# Kerberos

## Kerberos协议

Kerberos协议主要用于计算机网络的身份鉴别(Authentication), 其特点是用户只需输入一次身份验证信息就可以凭借此验证获得的票据(ticket-granting ticket)访问多个服务，即SSO(Single Sign On)。由于在每个Client和Service之间建立了共享密钥，使得该协议具有相当的安全性。

## 角色

Kerbores中有三种角色：

KDC：负责分发密钥的密钥分配中心

Client：需要使用kerbores服务的客户端

Service：提供具体服务的服务端

## 3.条件

如下图所示，Client与KDC， KDC与Service 在协议工作前已经有了各自的共享密钥，并且由于协议中的消息无法穿透防火墙，这些条件就限制了Kerberos协议往往用于一个组织的内部， 使其应用场景不同于X.509 PKI。



## 4.过程

Kerberos协议获取原始票据过程：

1. Client向KDC发送自己的身份信息；
2. KDC从Ticket Granting Service得到TGT(ticket-granting ticket)；
3. KDC用Client与KDC之间的密钥将TGT加密；
4. KDC将加密后的TGT发送给Client；
5. Client利用Client与KDC之间的密钥将加密的TGT解密，从而获得真正的TGT；

Kerberos协议获取服务票据过程：



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；
2. 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。
3. 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Service Ticket，用于Service对Client的身份鉴别；
4. KDC用KDC与Service之间的密钥将Service Ticket加密，这个Ticket是要给Service的，不能让Client看到；并且KDC用Client与它之间的密钥将Session Key加密；
5. KDC将加密的Service Ticket和加密的Session Key发送给Client ；
6. Client解密Session Key，然后将自己的用户名，用户地址（IP）打包Session Key加密也发送给Service；Client将收到的加密的Service Ticket转发到Service；
7. Service 收到Ticket后，用它与KDC之间的密钥将Service Ticket解密，从而获得Session Key和用户名，用户地址（IP），服务名，有效期。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。
8. 如果Service有返回结果，将其返回给Client。

## 5.keytab

keytab（密钥表）是“key table（密钥表）”的缩写，提供服务的每台主机都必须包含称为 keytab（密钥表）的本地文件。密钥表包含相应服务的主体，称为服务密钥。服务使用服务密钥向 KDC 进行自我验证，并且只有 Kerberos 和服务本身知道服务密钥。例如，如果您有基于 Kerberos 的 NFS 服务器，则该服务器必须具有包含其 nfs 服务主体的密钥表文件。

要将服务密钥添加至密钥表文件，应使用 kadmin 的 ktadd 命令，将相应的服务主体添加至主机的密钥表文件。由于要将服务主体添加至密钥表文件，因此该主体必须已存在于 Kerberos 数据库中，以便 kadmin 可验证其存在。

## 6.总结

概括起来说Kerberos协议主要做了两件事

1. Ticket的安全传递。
2. Session Key的安全发布。

再加上时间戳的使用就很大程度上的保证了用户鉴别的安全性。并且利用Session Key，在通过鉴别之后Client和Service之间传递的消息也可以获得Confidentiality(机密性), Integrity(完整性)的保证。不过由于没有使用非对称密钥自然也就无法具有抗否认性，这也限制了它的应用。不过相对而言它比X.509 PKI的身份鉴别方式实施起来要简单多了。

KDC为什么不直接将包发送给Client和Server？

* 由于一个Server会面对若干不同的Client, 而每个Client都具有一个不同的Session Key。那么Server就会为所有的Client维护这样一个Session Key的列表，这样做对于Server来说是比较麻烦而低效的。
* 由于网络传输的不确定性，可能出现这样一种情况：Client很快获得Session Key，并将这个Session Key作为Credential随同访问请求发送到Server，但是用于Server的Session Key确还没有收到，并且很有可能承载这个Session Key的永远也到不了Server端，Client将永远得不到认证。