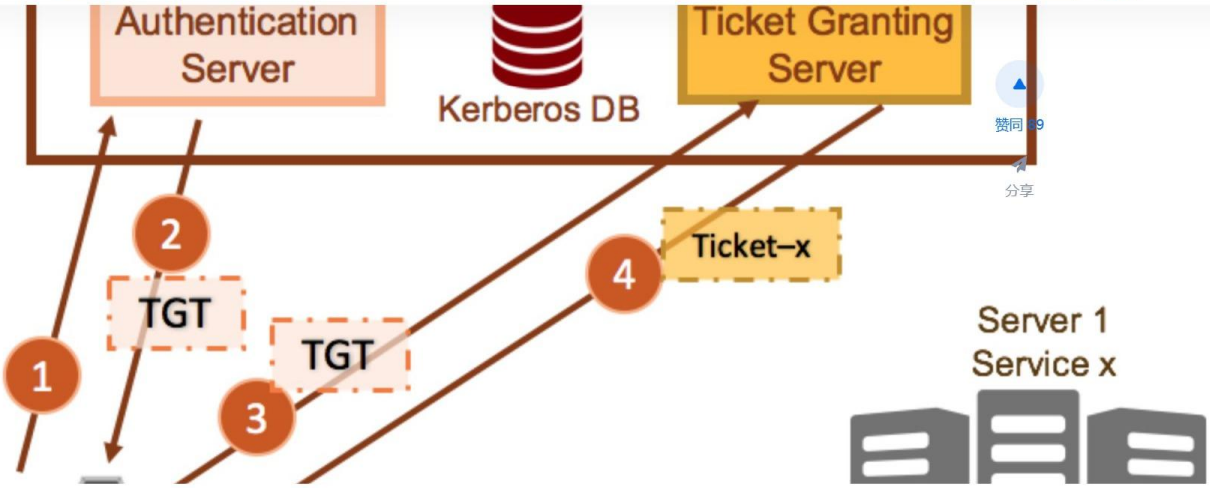


一文搞定Kerberos

tkanng
欢迎私信交流。

Kerberos 是一种身份认证协议，被广泛运用在大数据生态中，甚至可以说是大数据身份认证的事实标准。本文将详细说明 Kerberos 原理。

PS: 这是 Introduction To Kerberos 的文字版，感兴趣的小伙伴可以直接下载pdf。

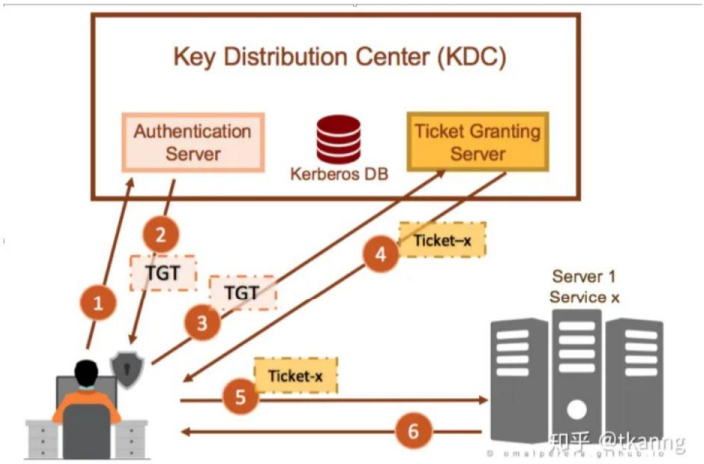
一、关于 Kerberos 的典型问题

1. Kerberos 是什么?
2. Kerberos 具体原理是什么?
3. 为什么 Kerberos 是一种成熟可靠的身份认证协议?
4. ...

二、Kerberos 是什么?

Kerberos 一词来源于古希腊神话中的 Cerberus —— 守护地狱之门的三头犬。下图是 Kerberos 的 LOGO:

一句话来说，Kerberos 是一种基于加密 Ticket 的身份认证协议。Kerberos 主要由三个部分组成：Key Distribution Center (即KDC)、Client 和 Service。 大致关系如下图所示:



客户端会先访问两次KDC，然后再访问目标Service，如：HTTP服务。

三、Kerberos 基本概念

3.1 基本概念

- Principal：大致可以认为是 Kerberos 世界的用户名，用于标识身份。principal 主要由三部分构成：primary，instance(可选) 和 realm。包含 instance 的principal，一般会作为server端的principal，如：NameNode，HiverServer2，Presto Coordinator等；不含有 instance 的principal，一般会作为 客户端的principal，用于身份认证。例子如下图所示：

含义	Principal
部署在 foo.alibaba.com节点的 hiveserver2 的 principal	hiveserver2/foo.alibaba.com@EXAMPLE.COM instance primary realm
用户小明 xiaoming	xiaoming@EXAMPLE.COM realm

- Keytab："密码本"。包含了多个 principal 与密码的文件，用户可以利用该文件进行身份认证。
- Ticket Cache：客户端与 KDC 交互完成后，包含身份认证信息的文件，短期有效，需要不断 renew。
- Realm：Kerberos 系统中的一个namespace。不同 Kerberos 环境，可以通过 realm 进行区分。

3.2 KDC

Key Distribution Center（即 KDC），是 Kerberos 的核心组件，主要由三个部分组成：

- Kerberos Database: 包含了一个 Realm 中所有的 principal、密码与其他信息。（默认：Berkeley DB）
- Authentication Service(AS): 进行用户信息认证，为客户端提供 Ticket Granting Tickets(TGT)。
- Ticket Granting Service(TGS): 验证 TGT 与 Authenticator，为客户端提供 Service Tickets。

四、Kerberos 原理

在深入了解 Kerberos 原理之前，先介绍一下 Kerberos 协议的几个大前提，帮助大家理解：

1. Kerberos 基于 Ticket 实现身份认证，而非密码。**如果客户端无法利用本地密钥，解密出 KDC 返回的加密Ticket，认证将无法通过。**

2. 客户端将**依次**与 Authentication Service, Ticket Granting Service 以及目标Service进行交互，共三次交互。

3. 客户端与其他组件交互是，都将获取到**两条**信息，其中一条可以通过本地密钥解密出，另外一条将无法解密出。

4. 客户端想要访问的目标服务，**将不会直接**与KDC交互，而是通过能否正确解密出客户端的请求来进行认证。

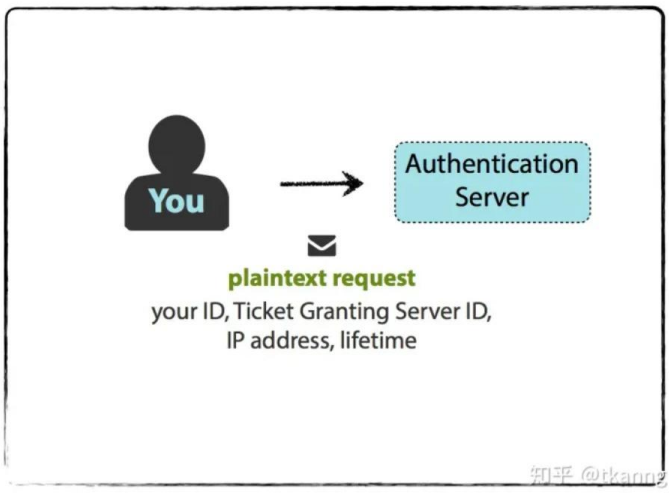
5. KDC Database 包含有所有 principal 对应的密码。

6. Kerberos 中信息加密方式一般是**对称加密**（可配置成非对称加密）。

下面，我们将以客户端访问 http 服务为例，解释整个认证过程。

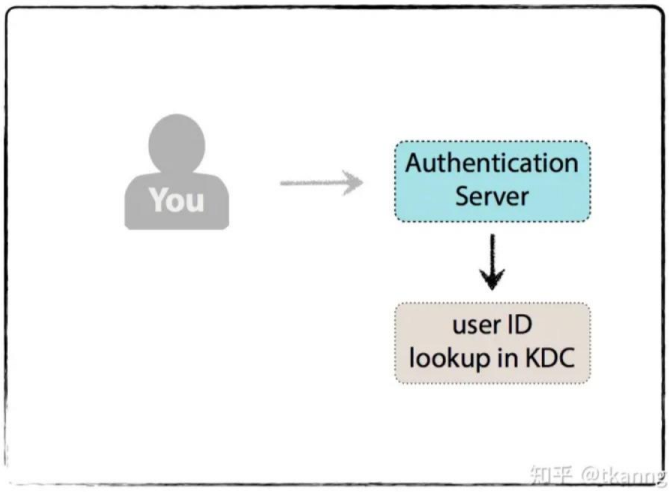
4.1 客户端与 Authentication Service

第一步，客户端通过 `kinit USERNAME` 或其他方式，将客户端ID, 目标HTTP服务ID, 网络地址（可能是多个机器的IP地址列表，如果想在任何机器上使用，则可能为空），以及TGT有效期的寿命等信息发送给 Authentication Service。

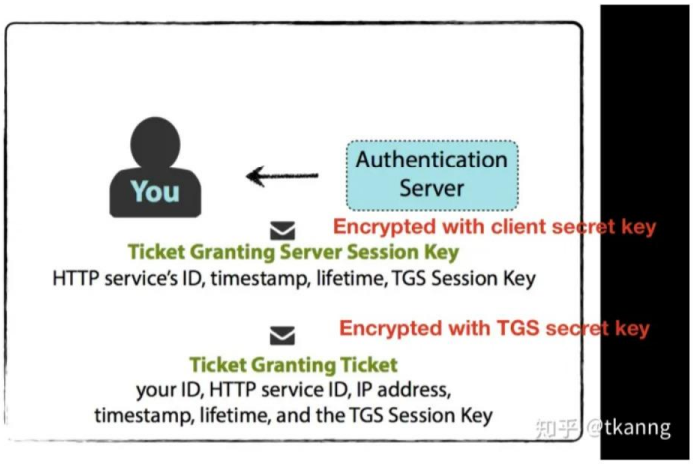


第二步，Authentication Server 将检查客户端ID是否在KDC数据库中

第二步，Authentication Server 将检查客户端ID是否在KDC数据库中。

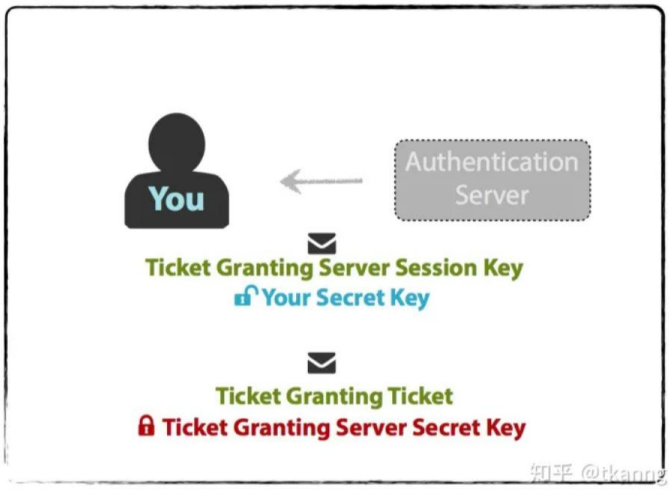


如果 Authentication Server 检查操作没有异常，那么KDC将随机生成一个 key，用于客户端与 Ticket Granting Service(TGS) 通信。这个Key，一般被称为 TGS Session Key。随后 Authentication Server 将发送**两条信息**给客户端。示意图如下：



其中一条信息被称为TGT，由TGS的密钥加密，客户端无法解密，包含客户端ID, TGS Session Key 等信息。另一条信息由客户端密钥加密，客户端可以正常解密，包含目标 HTTP 服务ID，TGS Session Key等信息。

第三步，客户端利用本地的密钥解密出第二条信息。如果本地密钥无法解密出信息，那么认证失败。示意图如下：



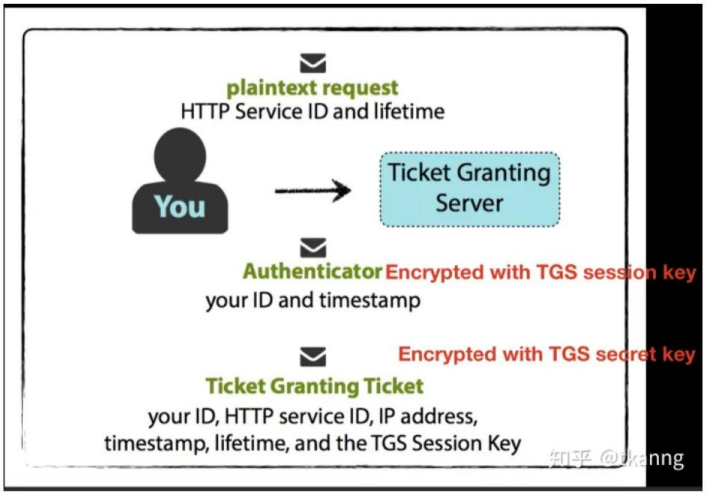
4.2 客户端与 Ticket Granting Service

这时候，客户端有了 TGT（由于本地没有TGS的密钥，导致无法解密出其数据）与 TGS Session Key。

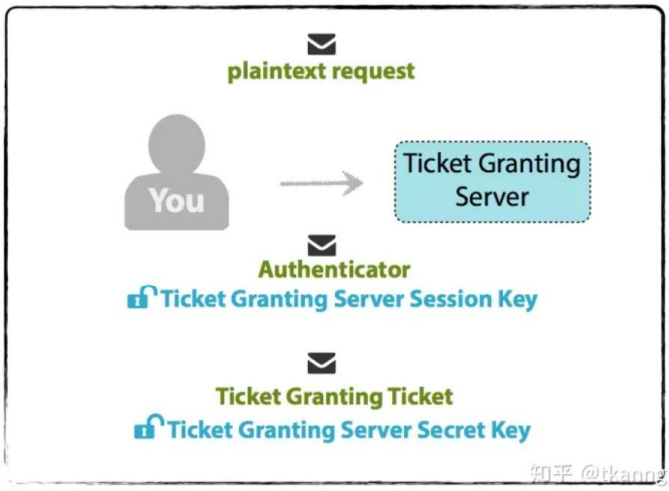
第四步，客户端将:

- “无脑”将 AS 发送过来的TGT（由**TGS密钥**加密）转发给TGS

- 将包含自身信息的Authenticator(由**TGS Session Key**加密)发送给TGS



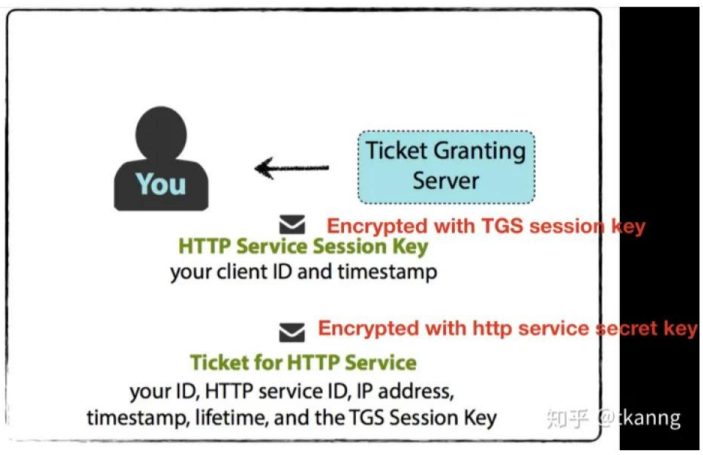
第五步，TGS 将利用 自身的密钥从TGT中解密出**TGS Session Key**，然后利用**TGS Session Key**从Authenticator 中解密出客户端的信息。



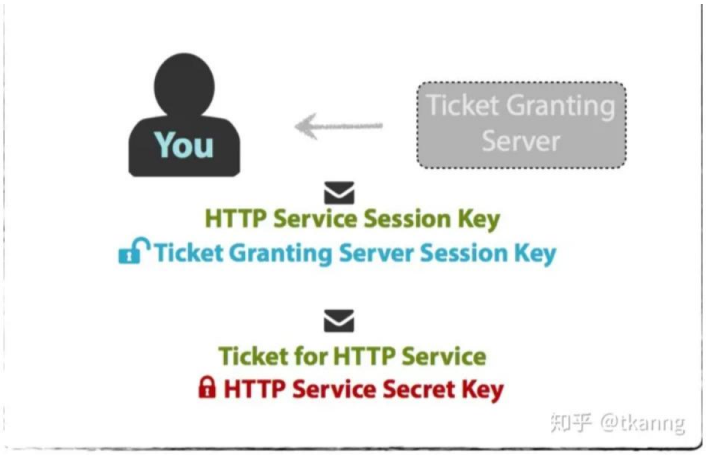
TGS 解密出所有信息后，将进行身份检查，进行认证：

- 将客户端ID与TGT的客户端ID进行比较
- 比较来自 Authenticator 的时间戳和TGT的时间戳 (典型的Kerberos系统的容忍度是2分钟，但也可以另行配置)
- 检查TGT是否过期
- 检查Authenticator是否已经在TGS的缓存中（为了避免重放攻击）

当所有检查都通过后， TGS 随机生成一个 Key 用于后续客户端与 HTTP 服务交互时进行通信加密使用，即 HTTP Session Key。同样地，TGS 将发送**两条信息**给客户端: 其中一条是 HTTP Ticket，由 **HTTP 服务的密钥**进行加密；另一条则由**TGS Session Key**加密，包含了客户端信息与时间戳。



第六步，客户端将利用**TGS Session Key**解密出其中一条信息，另一条信息由于是由目标HTTP服务加密，无法解密。



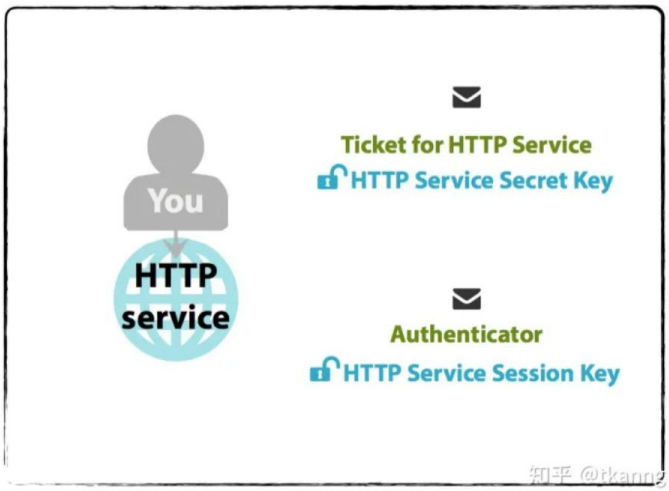
4.3 客户端与 HTTP Service

这时候，客户端有了HTTP Ticket（由于本地没有HTTP服务的密钥，导致无法解密出其数据）与HTTP Session Key。

第七步，客户端将:

- “无脑”将 AS 发送过来的 HTTP Ticket（由**HTTP 密钥**加密）转发给目标 http 服务。
- 将包含自身信息的Authenticator(由**HTTP Session Key**加密)发送给 http 服务。

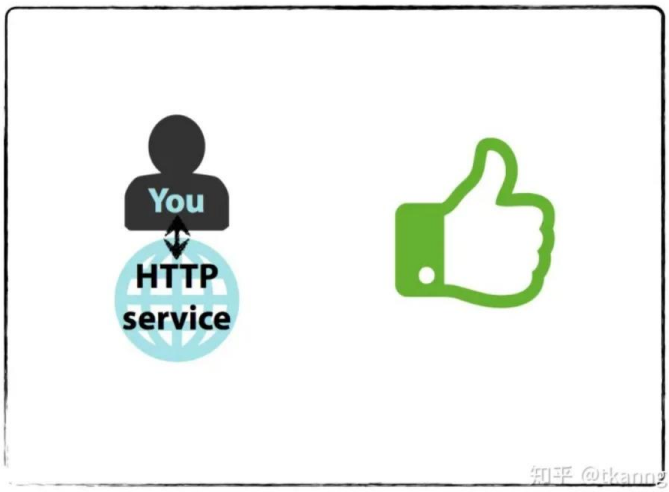
第八步，HTTP服务首先利用自身的密钥解密出 HTTP Ticket 的信息，得到 HTTP Session Key；随后，利用HTTP Session Key解密出用户的Authenticator信息。



信息解密完成后，HTTP 服务同样需要做一些信息检查:

- 将 Authenticator 中的客户端ID与HTTP Ticket中的客户端ID进行比较
- 比较来自 Authenticator 的时间戳和 HTTP Ticket 的时间戳 (典型的 Kerberos 系统对差异的容忍度是 2 分钟，但也可以另行配置)
- 检查Ticket是否过期
- 检查 Authenticator 是否已经在HTTP服务器的缓存中（为了避免重播攻击）

至此，所有的认证过程通过，客户端即可与远程HTTP服务完成了身份认证，可以进行后续的信息通信。



五、Kerberos 的优势

1. 密码无需进行网络传输。基于 Ticket 实现身份认证，保障密钥安全性。
2. 双向认证。整个认证过程中，不仅需要客户端进行认证，待访问的服务也需要进行身份认证。
3. 高性能。一旦Client获得用过访问某个Server的Ticket，该Server就能根据这个Ticket实现对Client的验证，而无须KDC的再次参与。

六、参考

编辑于 2020-10-17 23:20

真诚赞赏，手留余香

赞赏

还没有人赞赏，快来当第一个赞赏的人吧!

身份认证 用户认证体系 HTTP

暂无评论

文章被以下专栏收录

Hive源码阅读

Hive源码阅读
xxx

推荐阅读

Kerberos简介——教你做个好人

慕课网 发表于猿论

由浅入深理解Kerberos协议

看雪

kerberos域用户提权分析

ailx1... 发表于信息安全入...

亚马逊选品插件：Helium10隐秘的产品分析和评论下载功能...

韩大海