身份认证实例

郑燕飞

引言

- Kerberos: part of Project Athena at MIT
- Greek Kerberos: 希腊神话故事中一种三个头的狗，还有一个蛇形尾巴。是地狱之门的守卫。
- Modern Kerberos: 意指有三个组成部分的网络之门的保卫者。“ 三头 ”包括：
  - 认证(authentication)
  - 簿记(accounting)
  - 审计(audit)

问题

在一个开放的分布式网络环境中，用户通过工作站访问服务器上提供的服务。

- 服务器应能够限制非授权用户的访问并能够认证对服务的请求。
- 工作站不能够被网络服务所信任其能够正确地认定用户，即工作站存在三种威胁。
  - 一个工作站上一个用户可能冒充另一个用户操作；
  - 一个用户可能改变一个工作站的网络地址，从而冒充另一台工作站工作；
  - 一个用户可能窃听他人的信息交换，并用重放攻击获得对一个服务器的访问权或中断服务器的运行。

Kerberos要解决的问题

- 所有上述问题可以归结为一个非授权用户能够获得其无权访问的服务或数据。
- 不是为每一个服务器构造一个身份认证协议，Kerberos提供一个中心认证服务器，提供用户到服务器和服务器到用户的认证服务。
- Kerberos采用传统加密算法（无公钥体制）。
- Kerberos Version4和Version5 (RFC1510)。

信息系统资源保护的动机

- 单用户单机系统。用户资源和文件受到物理上的安全保护；
- 多用户分时系统。操作系统提供基于用户标识的访问控制策略，并用logon过程来标识用户。
- Client/Server网络结构。由一组工作站和一组分布式或中心式服务器组成。

C/S环境下三种可能的安全方案

- 相信每一个单独的客户工作站可以保证对其用户的识别，并依赖于每一个服务器强制实施一个基于用户标识的安全策略。
- 要求客户端系统将它们自己向服务器作身份认证，但相信客户端系统负责对其用户的识别。
- 要求每一个用户对每一个服务证明其标识身份，同样要求服务器向客户端证明其标识身份。

Kerberos的解决方案

- Kerberos支持以上三种策略。
- 在一个分布式的client/server体系机构中采用一个或多个Kerberos服务器提供一个认证服务。
- 总体方案是提供一个可信第三方的认证服务。

Kerberos系统应满足的要求

- 安全。网络窃听者不能获得必要信息以假冒其它用户；Kerberos应足够强壮以至于潜在的敌人无法找到它的弱点连接。
- 可靠。Kerberos应高度可靠，并且应借助于一 个分布式服务器体系结构，使得一个系统能够备份另一个系统
- 透明。理想情况下，用户除了要求输入口令以外应感觉不到认证的发生。
- 可伸缩。系统应能够支持大数量的客户和服务器。

Kerberos Version4

- 引入一个信任的第三方认证服务，采用一个基于Needham & Schroeder协议。
- 采用DES，精心设计协议，提供认证服务。

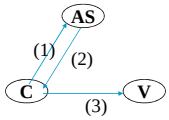
一个简单的认证对话

- 引入认证服务器(AS)，它知道所有用户的口令并将它们存储在一个中央数据库中。另外，AS与每一个服务器共有一个唯一的保密密钥。这些密钥已经通过物理上或以更安全的手段分发。

考虑以下假定的对话：

(1) C → AS: $ID_C \| P_C \| ID_V$
(2) AS → C: Ticket
(3) C → V : $ID_C \| Ticket$

$Ticket = E_{K_V}[ID_C \| AD_C \| ID_V]$



其中：

C	: client	ID_V	: identifier of V
AS	: Authentication Server	P_C	: password of user on C
V	: server	AD_C	: network address of C
ID_C	: identifier of user on C	K_V	: AS与V共有的保密密钥

AD_C防止何种攻击？

更安全的认证对话

- 两个主要问题
  - 希望用户输入口令的次数最少。
  - 口令以明文传送会被窃听。
- 解决办法
  - 票据重用（ticket reusable）
  - 票据许可服务器（ticket-granting server，TGS）

改进后的假想的对话：

用户登录的每次对话:

(1) C → AS : ID_C || ID_{tgs}

(2) AS → C : $E_{K_C}[Ticket_{tgs}]$

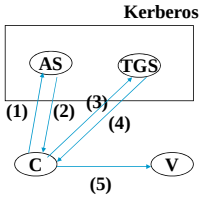
每种服务类型一次：

(3) C → TGS : ID_C || ID_v || $Ticket_{tgs}$

(4) TGS → C : $Ticket_V$

每种服务会话一次：

(5) C → V : ID_C || $Ticket_V$



$Ticket_{tgs} = E_{K_{tgs}}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V = E_{K_V}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

方案的详细描述

- 用户向AS请求代表该用户的票据许可票据。
- AS发回加密的票据，密钥由口令导出。
- 票据许可票据包含用户ID、网络地址、TGS的ID、时戳与生存期。
- 用户请求服务许可票据。
- TGS验证，如通过则发服务许可票据。
- 用户使用服务许可票据请求服务。

Kerberos V4 的认证对话

- 两个问题
  - 与TGS相关的生存期问题；
    - 太长则？太短则？如何应付票据的过期使用？
  - 需要服务器向客户进行认证其本身；
    - 假的服务器。
- 解决方案
  - 会话密钥（session key）

AS用安全方式向用户和TGS各自提供一块秘密信息，然后用户也以安全方式向TGS出示该秘密来证明自己的身份。这个秘密就是会话密钥。

Kerberos V4报文交换总结

- 认证服务交换：获得票据许可票据
  - (1) C → AS : ID_C || ID_{tgs} || TS_1
  - (2) AS → C : $E_{K_C}[K_{c,tgs}$ || ID_{tgs} || TS_2 || $Lifetime_2$ || $Ticket_{tgs}]$
  - $Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs}$ || ID_C || AD_C || ID_{tgs} || TS_2 || $Lifetime_2]$

Kerberos V4报文交换总结

- 票据许可服务交换：获得服务许可票据
  - (3) C → TGS : ID_V || $Ticket_{tgs}$ || $Authenticator_c$
  - (4) TGS → C : $E_{K_{c,tgs}}[K_{c,v}$ || ID_V || TS_4 || $Ticket_v]$

    $Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs}||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2]$

    $Ticket_v = E_{K_V}[K_{c,v}||ID_C||AD_C||ID_v||TS_4||Lifetime_4]$

    $Authenticator_c = E_{K_{c,tgs}}[ID_c||AD_c||TS_3]$

Kerberos V4报文交换总结

- 客户/服务器认证交换：获得服务
  - (5) C → V : $Ticket_v$ || $Authenticator_c$
  - (6) V → C : $E_{K_{c,v}}[TS_5+1]$ (for mutual authentication)

    $Ticket_v = E_{K_V}[K_{c,v}||ID_c||AD_c||ID_v||TS_4||Lifetime_4]$

    $Authenticator_c = E_{K_{c,v}}[ID_c||AD_c||TS_5]$

基本原理

(a) 认证服务交换

Message(1) Client 请求 ticket-granting ticket

ID_C ： 告诉AS本client端的用户标识；

ID_{tgs} ： 告诉AS用户请求访问TGS；

TS_1 ： 让AS验证client端的时钟是与AS的时钟同步的；

Message(2) AS返回ticket-granting ticket

E_{K_C} ： 基于用户口令的加密，使得AS和client可以验证口令，并保护Message(2)。

$K_{c,tgs}$ ： session key的副本，由AS产生，client可用于在AS与client之间信息的安全交换，而不必共用一个永久的key。

ID_{tgs} ： 确认这个ticket是为TGS制作的。

TS_2 ： 告诉client该ticket签发的时间。

$Lifetime_2$: 告诉client该ticket的有效期；

$Ticket_{tgs}$: client用来访问TGS的ticket。

(b) 票据许可服务交换

Message(3) client 请求service-granting ticket

ID_v: 告诉TGS用户要访问服务器V；

$Ticket_{tgs}$： 向TGS证实该用户已被AS认证；

$Authenticator_c$:由client生成，用于验证ticket；

Message(4) TGS返回service-granting ticket

$E_{K_{c,tgs}}$： 仅由C和TGS共享的密钥；用以保护Message(4)；

$K_{c,tgs}$: session key的副本，由TGS生成，供client和server之间信息的安全交换，而无须共用一个永久密钥。

ID_v： 确认该ticket是为server V签发的；

TS_4： 告诉client该ticket签发的时间；

$Ticket_V$： client用以访问服务器V的ticket；

$Ticket_{tgs}$: 可重用，从而用户不必重新输入口令；

$E_{K_{tgs}}$： ticket用只有AS和TGS才知道的密钥加密，以预防篡改；

$K_{c,tgs}$： TGS可用的session key副本，用于解密authenticator,从而认证ticket；

IDc： 指明该ticket的正确主人；

客户/服务器鉴别交换

Message(5) client 请求服务

$Ticket_v$： 向服务器证实该用户已被AS认证；

$Authenticator_c$:由客户生成，用于验证ticket有效；

Message(6) 客户对服务器的可选认证

$E_{K_{c,v}}$： 使C确认报文来自V；

TS_5+1： 使C确信这不使报文重放；

$Ticket_V$： client用以访问服务器V的ticket；

E_{K_V}： 用只有AS和TGS才知道的密钥加密的票据，以预防篡改；

$K_{c,v}$: 用户的会话密钥副本；

IDc： 票据的合法用户；

ADc： 防止非法使用；

IDv: 使服务器确信解密正确；

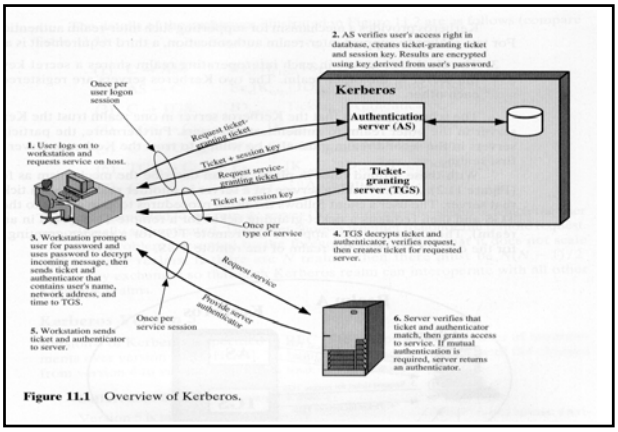


Figure 11.1 Overview of Kerberos.

Kerberos管辖范围与多重服务

- 一个完整的Kerberos环境包括一个Kerberos服务器，一组工作站，和一组应用服务器，满足下列要求：
  - Kerberos服务器必须在其数据库中拥有所有参与用户的ID(UID)和口令散列表。所有用户均在Kerberos服务器上注册。
  - Kerberos服务器必须与每一个服务器之间共享一个保密密钥。所有服务器均在Kerberos服务器上注册。

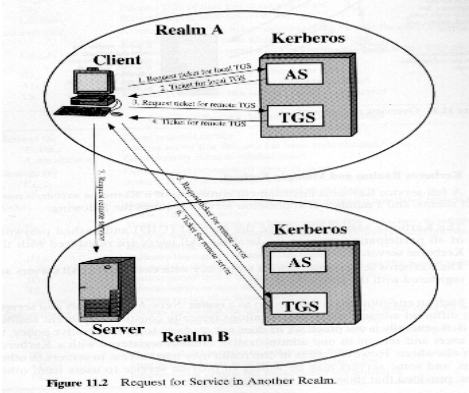


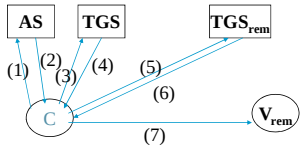
Figure 11.2 Request for Service in Another Realm.

获得另一领域中的认证服务

- 分三步骤：
  （1）获得本地TGS的访问权；
  （2）请求一张远程TGS的票据许可票据；
  （3）向远程TGS申请其领域内的服务许可票据

细节描述：

(1) C → AS : $ID_C \| ID_{tgs} \| TS_1$
(2) AS → C : $E_{K_C}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$
(3) C → TGS: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$
(4) TGS → C: $E_{K_{c,tgs}}[K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}]$
(5) C → TGS_{rem}: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$
(6) TGS → C: $E_{K_{c,tgsrem}}[K_{c,vrem} \| ID_{vrem} \| TS_b \| Ticket_{vrem}]$
(7) C → V_{rem}: $Ticket_{vrem} \| Authenticator_c$



Kerberos Version 5

- 改进version 4 的环境缺陷
  - 加密系统依赖性，需DES
  - Internet协议依赖性，需IP地址
  - 消息字节次序（不明确字节顺序）
  - Ticket的时效性（可能太短）
  - 认证转发，用户的认证不能转发到其它主机或用户。
  - 域间认证，

Kerberos Version 5

- 改进Version 4 的技术缺陷
  - Double encryption
  - PCBC encryption
  - Session key
  - Password attacks

X.509认证服务

- ITU-T的X.509建议书是定义目录服务的X.500系列推荐书的一部分。
- 目录是保存有关用户信息数据库的一个服务器或一组服务器。
- 信息包括从用户姓名到网络地址的映射以及用户的其他属性和信息。
- X.509定义了一个由X.500目录向它的用户提供的认证服务框架。目录可以看作公钥证书知识库。 X.509还定义了基于公钥证书的认证协议。

X.509认证服务

- X.509最早于1988年发布，1995起草了第三版。它是一个重要的标准，有广泛的应用：IP安全，SSL，SET等。
- X.509基于公钥加密和签名。推荐使用RSA，没有指定散列算法。

X.509证书格式

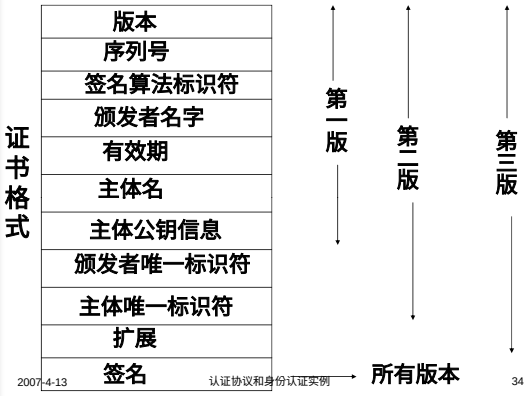
- X.509方案的核心是与每个用户联系的公开密钥证书。证书由可信证书权威机构（CA）创建，由CA或用户放在目录中。
- 目录服务器不负责公钥的生成或证书函数，它仅提供一个易于访问的位置以便用户获得证书。

X.509证书格式

- 版本：标识不同版本的证书；
- 序列号：标识证书唯一性的整数；
- 签名算法标识符：标识签名算法及参数；
- 颁发者名字：创建和签名该证书的CA的X.500名字；
- 有效期：两个日期组成：起始时间和结束时间；
- 主体名：证书提及的用户名；
- 主体的公钥信息：主体的公钥及这个密钥使用算法的标识符，和算法的相关参数；

X.509证书格式

- 颁发者的唯一标识符：可选的唯一字串用于证实CA，如果该X.500名字已经被其它实体使用。
- 主体的唯一标识符：可选的唯一字串用于证实主体，如果该X.500名字已经被其它实体使用。
- 扩展：扩展字段，在第三版使用。
- 签名：CA密钥加密的其它所有字段的散列值。



X.509证书格式

- 用下列符号标识一个证书

CA<< A >> = CA { V, SN, AI, CA, T_A, A, Ap}

  - Y<< X >>表示Y给X发的证书；
  - Y{I}表示Y对I的签名，I是“ V, SN, AI, CA, T_A, A, Ap”（分别对应上图的参数1－7）的散列值。

获得用户证书

- CA用户证书的特征：
  - 任何有CA公钥的用户都可以恢复被证明的用户公钥；
  - 除了CA外没有任何一方能不被察觉地更改证书；
- 因此，证书可放在无特殊保护的公开目录中。
- 问题：
  - 用户过多，不能都用同一个CA的证书；
  - 多个CA，不同的证书如何认证？

获得用户证书（不同CA）

- 解决以上问题的办法是：
  - CA之间能安全交换公钥
- 不同CA用户任何获得对方的公钥：
  - A获得X1签名的X2的证书，则A可得X2的公钥；
  - A可得X2签名的B的证书，则A可用X2的公钥验证B的公钥；
- 证书链（任意长度）：

A :X1 << X2 >> X2 << B >>

B :X2 << X1 >> X1<< A >>

证书撤销

- 证书撤销的原因
  - 用户密钥泄漏
  - 用户不再由这个CA颁发证书
  - CA证书泄漏
- CA维护一个证书撤销表
- 用户收到一个证书后，必须查表确定该证书是否被撤销。避免更多的开销，用户可保持一个证书和表的高速缓存。

认证过程

- 单向认证

$A\{t_A, r_A, B, sgnData, E_{kUb}[K_{ab}]\}$

- 双向认证

$A\{t_A, r_A, B, sgnData, E_{kUb}[K_{ab}]\}$

$B\{t_B, r_B, A, r_A, sgnData, E_{kUa}[K_{ba}]\}$

- 三向认证

$A\{t_A, r_A, B, sgnData, E_{kUb}[K_{ab}]\}$

$B\{t_B, r_B, A, r_A, sgnData, E_{kUa}[K_{ba}]\}$

$A\{r_B\}$ //无需检查时戳，当没有同步时钟时必须

X.509 Ver3

- 第二版不能满足的要求
  - 主体字段太短
  - 没有安全策略信息
  - 不能限制恶意CA的损害
  - 不能使同一拥有者在不同时间对不同密钥进行分离的验证。

X.509 Ver3

- 证书扩展
  - 密钥和策略信息
  - 主体和颁发者的属性
  - 证书路径约束