiceTicket（服务票证）  
　　3.Producer使用SessionKey和ServiceTicket与Broker建立连接，Broker使用自身的密钥解密ServiceTicket，获得与Producer通信的SessionKey，然后使用SessionKey验证Producer的身份，通过则建立连接，否则拒绝连接。