操作系统安全

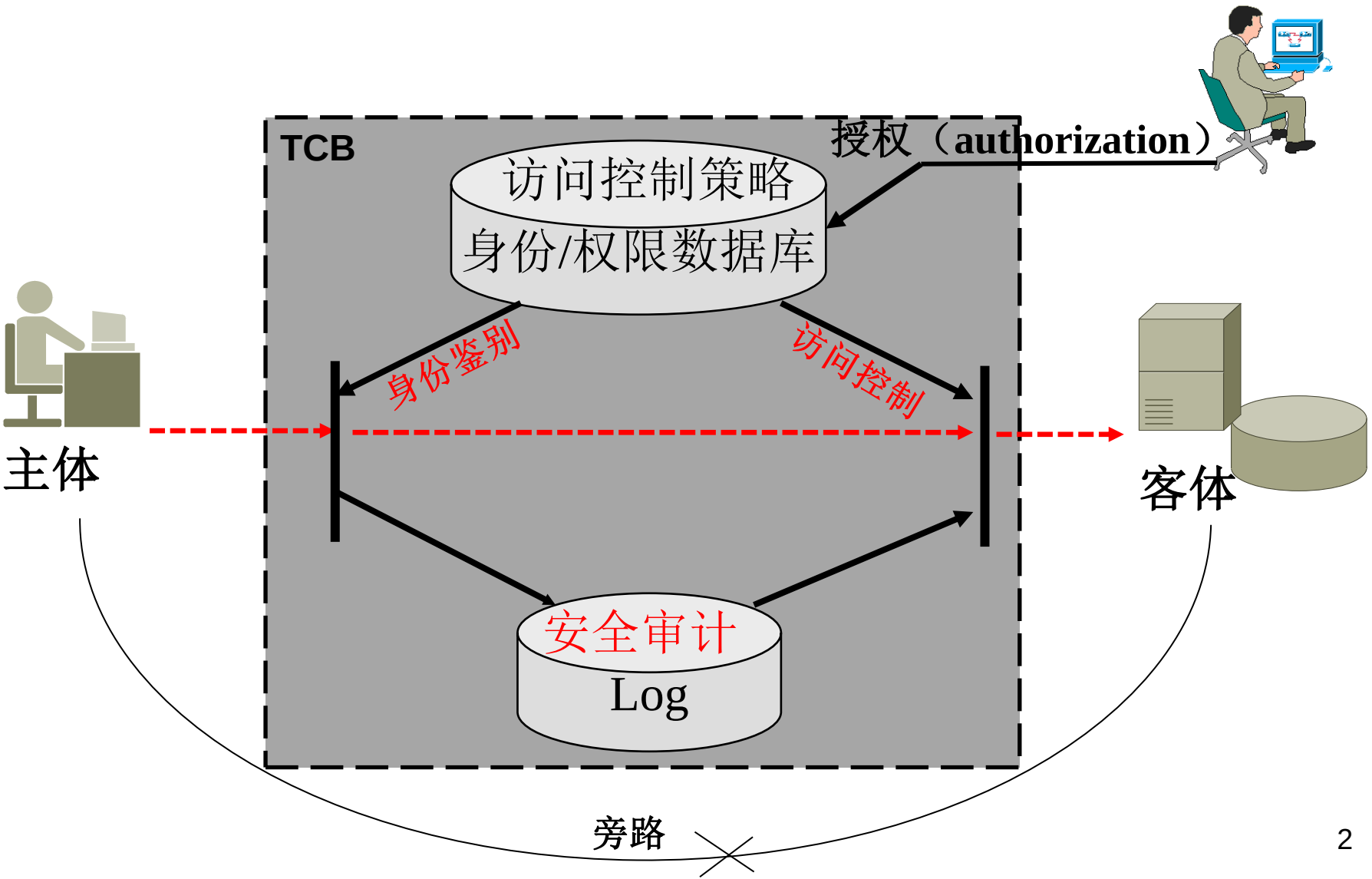
2-3 操作系统安全机制

中国科学院大学
网络空间安全学院
2018/4/13

中国科学院大学
University of Chinese Academy of Sciences

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

安全机制之间的关系


TCB
授权（authorization）
访问控制策略
身份/权限数据库
身份鉴别
访问控制
主体
客体
安全审计
Log
旁路

目录

标识与鉴别的概念

- 标识(identification)：用户向系统表明身份
  - 系统可以识别的用户的内部名称：用户名、智能卡、登录ID
  - 具有唯一性、不能被伪造
- 鉴别(authentication)：对用户宣称的身份标识的有效性进行校验和测试的过程。用户鉴别是有效实施其他安全策略的基础。

用户标识的名称和标识符均为公开的明码信息
用于鉴别的信息是非公开的和难以仿造的

标识与鉴别的要求

- 标识鉴别机制的设计和实现需要达到两方面的要求：
  - （1）标识鉴别系统的设计要协助安全操作系统实现新增的安全功能和安全策略，包括增加新的用户属性，并通过扩展标识鉴别命令来支持这些属性
  - （2）标识鉴别系统本身的安全性要达到安全操作系统本身的安全级别要求，增加新的安全功能，提高安全性。

标识与鉴别实现方法

- 1、口令——账户、密码
- 2、密码验证——PKI（公开密钥基础设施）
- 3、生物标识与鉴别技术——面部、指纹、虹膜、声音等特征
- 4、可信计算基TCB——与鉴别相关的认证机制

1.口令

- 口令鉴别机制简单易行，但最为脆弱
- 口令管理
- 系统管理员的职责
- 用户的职责
- 口令实现要点

口令鉴别机制-静态口令鉴别

- 静态口令鉴别机制
  - 传统的静态口令鉴别机制是利用用户名和口令核对的方法对系统进行维护。用户登录系统时，系统通过对比用户输入的口令和用户ID，来判断用户身份的合法性。

口令鉴别机制-动态口令鉴别

- 动态口令鉴别机制
  - 基本原理是在客户端登录过程中，基于用户的秘密通行短语加入不确定因素，对通行短语和不确定因素进行变换，所得结果作为认证数据（即动态口令），提交给认证服务器，认证服务器接收到用户的认证数据后，以事先规定的算法去验算认证数据，从而实现对用户身份的认证。
  - 根据口令生成时不确定因素的选择方式，动态口令机制有时间同步机制、挑战/应答机制、事件同步机制等。

口令鉴别机制-一次口令

- 一次口令
  - 固定口令方案存在的主要问题：防止窃听和随后的口令重放（无时变参数）
  - 解决办法：每个口令只用一次
  - 变体
    - 一次口令的共享列表
    - 顺序更新一次口令
    - 基于单向函数的一次口令序列：
    - w,H(w),H(H(w)), …

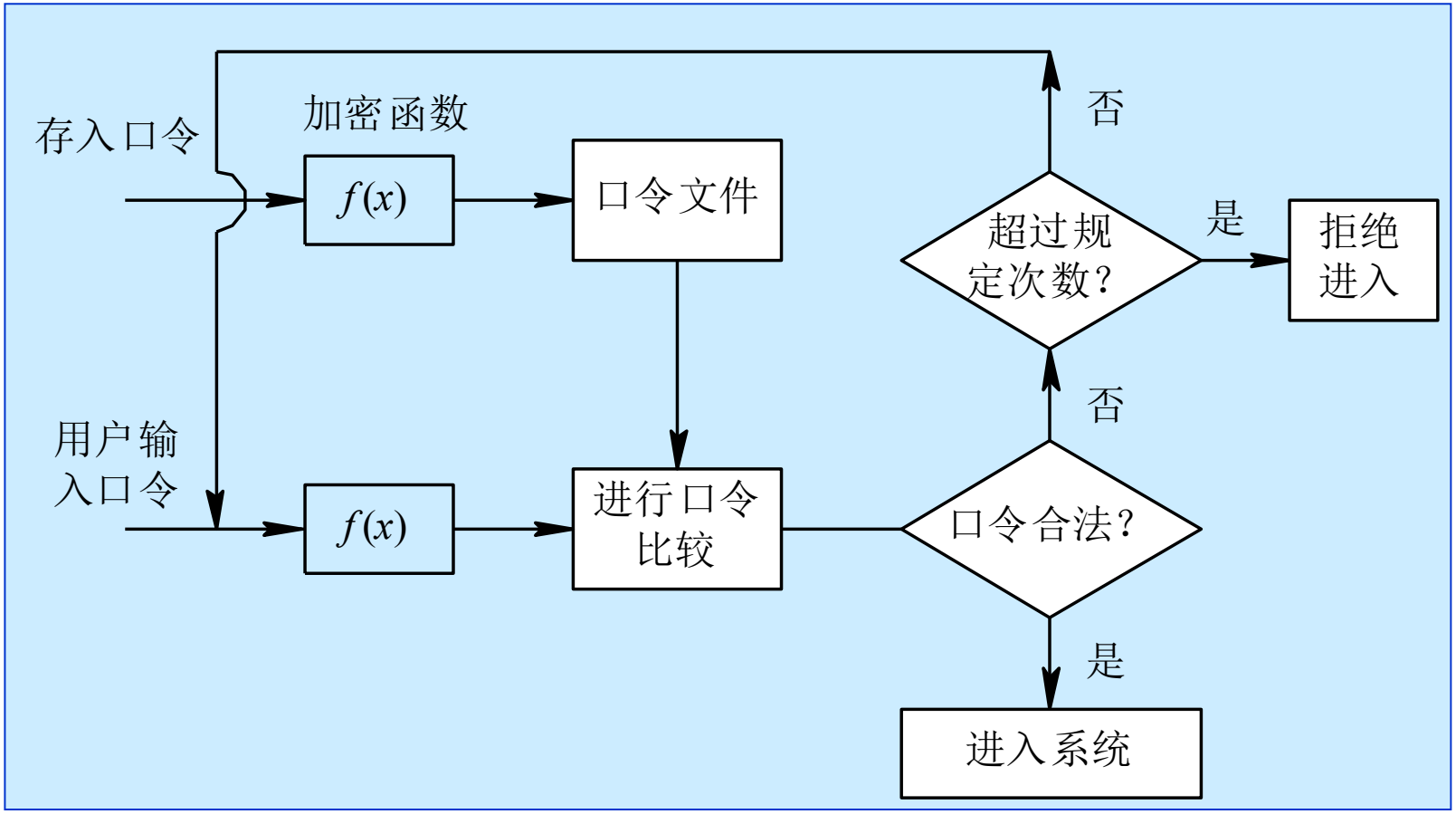
口令鉴别机制-口令文件

- 口令文件
  - 通常在口令机制中，都配置有一份口令文件，用于保存合法用户的口令和与口令相联系的特权。该文件的安全性至关重要，一旦攻击者成功地访问了该文件，攻击者便可随心所欲地访问他感兴趣的所有资源，这对整个计算机系统的资源和网络，将无安全性可言。
  - 显然，如何保证口令文件的安全性，已成为系统安全性的头等重要问题。

口令鉴别机制-口令文件

- 口令文件
  - 保证口令文件安全性的其中一个行之有效的方法是选择一个函数来对口令进行加密，该函数f(x)具有这样的特性：在给定了x值后，很容易算出f(x)；然而，如果给定了f(x)的值，却不能算出x的值。利用f(x)函数去编码(即加密)所有的口令，再将加密后的口令存入口令文件中。当某用户输入一个口令时，系统利用函数f(x)对该口令进行编码，然后将编码(加密)后的口令与存储在口令文件中的已编码的口令进行比较，如果两者相匹配，便认为是合法用户
  - 即使攻击者能获取口令文件中的已编码口令，他也无法对它们进行译码，因而不会影响到系统的安全性。

口令鉴别机制-口令文件



对加密口令的验证方法

口令管理

- 口令系统提供的安全性依赖于口令的保密性，这就要求:
  - 当用户在系统注册时，必须赋予用户口令
  - 用户口令必须定期更改
  - 系统必须维护一个口令数据库
  - 用户必须记忆自身的口令
  - 在系统认证用户时，用户必须输入口令

口令管理-系统管理员职责

- 初始化系统口令
- 初始口令分配
  - 生成的口令对系统管理员保密
  - 使口令暴露无效
  - 分级分配
- 口令更改认证
- 用户ID
- 用户ID重新生效

口令管理-用户职责

- 安全意识
- 更改口令

用户只允许更改自己的口令。为确保这一点，口令更改程序应要求用户输入其原始口令
更改口令发生在用户要求或口令过期的情况下

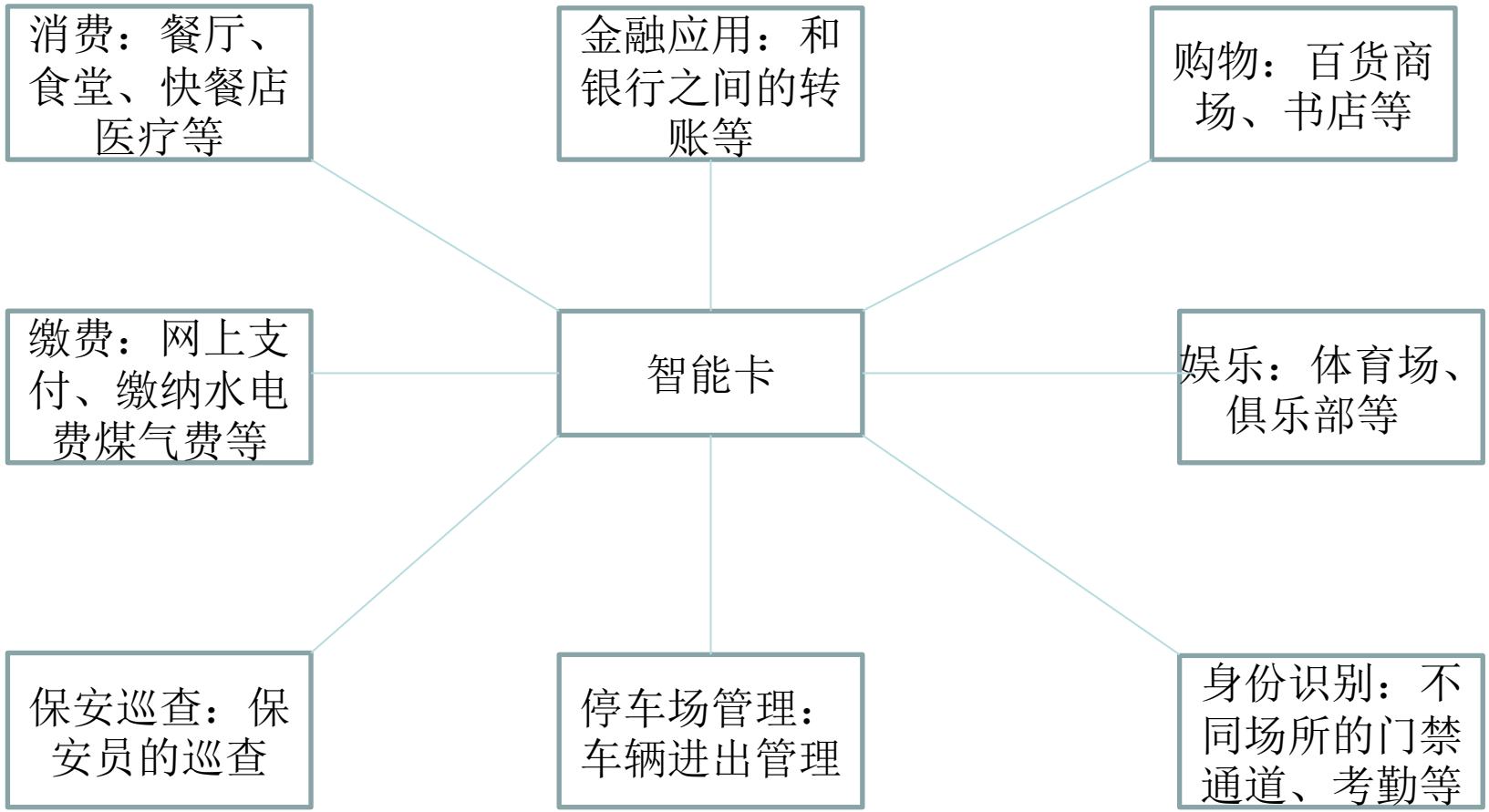
口令实现要点

- 口令的内部存储
  - 内部存储实行一定的访问控制和加密处理
  - 可以使用强制访问控制或自主访问控制
  - 无论采取何种访问控制机制，都应对存储的口令进行加密，因为访问控制有时可能被绕过。口令输入后应立即加密，存储口令明文的内存应在口令加密后立即删除，以后使用加密后的口令进行比较。
- 传输
- 尝试登录次数
- 用户安全属性
- 审计

2.密码验证（传统标识与鉴别技术）

- 传统的身份鉴别方法针对身份标识物品和身份标识知识
  - 个人钥匙、证件、用户名、密码等
- 智能卡

智能卡的广泛应用


消费：餐厅、食堂、快餐店医疗等
金融应用：和银行之间的转账等
购物：百货商场、书店等
缴费：网上支付、缴纳水电费煤气费等
智能卡
娱乐：体育场、俱乐部等
保安巡查：保安员的巡查
停车场管理：车辆进出管理
身份识别：不同场所的门禁通道、考勤等

3.生物标识与鉴别技术

- 通过计算机对人体固有的生理或行为特征进行个人身份鉴别。人的生理特征与生俱来，多为先天性
  - 常用的生物标识技术有脸像、虹膜、指纹、掌纹、声音、笔迹、步态等
- 生物标识与鉴别技术具有以下三大优点：
  - 1. 不容易被遗忘或丢失
  - 2. 不容易被伪造或被盗
  - 3. 可以随时携带，随时使用

生物标识与鉴别技术-生理特征介绍

- 生理特征介绍
  - 每个人所具有的唯一生理特征
    - 指纹，视网膜，声音，视网膜、虹膜、语音、面部、签名等
  - 指纹：
    - 一些曲线和分叉以及一些非常微小的特征；
    - 提取指纹中的一些特征并且存储这些特征信息：节省资源，快速查询；
  - 手掌、手型
    - 手掌有折痕，起皱，还有凹槽；
    - 还包括每个手指的指纹 ；
    - 人手的形状（手的长度，宽度和手指）表示了手的几何特征

生物标识与鉴别技术-生理特征介绍

- 生理特征介绍
  - 视网膜扫描
    - » 扫描眼球后方的视网膜上面的血管的图案；
  - 虹膜扫描
    - » 虹膜是眼睛中位于瞳孔周围的一圈彩色的部分；
    - » 虹膜有其独有的图案，分叉，颜色，环状，光环以及皱褶；
  - 语音识别
    - » 记录时说几个不同的单词，然后识别系统将这些单词混杂在一起，让他再次读出给出的一系列单词。
  - 面部扫描
    - » 人都有不同的骨骼结构，鼻梁，眼眶，额头和下颚形状。

生理特征的限制因素

- 第一，必须保证每个人的该项生物特征具有世界唯一性。
- 第二，生物特征必须提前识别，并存储在鉴别库中。
- 第三，由于生物特征会随着时间或者环境发生变化，鉴别库中的生物特征需要进行必要的更新，以保证较高的识别准确率。

4.可信计算基TCB

- 在安全操作系统中，可信计算基（TCB）要求先进行用户识别，之后才开始执行要TCB调节的任何其他活动。此外TCB要维持鉴别数据，不仅包括确定各个用户的许可证和授权的信息，而且包括为验证各个用户标识所需的信息（如口令等）。这些数据将由TCB使用，对用户识别进行鉴别，并确保由代表用户的活动所创建的TCB之外的主体的安全级和授权是受那个用户的许可证和授权支配的。TCB还必须保护鉴别数据，保证它不被任何非授权用户存取。

标识与鉴别（总结）

- 三类信息用作身份标识与鉴别
  - 用户知道的信息
  - 用户拥有的东西
  - 用户的生物特征

利用其中的任何一类都可进行身份认证，但若能利用多类信息，或同时利用三类中的不同信息，会增强认证机制的有效性和强壮性。

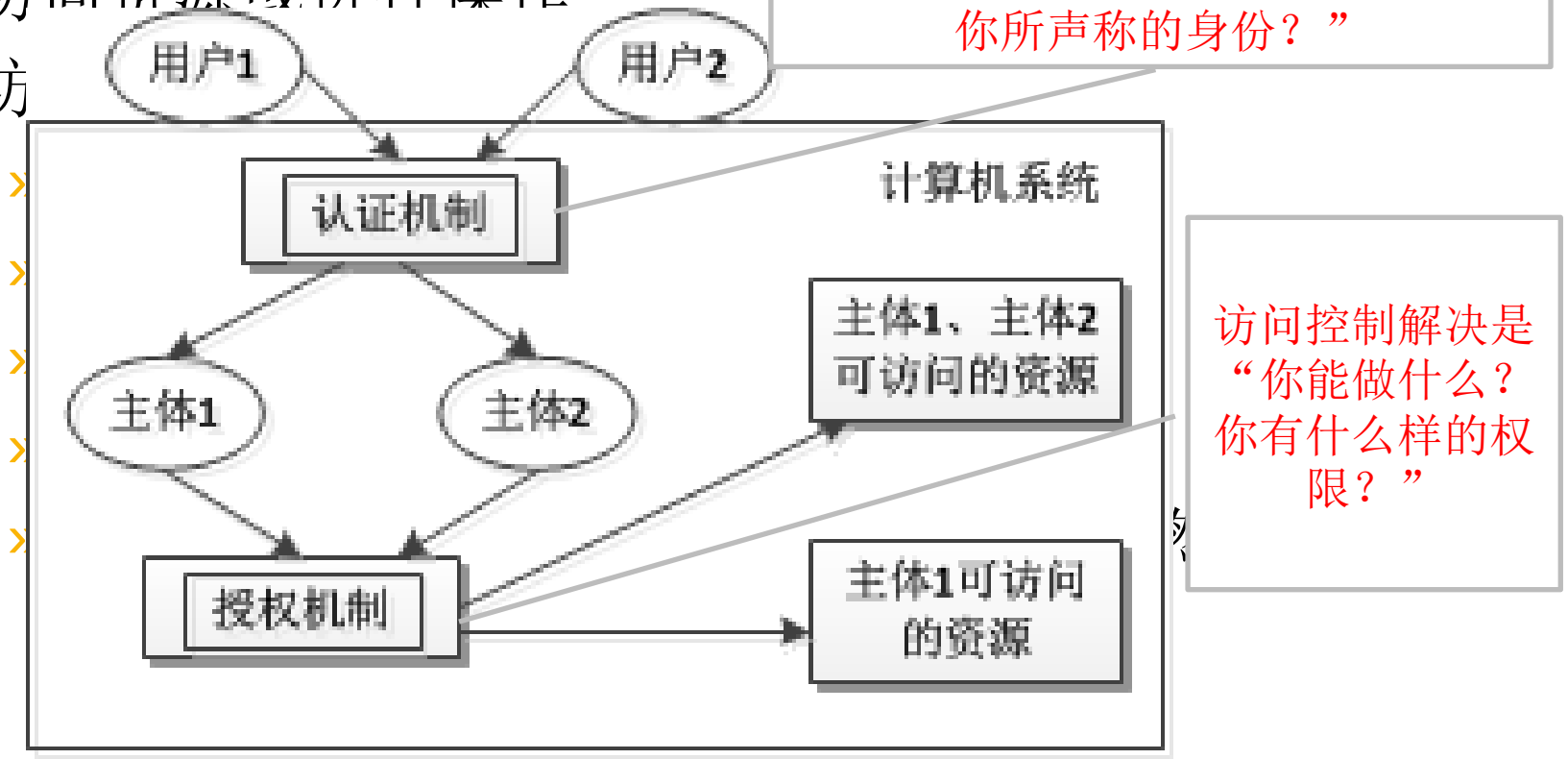
目录

访问控制的概念

- 什么是访问控制？
  - 访问控制是指主体依据某些控制策略或权限对客体本身或其资源进行的不同授权访问。访问控制是在身份认证的基础上，根据身份对提出的资源访问请求加以控制，是针对越权使用资源的现象进行防御的措施。访问控制是网络安全防范和保护的主要策略，它可以限制对关键资源的访问，防止非法用户或合法用户的不慎操作所造成的破坏。

认证和授权的关系

- 访问控制（认证和授权的关系）
  - 访问控制用来提供授权
  - 用户在通过身份鉴别后（认证），还需要通过授权才能访问资源或进行操作
  - 访



认证解决是“你是谁？你是否真的是你所声称的身份？”

访问控制解决是“你能做什么？你有什么样的权限？”

访问控制的目标和任务

- 基本目标：防止对任何资源（如计算资源、通信资源或信息资源）进行未授权的访问。从而使计算机系统在合法范围内使用；决定用户能做什么，也决定代表一定用户的程序能做什么。
- 基本任务：防止用户对系统资源的非法使用，保证对客体的所有直接访问都是被认可的。
- 三个要素：主体、客体、控制策略
- 措施
  - 确定要保护的资源
  - 授权：确定给予哪些主体访问哪些客体的权利
  - 确定访问权限：通常有读写执行删除追加等访问方式
  - 实施访问权限。

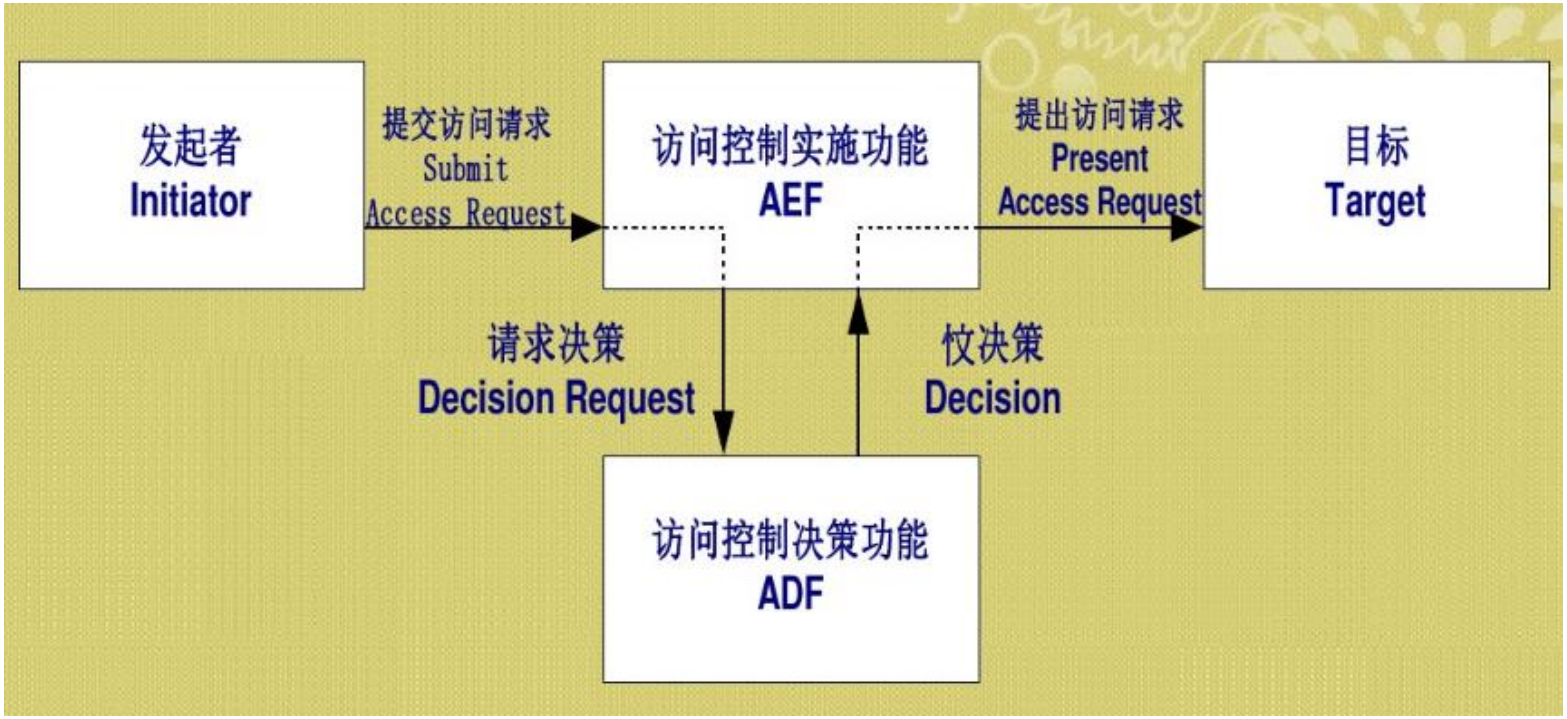
访问控制仅适用于计算机系统内的主体和客体，而不包括外界对系统的存取。控制外界对系统访问的技术是标识与鉴别。

访问控制的作用

- 访问控制对机密性、完整性起直接的作用
- 对于可用性，访问控制通过对以下信息的有效控制来实现：
  - （1）谁可以颁发影响网络可用性的网络管理指令
  - （2）谁能够滥用资源以达到占用资源的目的
  - （3）谁能够获得可以用于拒绝服务攻击的信息

访问控制的相关概念

- 客体（Object）：规定需要保护的资源，又称作目标（target）
- 主体（Subject）：或称为发起者（Initiator），是一个主动的实体，规定可以访问该资源的实体（通常指用户或代表用户执行的程序）
- 授权（Authorization）：规定可对该资源执行的动作（如读、写、执行或拒绝访问）。



访问控制的相关概念-主体属性

- 主体属性：用户特征，是系统用来决定访问控制的常用因素，一个用户的任何一种属性都可作为访问控制决策点，一般系统访问控制策略中常用的用户属性有：
  - a)用户ID/用户组ID：
  - b)用户访问许可级别：
  - c)用户需知属性：
  - d)角色：
  - e)权能列表：

访问控制的相关概念-客体属性

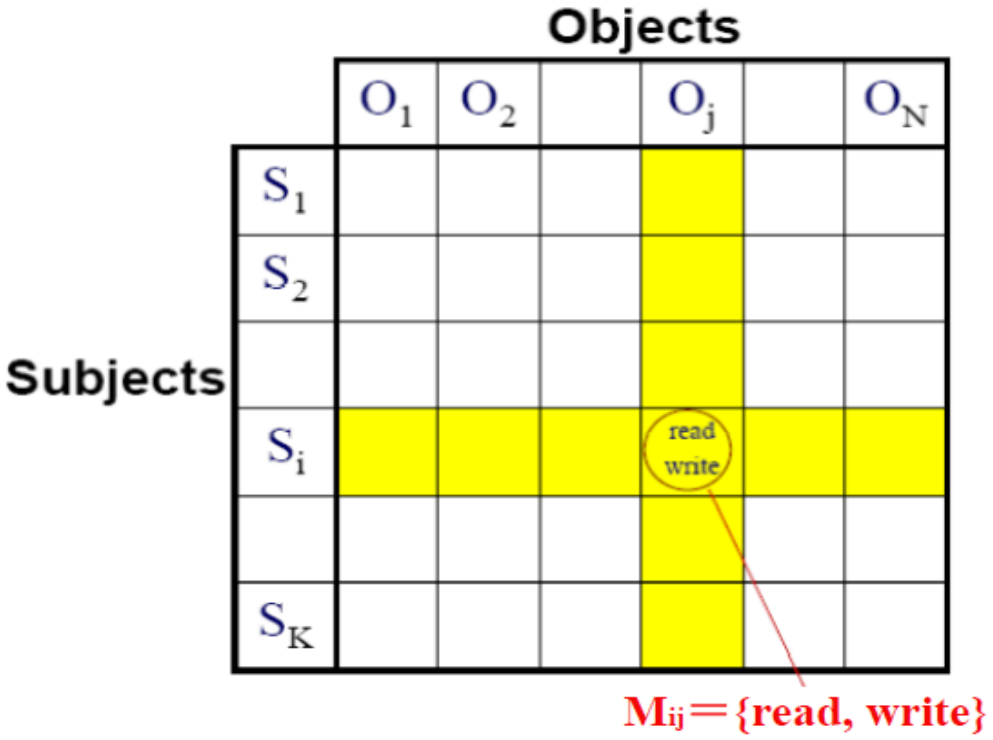
- 客体属性：与系统内客体相关联的属性也作为访问控制策略的一部分，客体安全属性有：
  - a)敏感性标记：信息按“安全等级”进行分类，如“公开信息”、“机密信息”、“秘密信息”、“绝密信息”；
  - b)还可将系统内的信息按非等级分类，进行模拟人力资源系统的划分，称“范畴”，如参谋部、作战部、后勤部等，
  - c)系统内信息的敏感性标记由等级与非等级两部分组成：敏感性级别和范畴。

访问控制的实现机制和方法

- 一般实现机制
  - 基于访问控制属性→访问控制表/矩阵
  - 基于用户和资源分级（“安全标签”） →多级访问控制
- 常见实现方法
  - 访问控制表ACL（Access Control Lists）
  - 访问能力表CL（Capabilities Lists）
  - 授权关系表

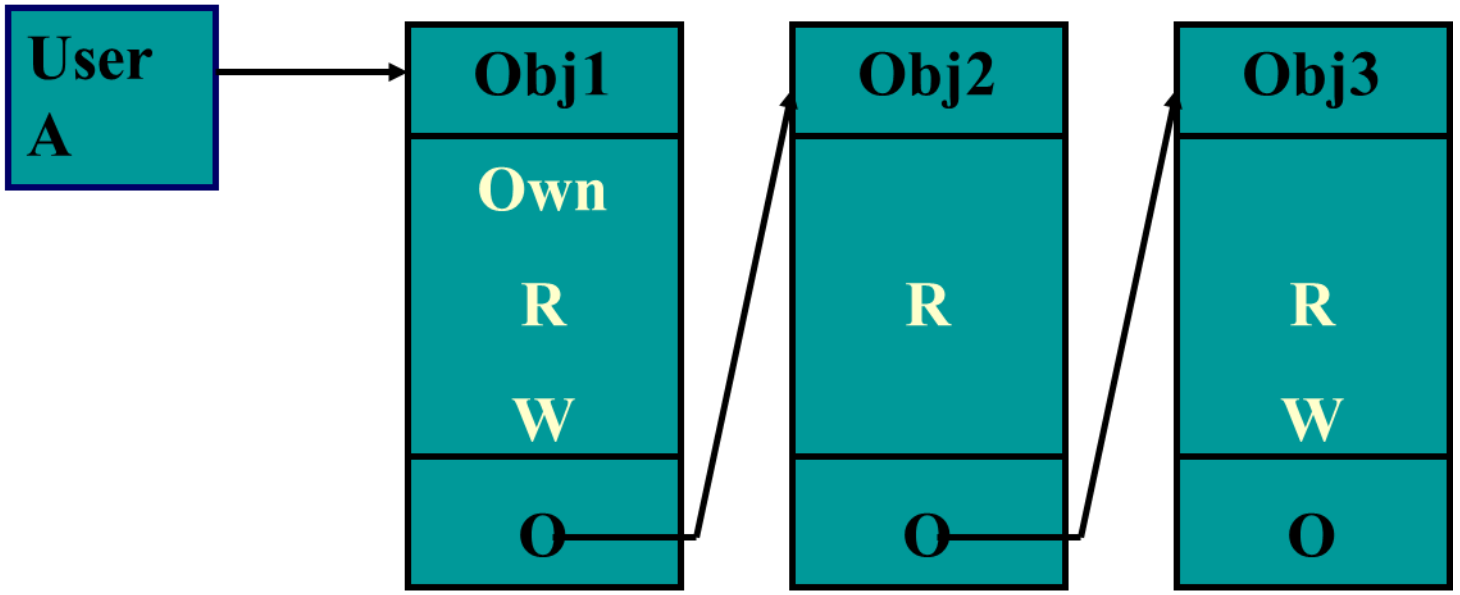
访问控制矩阵

- 任何访问控制策略最终均可被模型化为访问矩阵形式：由访问控制矩阵提供的信息必须以某种形式存放在系统中。
  - 访问矩阵中的每行表示一个主体
  - 每列则表示一个受保护的客体
  - 而矩阵中的元素，则表示主体可以对客体的访问模式



访问能力表CL

- 访问能力表(CL)



每个主体都附加一个该主体可访问的客体的明细表。

访问控制表ACL

- 存取控制表可以决定任何一个特定的主体是否可对某一个客体进行访问。它是利用在客体上附加一个主体明细表的方法来表示访问控制矩阵的。
- 表中的每一项包括主体的身份以及对该客体的访问权

访问控制表ACL

优化的访问控制表ACL

- 把用户按其所属或其工作性质进行分类，构成相应的组（group）
- 并设置一个通配符“*”，代表任何组名或主体标识符

文件X		
ID1	GROUP5	rwx
*	GROUP5	--x
ID3	*	---
*	*	r--

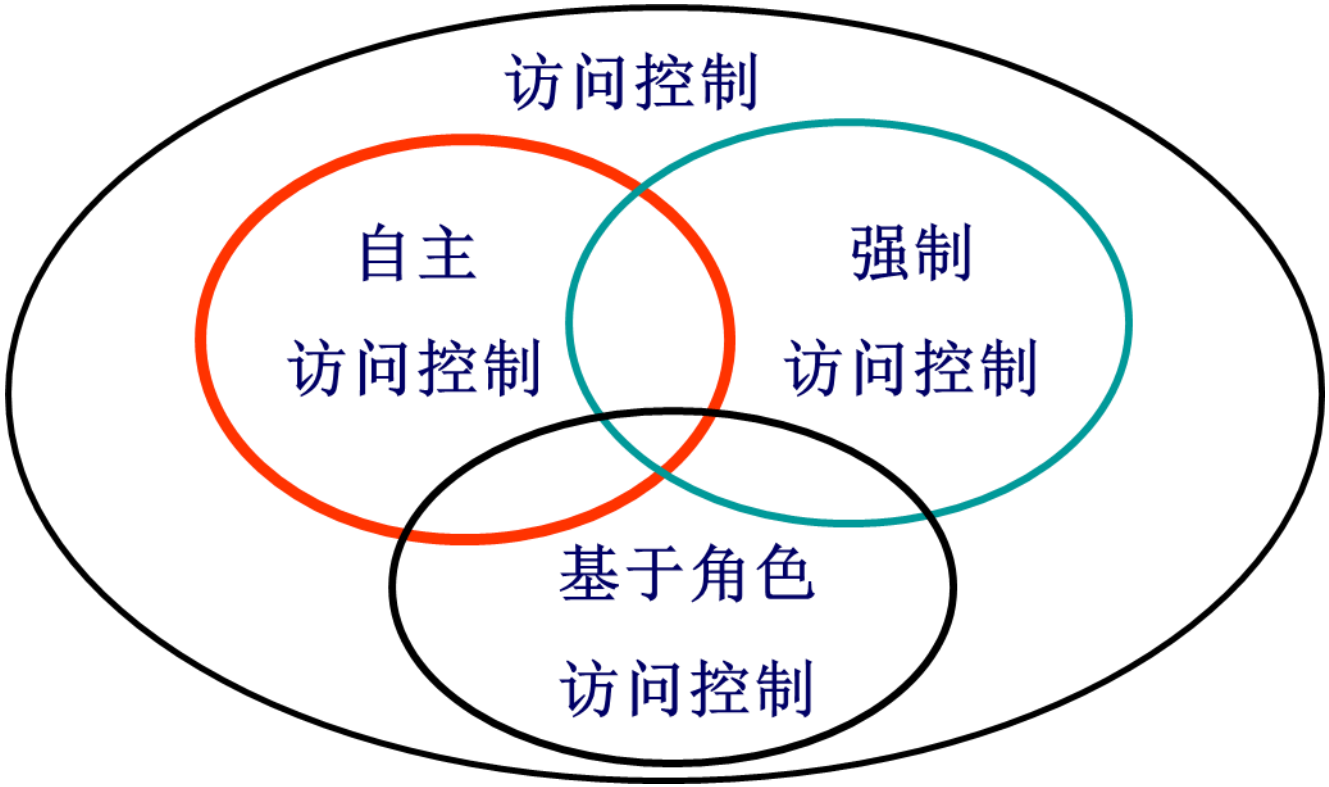
优化的访问控制表ACL

ACL、CL访问方式的对比

- 鉴别方面：二者需要鉴别的实体不同
- 保存位置不同
- 访问权限传递
  - ACL困难，CL容易
- 访问权限回收
  - ACL容易，CL困难
- 多数集中式操作系统使用ACL方法或类似方式
- 由于分布式系统中很难确定给定客体的潜在主体集，在现代OS中CL也得到广泛应用

三种常用的访问控制机制

- 自主访问控制DAC（Discretionary Access Control）
- 强制访问控制MAC（Mandatory Access Control ）
- 基于角色的访问控制RBAC（Role Based Access Control）



访问控制——自主访问控制

- 自主访问控制
  - 定义：是一种最为普遍的访问控制手段，其依据是用户的身份和授权，并为系统中的每个用户（或用户组）和客体规定了用户允许对客体进行访问的方式。
  - 自主：这里所谓的自主决定权是指主体可以自主地将访问权，或访问权的某个子集授予其它主体。
  - 特点：根据主体的身份和授权来决定访问模式。
  - 缺点：安全性最低，因为信息在移动过程中其访问权限关系会被改变。如用户A可将其对目标O的访问权限传递给用户B，从而使不具备对O访问权限的B可访问O。

基于行的自主访问控制机制

- 所谓基于行的自主访问控制是在每个主体上都附加一个该主体可访问的客体的明细表。根据表中信息的不同可以分为以下三种形式：
  - 能力表（CL：Capabilities List）：权限字
  - 前缀表(Prefixes)：包括受保护的客体名和主体对它的访问权限
  - 口令（Password）：每个客体有一个

基于列的自主访问控制机制

- 所谓基于列的访问控制是指按客体附加一份可访问它的主体的明细表。基于列的访问控制可以有两种方式：
  - 保护位（protection bits）：对客体的拥有者及其他主体、主体组，规定的对该客体访问模式的集合
  - 访问控制表（Access Control List，ACL）：对某个特定资源制定任意用户的访问权限。

保护位

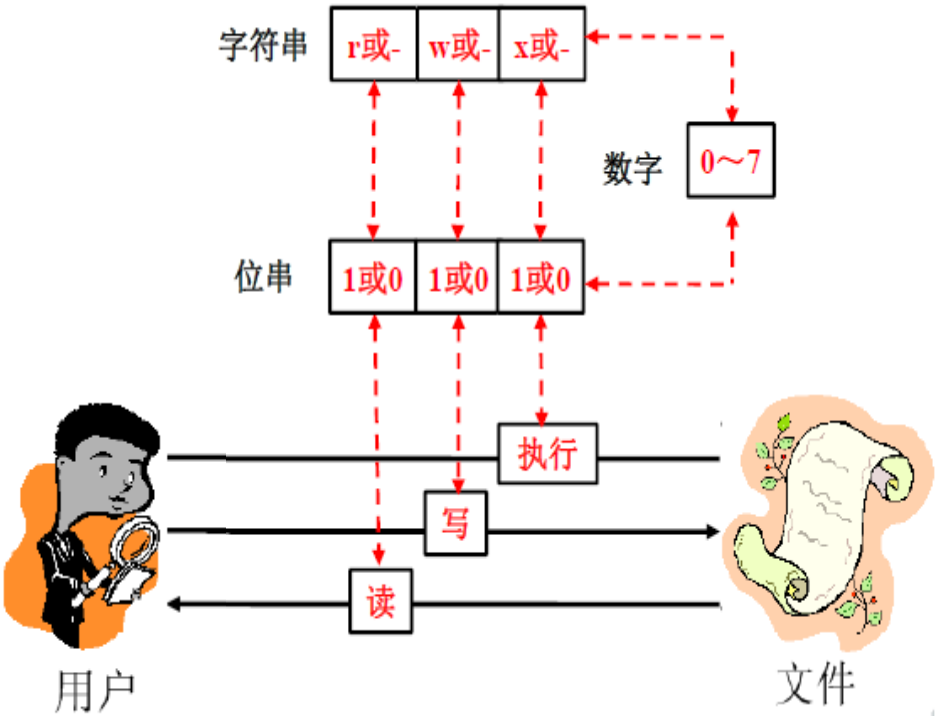
- 保护位对所有的主体、主体组（用户、用户组）以及该客体（文件）的拥有者，规定了一个访问模式的集合。保护位机制是一种简单易行的自主访问控制方式，能在一定程度上表达访问控制矩阵。
  - UNIX系统采用了保护位的方法。在Unix系统中，用户组是具有相似功能特点的用户集合。某客体的拥有者实际就是生成客体的主体，它对客体的所有权仅能通过超级用户特权来改变。
  - 拥有者（超级用户除外）是唯一能够改变客体保护位的主体。一个用户可能属于多个静态的用户组，但是在某个时刻，一个用户只能属于一个活动的用户组。用户组及拥有者的权限都体现在客体的保护位中。

自主访问控制实现举例

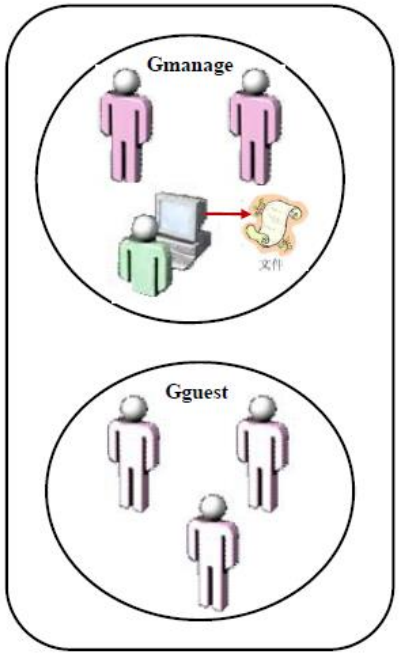
- 1、“拥有者/同组用户/其他用户”模式
- 2. “访问控制列表ACL”和“拥有者/同组用户/其他用户”结合模式

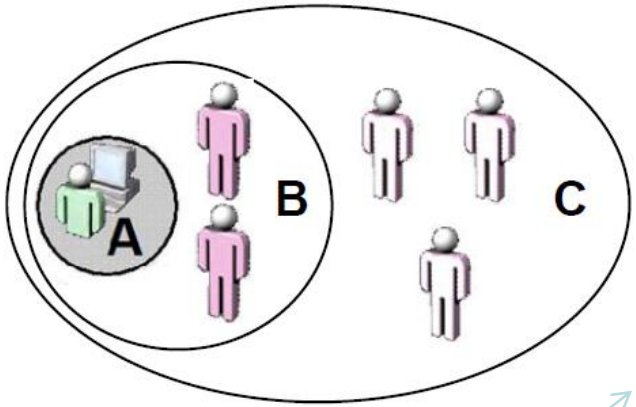
基于权限位的访问控制

- 读、写、执行——三位二进制



基于权限位的访问控制- 用户的划分与访问控制





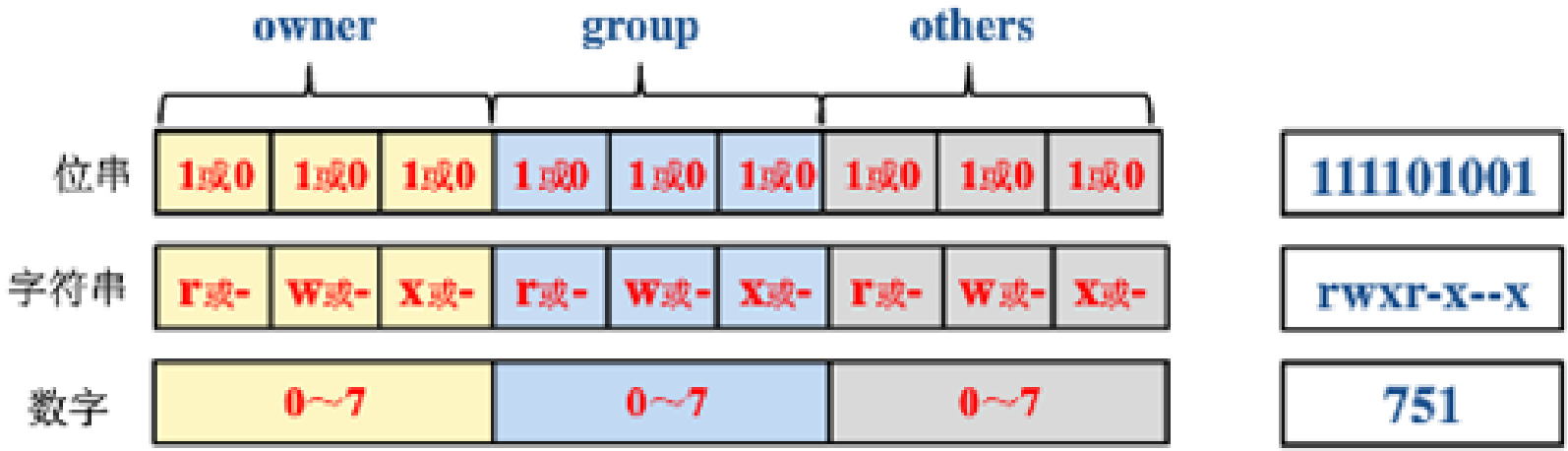
A：属主（ower）
B：属组（group）
C：其它（other）

此客体的拥有者对它的访问权限

Owner同组用户对此客体的访问权限

其他用户对此客体的访问权限

“属主/属组/其它”访问控制方式



自主访问控制实现实例

- 自主访问控制实现举例
  - 例：设在某LINUX操作系统中，部分用户组的配置信息如下：

    grp1:x:300:usr1,usr3,usr4

    grp2:x:301:usr5,usr6,usr7,usr8,usr9

    系统中部分文件的权限配置信息如下：

    rw-r-x--x　usr1　grp1　……file1

    r---w---x　usr5　grp2　……file2

    请问，用户usr1、usr2和usr5可以对文件file1执行什么操作？　用户usr6可以对文件file2执行什么操作？

自主访问控制实现实例

- 自主访问控制实现举例
  - 例：设在某LINUX操作系统中，部分用户组的配置信息如下：

    grp1:x:300:usr1,usr3,usr4

    grp2:x:301:usr5,usr6,usr7,usr8,usr9

    系统中部分文件的权限配置信息如下：

    rw-r-x--x　usr1　grp1　……file1

    r---w---x　usr5　grp2　……file2

请问，用户usr1、usr2和usr5可以对文件file1执行什么操作？　用户usr6可以对文件file2执行什么操作？

Usr1对file1可以执行rw操作；usr2对file1可以执行x操作；usr5对file1可以执行x操作。
Usr6可以对文件file2执行w操作。

自主访问控制实现实例

- 自主访问控制实现举例
  - 思考：设在某LINUX操作系统中，自主访问控制机制的部分相关文件配置信息

    如下：

    grp2:x:301:usr5,usr6,usr7,usr8,usr9

    rw-r--r--　usr5　grp2　……file3

    请问，用户usr6是否可能对文件file3进行写操作？

自主访问控制实现实例

- 自主访问控制实现举例
  - 思考：设在某LINUX操作系统中，自主访问控制机制的部分相关文件配置信息

    如下：

    grp2:x:301:usr5,usr6,usr7,usr8,usr9

    rw-r--r--　usr5　grp2　……file3

    请问，用户usr6是否可能对文件file3进行写操作？

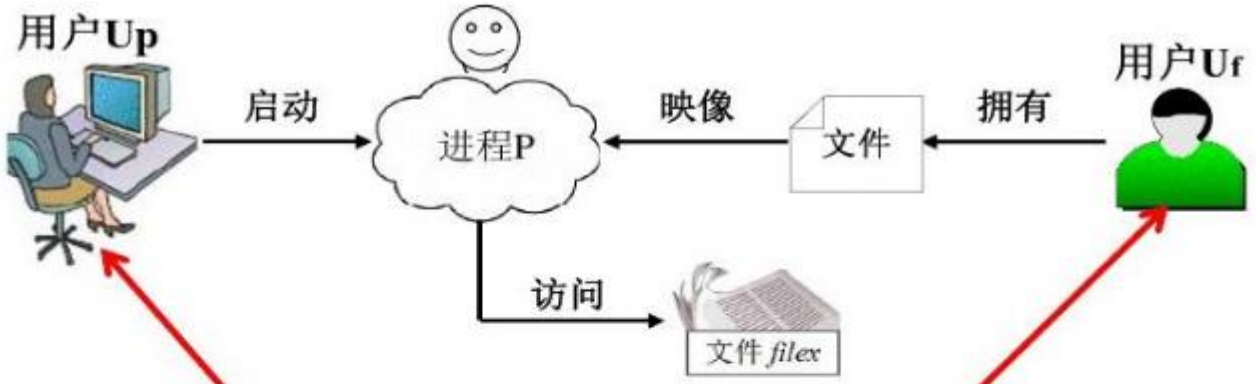
Usr6不能对文件file3进行写操作

基于权限位的访问控制-setuid/setgid

- Linux系统中每个进程都有2个ID
  - 用户ID：表示进程的创建者（属于哪个用户创建），可以通过getuid()取得
  - 有效用户ID（EUID）：表示进程对于文件和资源的访问权限（具备等同于哪个用户的权限），可以通过geteuid()取得
- setuid（设置用户标识）是允许用户以文件所有者的权限执行一个程序的权限位
- setgid（设置组标识）是允许用户以用户组成员的权限执行一个程序的权限位

基于权限位的访问控制-setuid/setgid

- 进程运行时能够访问哪些资源或文件，不取决于进程文件的属主属组，而是取决于运行该命令的用户身份的uid/gid，以该身份获取各种系统资源



- 真实用户属性：RUID+RGID
- 有效用户属性：EUID+EGID
- 进程访问判定时：g(进程) → EUID+EGID

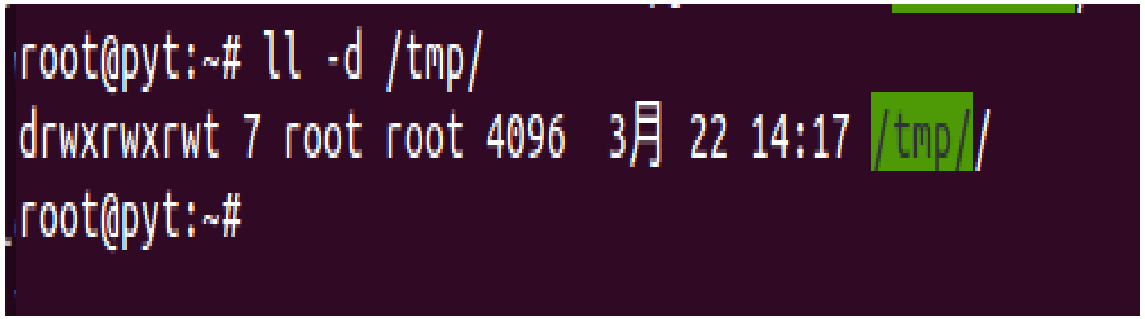
基于权限位的访问控制-setuid/setgid

- 对一个属主为root的可执行文件，如果设置了SUID位，则其他所有普通用户都将可以以root身份运行该文件，获取相应的系统资源
- 可以简单地理解为让普通用户拥有可以执行“只有root权限才能执行”的特殊权限
- setuid/setgid的作用是让执行该命令的用户以该命令拥有者的权限去执行，比如普通用户执行passwd时会拥有root的权限，这样就可以修改/etc/passwd这个文件了。它的标志为：s，会出现在x的地方，而setgid的意思和它是一样的，即让执行文件的用户以该文件所属组的权限去执行

```
root@pyt:~# ll /usr/bin/passwd
-rwsr-xr-x 1 root root 45420  2月 17  2014 /usr/bin/passwd*
root@pyt:~#
```

基于权限位的访问控制-setuid/setgid

- /tmp是系统的临时文件目录，所有的用户在该目录下拥有所有的权限，也就是说在该目录下可以任意创建、修改、删除文件，那如果用户A在该目录下创建了一个文件，用户B将该文件删除了，这种情况我们是不能允许的。为了达到该目的，就出现了stick bit(粘滞位)的概念。它是针对目录来说的，如果该目录设置了stick bit(粘滞位)，则该目录下的文件除了该文件的创建者和root用户可以删除和修改/tmp目录下的stuff，别的用户均不能动别人的，这就是粘滞位的作用



基于权限位的访问控制-setuid/setgid

- 如何设置UID、GID、STICK_BIT

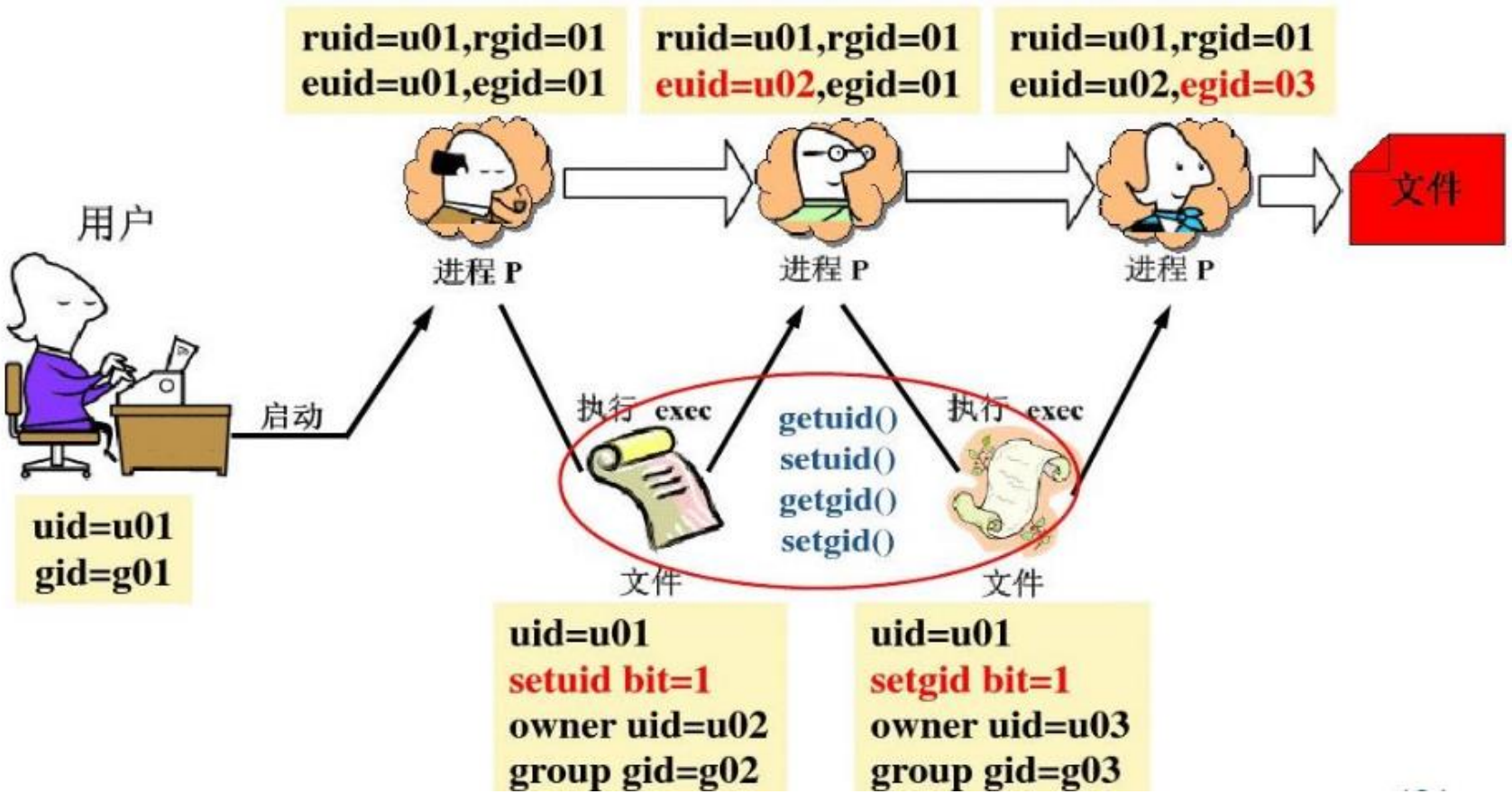
SUID：置于 u 的 x 位，原位置有执行权限，就置为 s，没有x为 S .
chmod u+s xxx # 设置setuid权限
chmod 4551 file // 权限： r-sr-x—x.使用chmod命令的数字模式设置setuid的方法是，在3个权限位的前面加数字4

SGID：置于 g 的 x 位，原位置有执行权限，就置为 s，没有了为 S .
chmod g+s xxx # 设置setgid权限
chmod 2551 file // 权限： r-xr-s—x.使用chmod命令的数字模式设置setgid的方法是，在3个权限位的前面加数字2

STICKY：粘滞位，置于 o 的 x 位，原位置有执行权限，就置为 t ，否则为T .
chmod o+t xxx # 设置stick bit权限，针对目录
chmod 1551 file // 权限： r-xr-x—t.使用chmod命令的数字模式设置sticky的方法是，在3个权限位的前面加数字1
使用chmod命令移除setuid和setgid权限位的方法是，在3个权限位的前面加数字0

基于权限位的访问控制-setuid/setgid

- 基于权限位的访问控制-setuid/setgid(例)
  - 进程有效身份变化



基于权限位的访问控制-setuid/setgid(例)

progf1:	progf2:	progf3:

printf("China");
exec("progf2");
printf("America");
return;
	printf("England");
	exec("progf3");
	printf("Canada");
	return;
		printf("Australia");
		return;

组配置信息：grp2:x:301:usr5,usr6,usr7,usr8,usr9

系统中部分文件的权限配置信息如下：

```
--x--x--x    usr1   grp1   ..... progf1
--x--s--x    usr6   grp2   ..... Progf2
--s--x--x    usr5   grp2   ..... Progf3
rwxr-xr--    usr5   grp2   ..... filex
```

- 假设：用户usr1 执行程序progf1启动进程P。
- 问题：进程P显示China时，对文件filex拥有什么访问权限？
- 结果：读（r）权限。

基于权限位的访问控制-setuid/setgid(例)

```
progf1:
printf("China");
exec("progf2");
printf("America");
return;
```

```
progf2:
printf("England");
exec("progf3");
printf("Canada");
return;
```

```
progf3:
printf("Australia");
return;
```

```
组配置信息: grp2:x:301:usr5,usr6,usr7,usr8,usr9
系统中部分文件的权限配置信息如下:
--x--x--x   usr1  grp1  ..... progf1
--x--s--x   usr6  grp2  ..... Progf2
--s--x--x   usr5  grp2  ..... Progf3
rwxr-xr--   usr5  grp2  ..... Filex
```

- 假设：用户usr1 执行程序progf1启动进程P。
- 问题：进程P显示England时，对文件filex拥有什么访问权限？
- 结果：读、执行（rx）权限。

基于权限位的访问控制-setuid/setgid(例)

```
progf1:
printf("China");
exec("progf2");
printf("America");
return;
```

```
progf2:
printf("England");
exec("progf3");
printf("Canada");
return;
```

```
progf3:
printf("Australia");
return;
```

```
组配置信息: grp2:x:301:usr5,usr6,usr7,usr8,usr9
系统中部分文件的权限配置信息如下:
--x--x--x   usr1  grp1  ..... progf1
--x--s--x   usr6  grp2  ..... Progf2
--s--x--x   usr5  grp2  ..... Progf3
rwxr-xr--   usr5  grp2  ..... Filex
```

- 假设：用户usr1 执行程序progf1启动进程P。
- 问题：进程P显示Australia时，对文件filex拥有什么访问权限？
- 结果：读、写、执行（rwx）权限。

基于权限位的访问控制-setuid/setgid(例)

已知三个可执行程序的伪代码如下：

```
Progf1:
        printf( “A” );
        exec( “progf2” );
        printf( “D” );
Progf2:
        printf( “B” );
        exec( “progf3” );
        printf( “E” );
        return;
Progf3:
        printf( “C” );
        return;
```

设在某个UNIX操作中，部分用户组的配置信息如下：

siselab_ms:x:301:zys,zbh,kankan,xiaoli,liuxing

系统中部分文件的权限配置信息如下：

--x--x--x	swm	sisefellow	…	progf1
--s--x--x	zys	siselab_ms	…	progf2
--x--s--x	zbh	siselab_ms	…	progf3
rw-r-----	zys	siselab_ms	…	filex

用户swm执行程序progf1启动了进程P，请问：

(1) 进程P在显示A时，对文件filex拥有什么访问权限？

(2) 进程P在显示B时，对文件filex拥有什么访问权限？

(3) 进程P在显示C时，对文件filex拥有什么访问权限？

基于权限位的访问控制-setuid/setgid(例)

已知三个可执行程序的伪代码如下：

```
Progf1:
        printf( “A” );
        exec( “progf2” );
        printf( “D” );
Progf2:
        printf( “B” );
        exec( “progf3” );
        printf( “E” );
        return;
Progf3:
        printf( “C” );
        return;
```

设在某个UNIX操作中，部分用户组的配置信息如下：

siselab_ms:x:301:zys,zbh,kankan,xiaoli,liuxing

系统中部分文件的权限配置信息如下：

--x--x--x	swm	sisefellow	…	progf1
--x--s--x	zys	siselab_ms	…	progf2
--s--x--x	zbh	siselab_ms	…	progf3
rw-r-----	zys	siselab_ms	…	filex

用户swm执行程序progf1启动了进程P，请问：

(1) 进程P在显示A时，对文件filex拥有什么访问权限？

(2) 进程P在显示B时，对文件filex拥有什么访问权限？

(3) 进程P在显示C时，对文件filex拥有什么访问权限？

已知三个可执行程序的伪代码如下：

```
Progf1:
        printf( “A” );
        exec( “progf2” );
Progf2:
        i=getuid();
        setuid(i);
        printf( “B” );
        exec( “progf3” );
Progf3:
        i=getgid();
        setgid(i);
        printf( “C” );
```

设在某个UNIX操作中，部分用户组的配置信息如下：

siselab_ms:x:301:zys,zbh,kankan,xiaoli,liuxing

系统中部分文件的权限配置信息如下：

--x--x--x	swm	sisefellow	…	progf1
--x--x--x	zys	siselab_ms	…	progf2
--x--x--x	zbh	siselab_ms	…	progf3
rw-r-----	zbh	siselab_ms	…	filex

用户swm执行程序progf1启动了进程P，请问：

(1) 进程P在显示A时，对文件filex拥有什么访问权限？

(2) 进程P在显示B时，对文件filex拥有什么访问权限？

(3) 进程P在显示C时，对文件filex拥有什么访问权限？

自主访问控制面临的问题

- 自主访问控制面临的最大问题：
  - 在自主访问控制中，具有某种访问权的主体能够自行决定将其访问权直接或间接地转交给其他主体。自主访问控制允许系统的用户对于属于自己的客体，按照自己的意愿，允许或者禁止其他用户访问。
  - 在DAC系统中，主体的拥有者负责设置访问权限。也就是说，主体拥有者对访问的控制有一定权利。但正是这种权利使得信息在移动过程中，其访问权限关系会被改变。如用户A可以将其对客体目标O的访问权限传递给用户B，从而使不具备对O访问权限的B也可以访问O，这样做很容易产生安全漏洞，所以DAC的安全级别很低。

强制访问控制

- 根据客体中信息的敏感标记和访问敏感信息的主体的访问级对客体访问实行限制
- 用户的权限和客体的安全属性都是固定的
- 强制：就是安全属性由系统管理员人为设置，或由操作系统自动地按照严格的安全策略与规则进行设置，用户和他们的进程不能修改这些属性。
- 强制访问控制：访问发生前，系统通过比较主体和客体的安全属性来决定主体能否以他所希望的模式访问一个客体。

强制访问控制

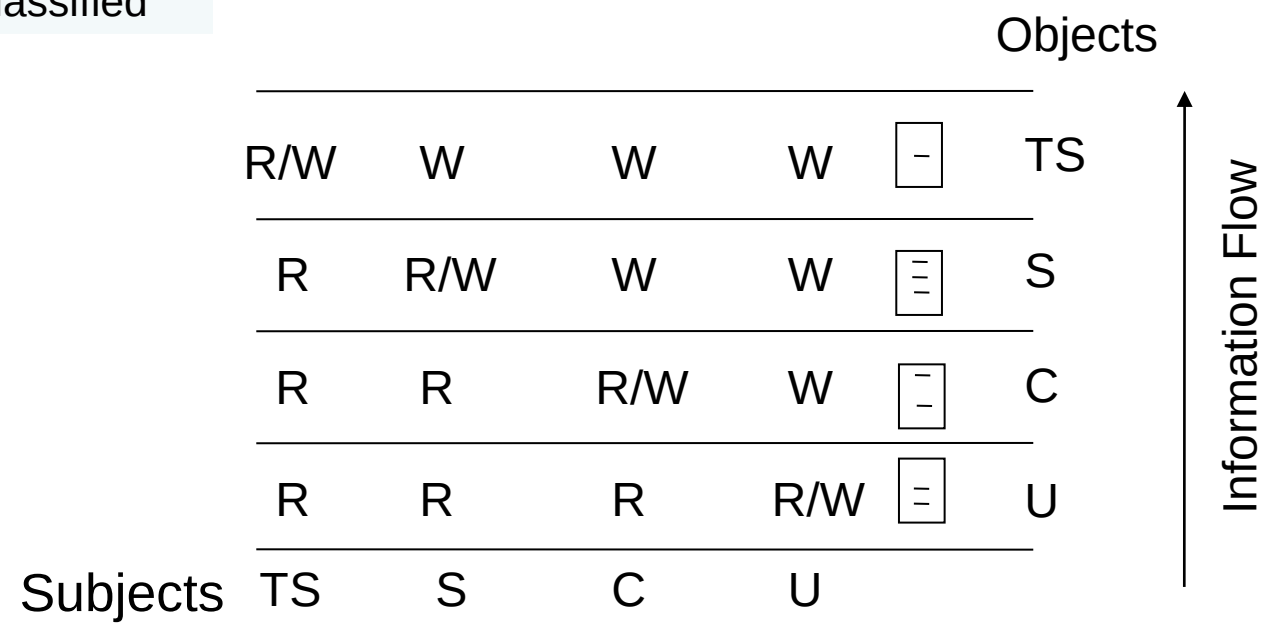
- 强制访问控制系统为所有的主体和客体指定安全级别，比如绝密级、秘密级和无密级。不同级别标记了不同重要程度和能力的实体。不同级别的主体对不同级别的客体的访问是在强制的安全策略下实现的。

- 强制访问控制机制中，系统为每个进程、每个文件以及每个IPC（Inter-Process Communication）客体赋予了相应的安全属性，由系统管理员或操作系统自动的按照严格的规则来设置。 其访问控制关系分为：**上读/下写**（完整性）， **下读/上写**（机密性）。
- 机密性要求防止了信息泄露给未授权的用户；完整性要求防止了未授权用户对信息的修改。

强制访问控制的安全属性

- 规定：一般安全属性分为四个级别

密级	英文
绝密TS	Top Secret
机密C	Confidential
秘密S	Secret
无密级U	Unclassified



四种强制访问控制策略

- 具有如下四种强制访问控制策略：
  - 下读：用户级别大于文件级别的读操作；
  - 上写：用户级别小于文件级别的写操作；
  - 下写：用户级别大于文件级别的写操作；
  - 上读：用户级别小于文件级别的读操作；
- 这些策略保证了信息流的单向性，上写-下读方式则保证了信息的安全性，上读-下写方式保证数据的完整性。

多级安全策略（MLS）

- 强制访问控制——MLS（多级安全策略）
  - MLS是军事安全策略的数学描述
  - 军事安全策略的目的是防止用户取得他不应得到的密级较高的信息，因此对系统的主体和客体分别赋予与其身份相对称的安全属性的外在表示——安全标签（安全级）
    - 保密级别：可分为公开、秘密、机密、绝密等
    - 范畴集：该安全级涉及的领域，如人事处，财务处等

多级安全策略（MLS）

- 强制访问控制——MLS（多级安全策略）
  - 安全级包括一个保密级别，范畴集包括任意多个范畴。安全级通常写作保密级别后随一范畴集的形式。如下：

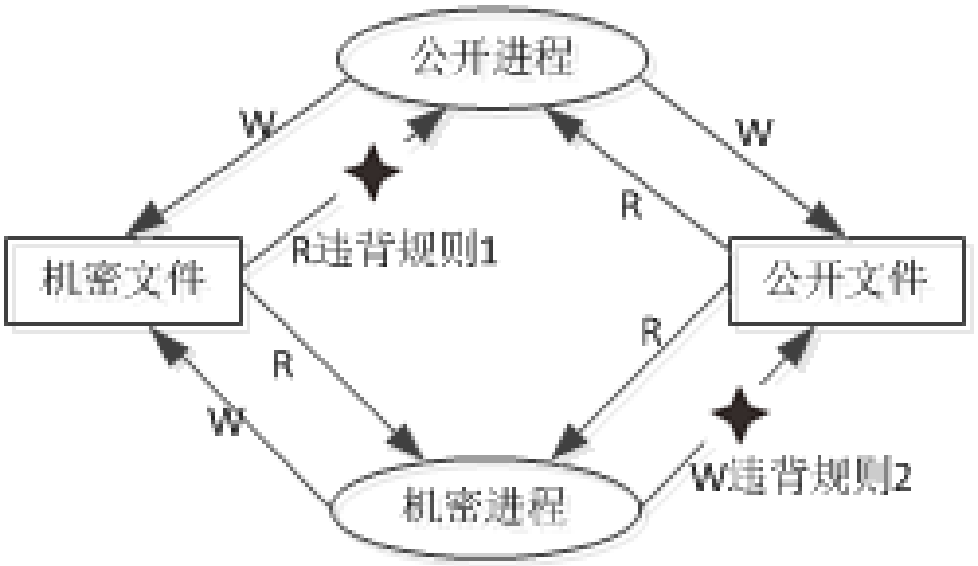
机密：人事处，财务处，科技处

多级安全策略（MLS）

- 强制访问控制——MLS（多级安全策略）
  - 安全级中的保密级别是线性排列的
    - 公开 < 秘密 < 机密 < 绝密
  - 两个安全级中的关系有以下几种：
    - 1. 第一安全级支配第二安全级，即第一安全级的级别不小于第二安全级的的级别，第一安全级的范畴集包含第二安全级的范畴；
    - 2. 第一安全级支配于第二安全级，即第二安全级的级别不小于第一安全级的的级别，第二安全级的范畴集包含第一安全级的范畴；
    - 3. 第一安全级等于第二安全级，即第一安全级的级别等于第二安全级的的级别，第一安全级的范畴集等于第二安全级的范畴；
    - 4. 两个安全级无关，即第一安全级的范畴集不包含第二安全级的范畴；第二安全级的范畴集不包含第一安全级的范畴；

强制访问控制实例

- 强制访问控制实例——Multics方案
  - 用户和文件（包括目录文件）都有相应的安全级
  - 用户对文件的访问遵循下述安全策略：
    - » 仅当用户的安全级别不低于文件的安全级别时，用户才可以读文件
    - » 仅当用户的安全级别不高于文件的安全级别时，用户才可以写文件



强制访问控制的作用

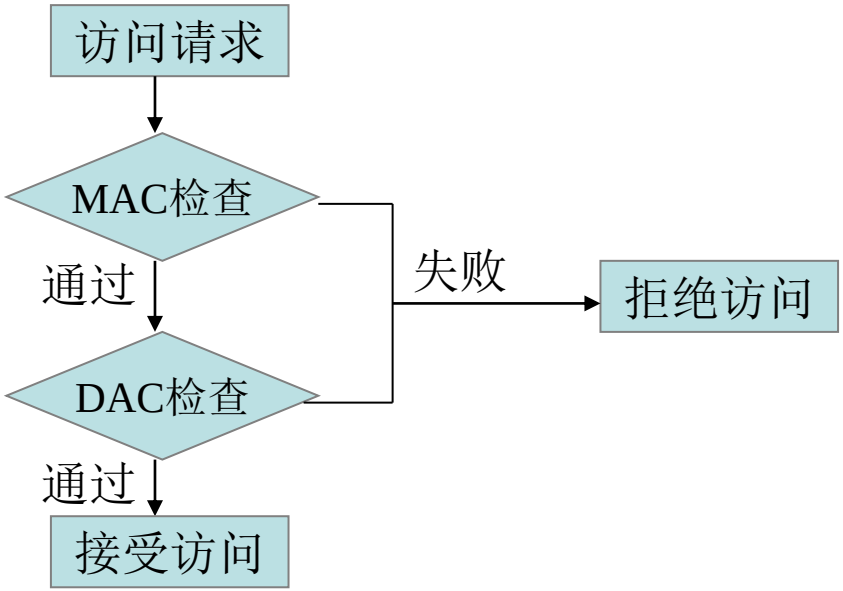
- 解决特洛伊木马的一个有效方法是使用强制存取控制机制。在强制存取控制的情况下，对于违反强制存取控制的特洛伊木马，可以防止它取走信息。强制存取控制能阻止正在机密安全级上运行的进程中的特洛伊木马把机密信息写入一个公开的文件里。

强制访问控制的缺点

- 不灵活
- 级别定义繁琐，工作量大，管理不便
- 有些场合无法定义合理的级别，主体信任级别和客体安全级别和现实情况不能一致
- 过于强调保密性，对系统连续工作能力和可管理性考虑不足

MAC和DAC

- MAC和DAC是两种不同类型的存取控制机制，常结合起来使用



- MAC用于将系统中的信息分密级和类进行管理，适用于政府部门、军事和金融等领域。

基于角色的访问控制

- 现实社会中，上述访问控制方式表现出很多弱点，不能满足实际需求。主要问题在于：
  - （1）同一用户在不同的场合需要以不同的权限访问系统，按传统的做法，变更权限必须经系统管理员授权修改，因此很不方便。
  - （2）当用户量大量增加时，按每用户一个注册账号的方式将使得系统管理变得复杂、工作量急剧增加，也容易出错。
  - （3）传统访问控制模式不容易实现层次化管理。即按每用户一个注册账号的方式很难实现系统的层次化分权管理，尤其是当同一用户在不同场合处在不同的权限层次时，系统管理很难实现。除非同一用户以多个用户名注册。
  - （4）自主式太弱、强制式太强
  - （5）二者工作量大，不便管理

基于角色的访问控制介绍

- RBAC介绍：
  - 由IST的Ferraiolo等人在90年代提出的
  - 96年提出一个较完善的基于角色的访问控制参考模型RBAC96
  - 角色控制与自主式/强制式控制的区别：
    - 角色控制相对独立，根据配置可使某些角色接近DAC，某些角色接近MAC。

基于角色的访问控制基本思想

- 角色访问控制的基本思想
  - 授权给用户的访问权限，通常由用户在一个组织中担当的角色来确定。以角色作为访问控制的主体。
    - » 用户以什么样的角色对资源进行访问，决定了用户拥有的权限以及可执行何种操作。
- 角色：指一个或一群用户在组织内可执行的操作的集合。角色就充当着主体（用户）和客体之间的关联的桥梁。这是与传统的访问控制策略的最大区别所在。
- 角色与组的区别
  - 组：一组用户的集合
  - 角色：一组用户的集合 + 一组操作权限的集合

基于角色的访问控制基本概念

- 基于角色的访问控制模式中，用户不是自始至终以同样的注册身份和权限访问系统，而是以一定的角色访问，不同的角色被赋予不同的访问权限，系统的访问控制机制只看到角色，而看不到用户。
- 在RBAC系统中，要求明确区分权限（authority）和职责（responsibility）这两个概念。职责之间的不同是通过不同的角色来区分的。RBAC的功能相当强大，适用于许多类型（从政府机构到商业应用）的用户需求。Netware、Windows NT、Solaris和SeLinux等操作系统中都采用了类似的RBAC技术作为存取控制手段。

基于角色的访问控制相关概念

- 相关概念
  - 用户（user）
    - » 访问计算机资源的主体。用户集合为U
  - 角色（role）
    - » 一种岗位，代表一种资格、权利和责任。角色集合为R
  - 权限（permission）
    - » 对于客体的操作权力。权限集合为P
  - 用户分配（User Assignment）
    - » 将用户与角色关联
    - » 用户分配集合为UA={(u,r)|u∈U, r∈R}
    - » 用户 u与角色 r关联后，将拥有 r的权限。

基于角色的访问控制基本机制

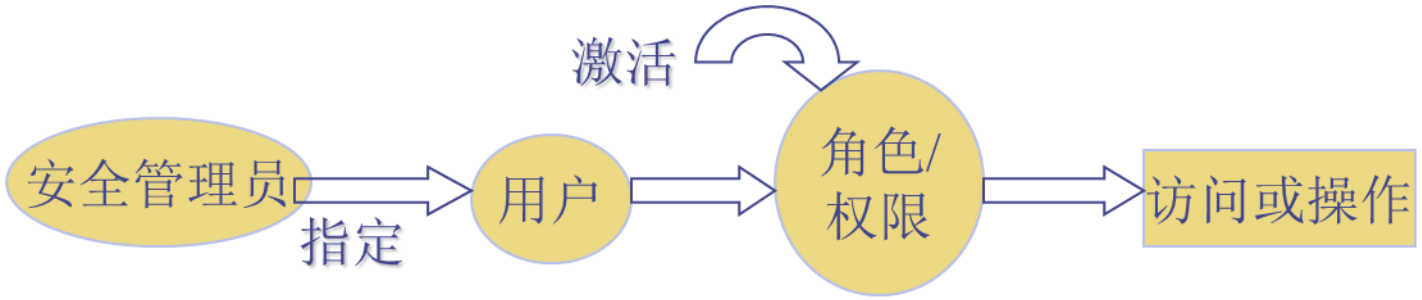
- RBAC的基本机制
  - RBAC的授权机制
    - 分为两步
      - 将用户分配给角色
      - 将访问权限分配给角色
    - 授权要满足的安全约束条件
      - 最小特权原则
      - 职责分离原则
      - 角色互斥原则
      - 角色激活限制原则
    - 角色分级，高级角色可以继承低级角色的访问权限

基于角色的访问控制基本机制

- RBAC的基本机制
  - RBAC的用户与角色的关系（多对多的关系）
    - 一个用户可担当多个角色
    - 一个角色可分配给多个用户
  - 角色和权限的之间的关系（多对多的关系）
    - 个角色可以拥有多个访问权限，
    - 不同的角色也可以拥有相同的权限
  - 角色和角色的关系（分级关系）
    - 高级角色可以继承低级角色的访问权限

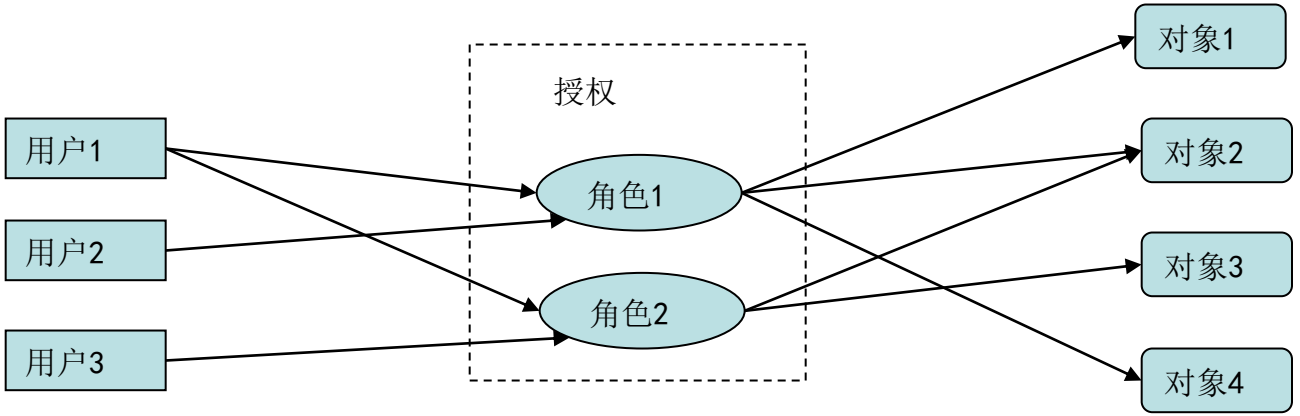
与传统访问控制的差别

- 一个用户必须扮演某种角色，而且还必须激活这一角色，才能对一个对象进行访问或执行某种操作。
- 增强一层间接性带来了灵活性



基于角色的访问控制特点

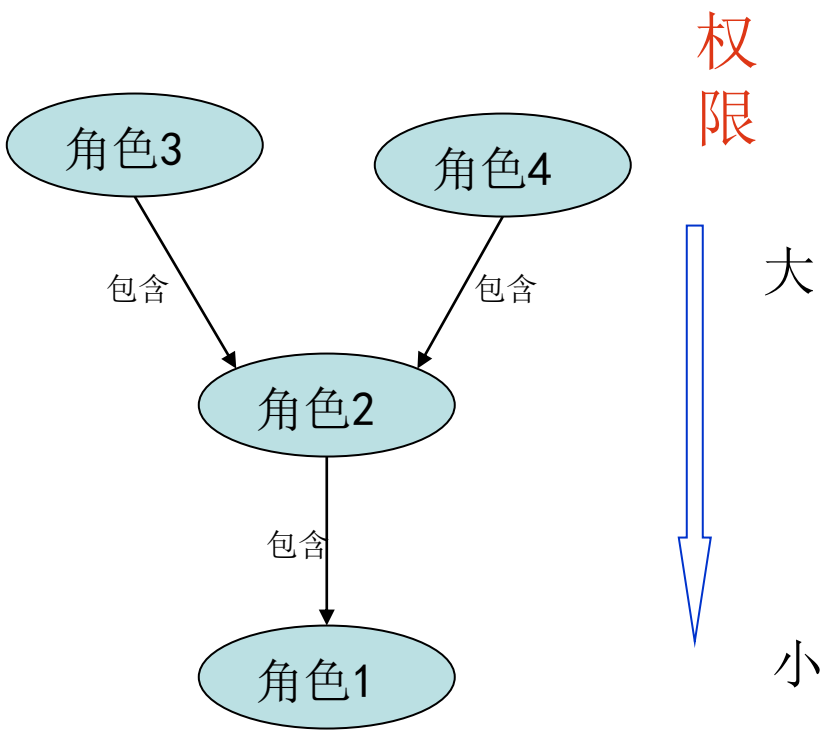
- (1)提供了三种授权管理的控制途径：
  - 改变客体的访问权限；
  - 改变角色的访问权限；
  - 改变主体所担任的角色。



角色、用户、权限、授权管理之间的关系

基于角色的访问控制特点

- (2)提供了层次化的管理结构，由于访问权限是客体的属性，所以角色的定义可以用面向对象的方法来表达，并可用类和继承等概念来表示角色之间的关系。



基于角色的访问控制特点

- (3)具有提供最小权限的能力，由于可以按照角色的具体要求来定义对客体的访问权限，因此具有针对性，不出现多余的访问权限，从而降低了不安全性。
- (4)具有责任分离的能力，不同角色的访问权限可相互制约，即定义角色的人不一定能担任这个角色。因此具有更高的安全性。

非任意访问控制（non-discretionary access control）是为满足安全策略和目标而采用的一系列集中管理的控制手段。访问控制是由访问者在机构中的角色决定的。角色包括职务特征、任务、责任、义务和资格。访问者在系统中的角色有管理者赋予或吊销。

RBAC的应用

- 在银行环境中，用户角色可以定义为出纳员、分行管理者、顾客、系统管理者和审计员
- 访问控制策略的一个例子如下：
  - （1）允许一个出纳员修改顾客的帐号记录（包括存款和取款、转帐等），并允许查询所有帐号的注册项
  - （2）允许一个分行管理者修改顾客的帐号记录（包括存款和取款，但不包括规定的资金数目的范围）并允许查询所有帐号的注册项，也允许创建和终止帐号
  - （3）允许一个顾客只询问他自己的帐号的注册项
  - （4）允许系统的管理者询问系统的注册项和开关系统，但不允许读或修改用户的帐号信息
  - （5）允许一个审计员读系统中的任何数据，但不允许修改任何事情

RBAC的优势

- 便于授权管理，如系统管理员需要修改系统设置等内容时，必须有几个不同角色的用户到场方能操作，从而保证了安全性。
- 便于根据工作需要分级，如企业财务部门与非财力部门的员工对企业财务的访问权就可由财务人员这个角色来区分。
- 便于赋于最小特权，如即使用户被赋于高级身份时也未必一定要使用，以便减少损失。只有必要时方能拥有特权。
- 便于任务分担，不同的角色完成不同的任务。
- 便于文件分级管理，文件本身也可分为不同的角色，如信件、账单等，由不同角色的用户拥有。

RBAC的缺点

- 静态授权，任务完成后权限没有收回，没有考虑上下文
- 基于主体、客体、被动访问控制，从系统角度出发，不能主动防御

三种访问控制——总结

- 自主访问控制机制虽然有着很大的灵活性，但同时也存在着安全隐患；
- 强制访问控制机制虽然大大提高了安全性，但是在灵活性上就会大打折扣。这两种传统的访问控制机制存在的问题使得访问控制的研究向着新的方向发展，基于角色的访问控制机制就是在这种形势下的产物，它克服了传统的访问能控制机制的不足，是一种有效而灵活的安全策略，它是未来访问控制的发展趋势

其他访问控制技术

- 基于任务的访问控制——93～97年R.K.Thomas等人提出的TBAC
  - 是一种基于任务采用动态授权的主动安全模型
  - 基本思想：
    - 将访问权限与任务相结合,每个任务的执行都被看作是主体使用相关访问权限访问客体的过程。在任务执行过程中，权限被消耗，当权限用完时，主体就不能再访问客体了。
    - 系统授予给用户的访问权限，不仅仅与主体、客体有关，还与主体当前执行的任务、任务的状态有关。客体的访问控制权限并不是静止不变的, 而是随着执行任务的上下文环境的变化而变化
  - 缺点：没有将角色与任务清楚地分离开来, 也不支持角色的层次等级；TBAC并不支持被动访问控制, 需要与RBAC结合使用。

其他访问控制技术

- 基于角色-任务的访问控制——98年G.Coulouria等人提出的T-RBAC
  - 先将访问权限分配给任务,再将任务分配给角色,角色通过任务与权限关联,任务是角色和权限交换信息的桥梁。
  - 在T-RBAC模型中, 任务具有权限,角色只有在执行任务时才具有权限, 当角色不执行任务时不具有权限;
  - 权限的分配和回收是动态进行的,任务根据流程动态到达角色,权限随之赋予角色,当任务完成时,角色的权限也随之收回;
  - 角色在工作流中不需要赋予权限。
  - 这样, 不仅使角色的操作、维护和任务的管理变得简单方便,也使得系统变得更为安全。

其他访问控制技术

- 基于属性的访问控制-ABAC
  - 为解决行业分布式应用可信关系访问控制模型
  - 利用相关实体(如主体、客体、环境)的属性作为授权的基础来研究如何进行访问控制
  - 基于这样的目的,可将实体的属性分为主体属性、实体属性和环境属性
  - 在基于属性的访问控制中，访问判定是基于请求者和资源具有的属性，请求者和资源在ABAC 中通过特性来标识
  - ABAC具有足够的灵活性和可扩展性，同时使得安全的匿名访问成为可能，这在大型分布式环境下是十分重要的

其他访问控制技术

- 基于规则策略的访问控制E.Bertino等人提出的
- 面向服务的工作流访问控制SOWAC
- 基于状态的访问控制
- 基于对象的访问控制OBAC
- 域和类型执行的访问控制DTE

目录

安全审计的必要性

- 一旦我们采用的防御体系被突破怎么办？至少我们必须知道系统是怎样遭到攻击的，这样才能恢复系统
- 此外我们还要知道系统存在什么漏洞，如何能使系统在受到攻击时有所察觉，如何获取攻击者留下的证据
- 安全审计的概念就是在这样的需求下被提出的，它相当于飞机上使用的“黑匣子”

安全审计的必要性

- 在TCSEC和CC等安全认证体系中，安全审计功能都是放在首要位置的，它是评判一个系统是否真正安全的重要尺码
- 安全审计系统能够帮我们对操作系统安全进行实时监控，及时发现整个系统的动态，发现网络入侵和违规行为，忠实记录系统上发生的一切，提供取证手段

安全审计的相关标准

- 安全审计的相关标准

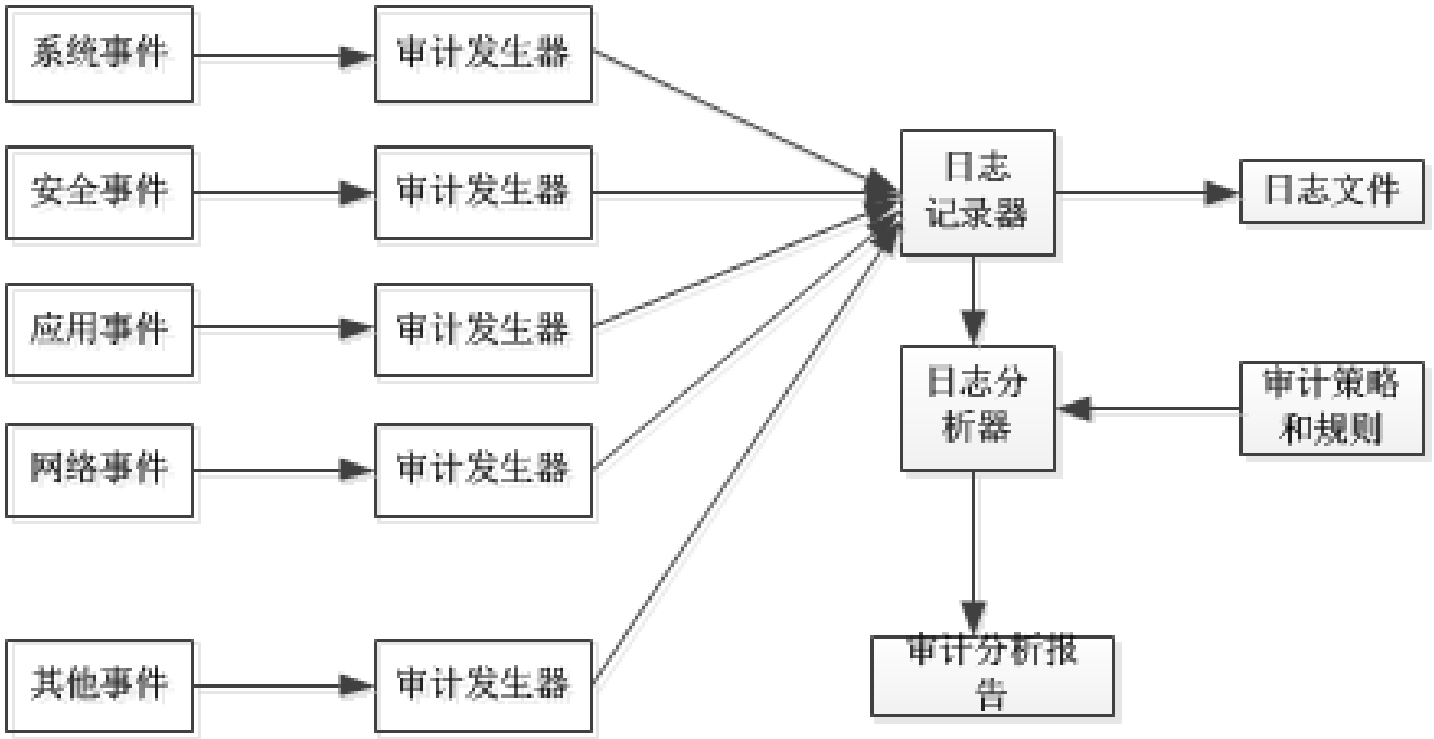
安全审计的相关概念

- 相关概念
  - 安全审计就是对系统中有关安全的活动进行记录、检查及审核。
  - 主要目的：检测和阻止非法用户对计算机系统的入侵，并显示合法用户的误操作。
  - 审计机制是通过对日志的分析来完成的。
  - 日志就是记录的事件或统计数据，都能提供关于系统使用及性能方面的信息。
  - 主要作用：
    - 发现不安全因素，及时报警
    - 对违反安全规则的行为或企图提供证据
    - 对已受攻击的系统，可以提供信息帮助进行损失评估和系统恢复

安全审计机制类型

- 系统级审计
  - 登录情况、登录识别号、每次登录尝试的日期和具体时间、每次退出的日期和时间、所使用的设备、登录后运行的内容
- 应用级审计
  - 打开和关闭数据文件、读取、编辑和删除记录或字段的特定操作、打印报告等用户活动
- 用户级审计
  - 用户直接启动的所有命令、用户所有鉴别和认证尝试、用户所访问的文件和资源等方面

审计系统的组成


系统事件
安全事件
应用事件
网络事件
其他事件
审计发生器
审计发生器
审计发生器
审计发生器
审计发生器
日志
记录器
日志文件
日志分
析器
审计策略
和规则
审计分析报
告

日志记录的原则

- 在理想情况下，日志应该记录每个可能的事件，以便分析发生的所有事件，并恢复任何时刻进行的历史情况。但这样存储量过大，并且将严重影响系统的性能。因此。日志的内容应该是有选择的。一般情况下日志的记录应该满足如下的原则：
  - （1） 日志应该记录任何必要的事件，以检测已知的攻击模式
  - （2） 日志应该记录任何必要的事件，以检测异常的攻击模式。
  - （3） 日志应该记录关于记录系统连续可靠工作的信息。

日志的内容

- 日志系统根据安全要求记录下面事件的部分或全部。通常，对于一个事件，日志应包括事件发生的日期和时间、引发事件的用户（地址）事件、和源目的的位置、事件类型、事件成败等：
  - 审计功能的启动和关闭
  - 使用身份鉴别机制
  - 将客体引入主体的地址空间
  - 删除客体
  - 管理员、安全员、审计员和一般操作人员的操作
  - 其他专门定义的可审计事件

安全审计分析

- 日志分析就是在日志中寻找模式，其主要内容：
  - （1）潜在侵害分析：日志分析应能用一些规则去监控审计事件，并根据规则发现潜在的入侵。
  - （2）基于异常检测的轮廓：确定正常行为轮廓，当日志中的事件违反它或超出他的一定门限，能指出将要发生的威胁。
  - （3）简单攻击探测：对重大威胁特征有明确描述，当攻击现象出现，能及时指出。
  - （4）复杂攻击检测：要求高的日志分析系统还应能检测到多部入侵序列，当攻击序列出现，能预测其发生的步骤。

审计事件查阅

- 由于审计系统是追踪、恢复的直接依据，甚至是司法依据，因此其自身的安全性十分重要。审计系统的安全性主要是查阅和存储的安全。
- 审计事件的查阅应该受到严格的限制，不能篡改日志。通常通过以下不同的层次来保证查阅的安全。
  - （1）审计查阅：审计系统以可理解的方式为授权用户提供查阅日志和分析结果的功能。
  - （2）有限审计查阅：审计系统只能提供对内容的读权限，因此应拒绝具有读以外权限的用户访问审计系统。
  - （3）可选审计查阅：在有限审计查阅的基础上限制查阅的范围。

审计事件存储

- 审计事件的存储也有安全性的要求，具体有如下几种情况：
  - （1）受保护的审计踪迹存储：即要求存储系统对日志事件具有防护功能，防止未授权的修改和删除，并具有检测修改和删除的能力。
  - （2）审计数据的可用性保证：在审计存储系统遭受意外时，能防止或检测审计记录的修改，在存储介质存满或存储失败时，能确保记录不被破坏。
  - （3）防止审计数据丢失：在审计踪迹超过预定的门限或记满时，应采取相应的措施防止数据丢失。这种措施可以是忽略可审计事件、只允许记录有特殊权限的事件、覆盖以前记录、停止工作等等。

目录

可信路径

- 具体实施安全策略的软硬件构成安全内核，而用户是与安全周界外部的不可信的中间应用层及操作系统交互的，但用户登录、定义用户的安全属性、改变文件的安全级别等安全关键性操作，用户必须能够确认与安全内核进行交互，而不是与一个特洛伊木马程序打交道。这就需要提供一种安全机制，保障用户和安全内核之间的通信，而这种机制就是由可信路径提供的。
- 可信路径（Trusted path）机制即终端人员能借以直接与可信计算基TCB通信的一种机制。该机制只能由有关终端操作人员或可信计算基启动，并且不能被不可信软件模拟。保障用户和内核的可信通信。

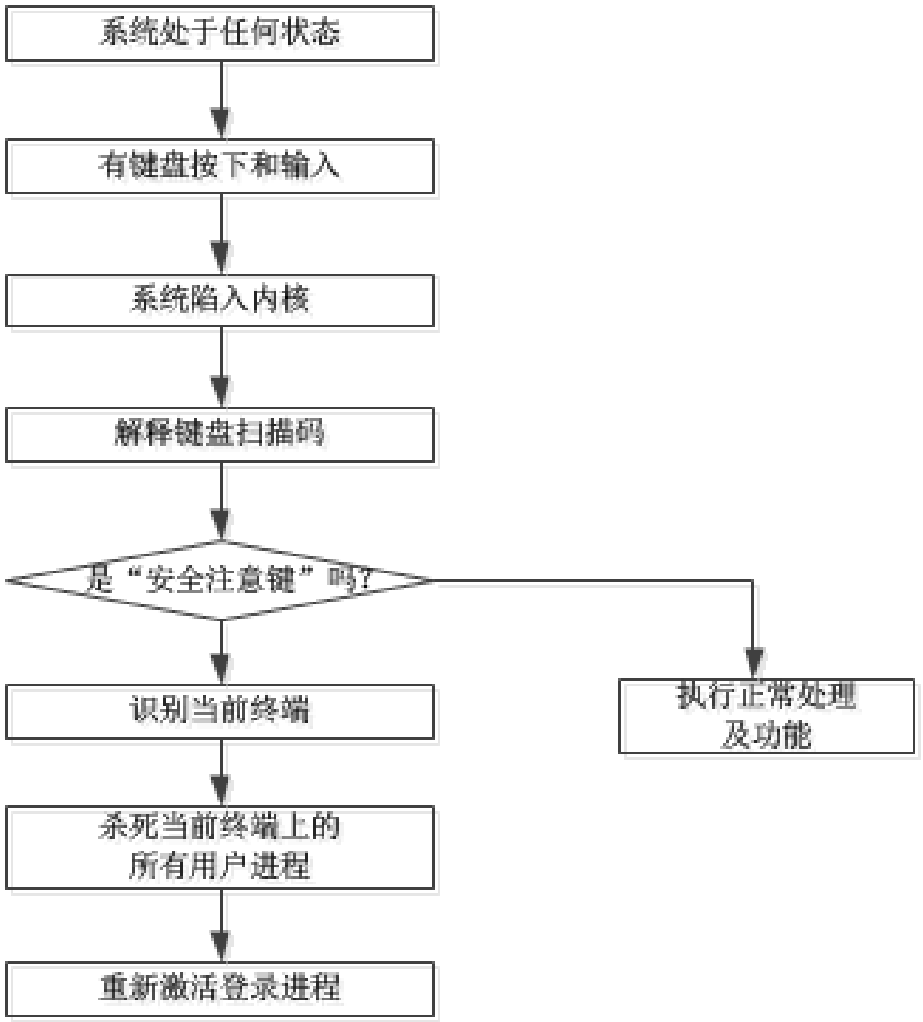
构建可信路径的方法

- 构建可信路径的两种方法：
  - 构建可信路径的简单方法是为每个用户提供两台终端，一台用于完成日常的普通工作，另一台用于实现与安全内核的硬链接及专职执行安全敏感操作。显然，此法具有代价昂贵的致命缺陷，同时还会引入诸如如何确保“安全终端”的安全可靠及如何实现“安全终端”和“普通终端”的协调工作等新问题。
  - 更为现实的方法是要求用户在执行敏感操作前，使用一般的通用终端和向安全内核发送所谓的“安全注意符”（即不可信软件无法拦截、覆盖或伪造的特定信号）来触发和构建用户与安全内核间的可信通路。

Linux的安全注意键

- 可信路径：
  - 为了使用户确信自己的用户名和口令不被别人窃走，linux提供了“安全注意键**SAK：Security Attention Key**）”
    - 它是一个键或一组键（Linux操作系统提供的安全提示键在X86平台下为ALT+SysRq+k。Windows操作系统则为CTRL+ALT+DEL)
    - 按下它们后，保证用户看到真正的登录提示，而非登录模拟器（保证是真正的登录程序读取用户的账号和口令）

基于安全注意键的可信路径



可信路径实现技术关键环节

- 截取键盘输入，即只要有键盘事件发生，就陷入系统内核进行解释，并根据其扫描码判断是否为“安全注意键”
- 根据键入“安全注意键”的键盘事件来源，从当前终端列表中找到当前活跃的终端
- 根据进程标识符及进程家族关系，杀死当前活跃终端上的全部用户进程。这主要是基于“作为可信计算基组成部分的系统内核（具体体现为系统进程）肯定可靠，而用户进程则不一定可靠”的假设与考虑

谢谢！

目录

Linux安全机制

- 标识
  - /etc/passwd和/etc/shadow中保存了用户标识和口令
- 鉴别
  - PAM（Pluggable authentication modules）模块
- 访问控制
  - 存取权限：可读、可写、可执行
  - 改变权限：chmod
  - 特殊权限位：SUID和SGID
  - 控制策略：smack、selinux等

Linux安全机制

- 审计
  - 日志文件：dmesg、auth.log、fail.log、audit.log、syslog等
  - 审计服务程序：syslogd
- 网络安全性
  - 有选择地允许用户和主机与其他主机的连接
- 网络监控和入侵检测:LIDS等
- 备份/恢复:实时备份、整体备份、增量备份

用户标识的配置文件

- /etc/passwd
  - 记录了Linux系统中每个用户的一些基本属性，并且对所有用户可读
  - /etc/passwd中每一行记录对应一个用户账号，有几行就代表有几个账号在你的系统中

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
……
squid:x:23:23::/var/spool/squid:/sbin/nologin
SH:x:500:500:songhong:/home/SH:bin/bash
```

身份标识

账户名　口令（密码）　组标识　账户注释　默认工作目录　登录后启动的程序

用户权限管理的配置文件

- /etc/shadow
  - 用户影子文件
  - 该文件只有root用户拥有读权限

```
root:$1$jYTJgmNb$bJ5LQwc.91D4MMangK.Sm.:14028:0:99999:7:::
bin:*:14028:0:99999:7:::
daemon:*:14028:0:99999:7:::
rpc:!!:14028:0:99999:7:::
sshd:!!:14028:0:99999:7:::
wenchang:$1$k7TyrJaO$DS/P61XHIz1xWAqLcdnQz1:14091:0:99999:7:::
......
```

账户名　密码　最后一次修改时间　最小时间间隔　最大时间间隔　不活动时间　警告时间　失效时间　保留

活动

用户权限管理的配置文件

- /etc/group
  - 用户组配置文件

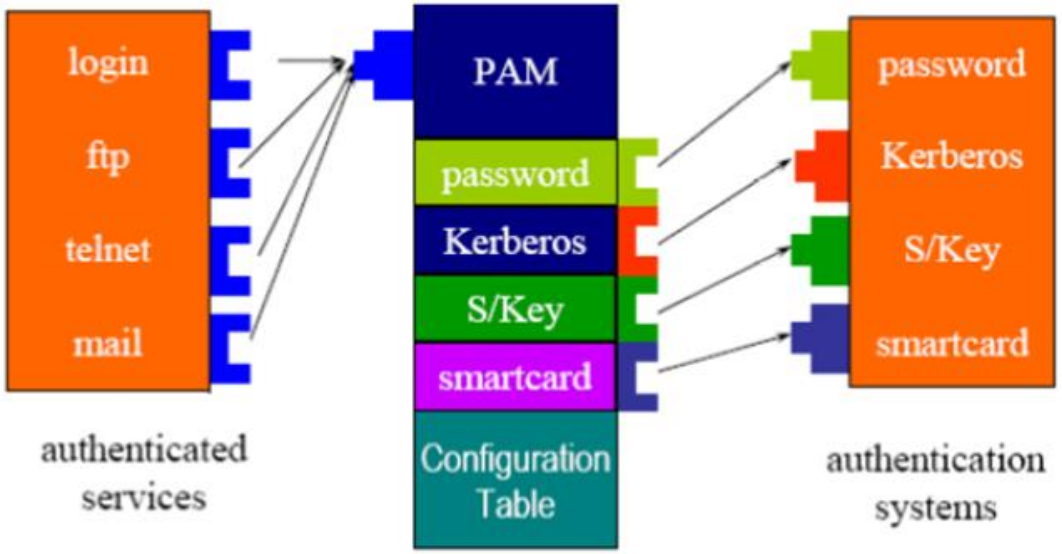
```
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
sisefellow:x:300:wenchang,binliang,zhiyong,zhaohui
sislab_ms:x:301:weinan,hanchao,kankan,xiaoli,liuxing
sislab_phd:x:302:xin08,hongwei,yanjun
.....
```

Linux系统的身份鉴别

- 基本认证过程（单机版）：
  - /sbin/init
  - /etc/rc.d/rc.sysinit
  - /etc/rc.d/rc0.d~rc6.d/*
  - /etc/rc.d/rc.local
  - /sbin/mingetty

Linux系统的身份鉴别

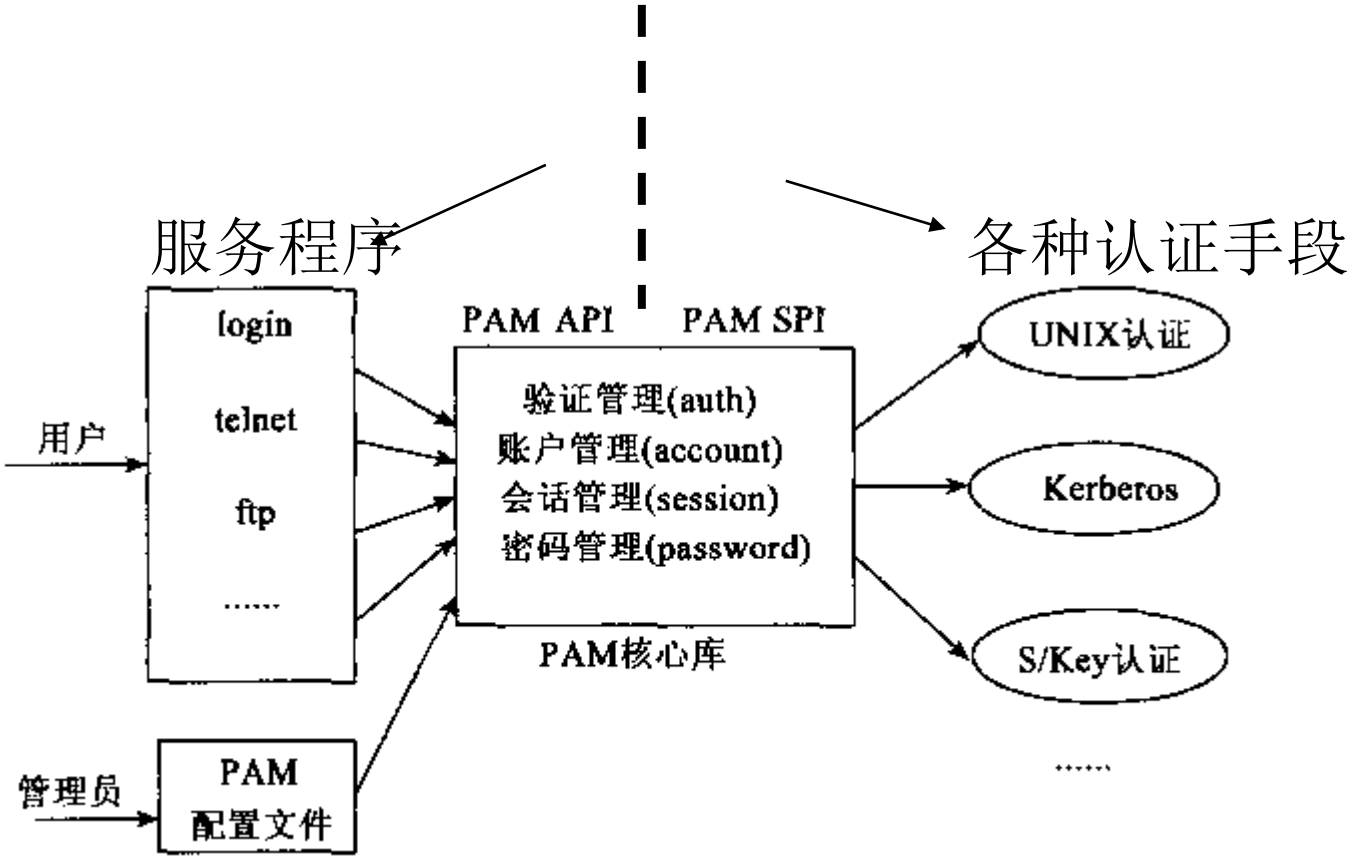
- 网络上的认证：
  - 密钥机制
  - 网上身份认证系统NIS——美国sun公司开发的
  - 面向服务的网上认证系统Kerberos——美国麻省理工学院开发的
  - 基于PAM的统一认证框架——美国sun公司为Solaris操作系统开发



PAM的组成

- PAM（pluggable Authentication Modules）的组成：
  - PAM API（应用接口层）
  - 动态装载库：身份认证、账户管理、口令管理、会话管理（鉴别模块层）——/lib/security/*
  - PAM配置文件（应用接口层）——/etc/pam.d/*

PAM的组成


服务程序
各种认证手段
login
telnet
ftp
.....
用户
PAM API
PAM SPI
验证管理(auth)
账户管理(account)
会话管理(session)
密码管理(password)
PAM核心库
UNIX认证
Kerberos
S/Key认证
.....
管理员
PAM
配置文件

PAM的组成

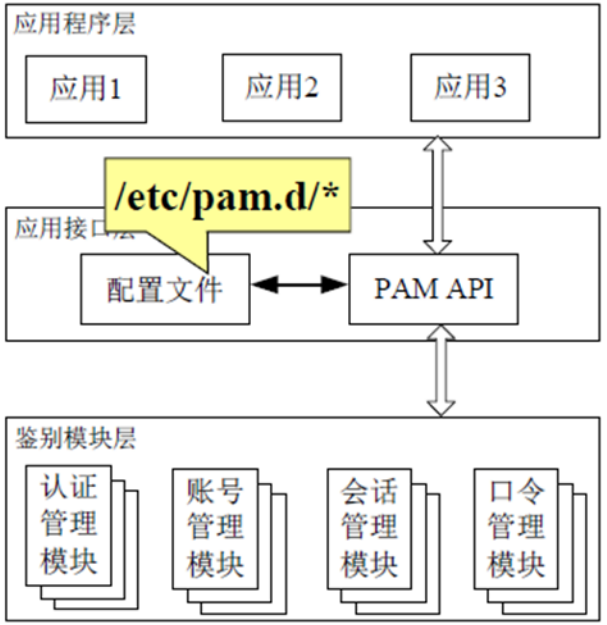
/etc/pam.d/*

```
[root@localhost pam.d]# ls
authconfig          passwd                      setup
chfn                poweroff                    smtp
chsh                ppp                         sshd
halt                reboot                      su
internet-druid      redhat-config-mouse         sudo
kbdrate             redhat-config-network       system-auth
login               redhat-config-network-cmd   up2date
neat                redhat-config-network-druid up2date-config
other               rhn_register                up2date-nox
```

vi /etc/pam.d/login

```
#%PAM-1.0
auth       required     pam_securetty.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
session    optional     pam_console.so
```

- required
- requisite
- sufficient
- optional



95

PAM的组成

/etc/pam.d/*

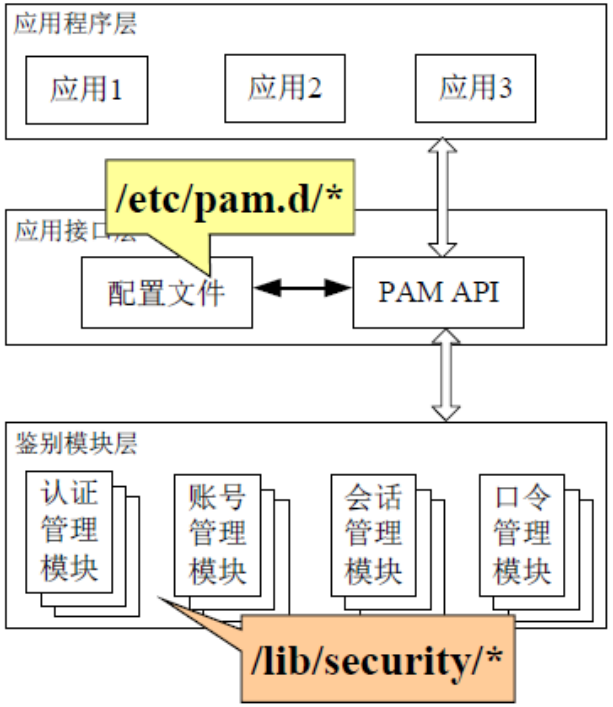
```
[root@localhost pam.d]# ls
authconfig          passwd                          setup
chfn                poweroff                        smtp
chsh                ppp                             sshd
halt                reboot                          su
internet-druid      redhat-config-mouse             sudo
kbdrate             redhat-config-network           system-auth
login               redhat-config-network-cmd       up2date
neat                redhat-config-network-druid     up2date-config
other               rhn_register                    up2date-nox
```

vi /etc/pam.d/login

```
#%PAM-1.0
auth       required     pam_securetty.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
session    optional     pam_console.so
```

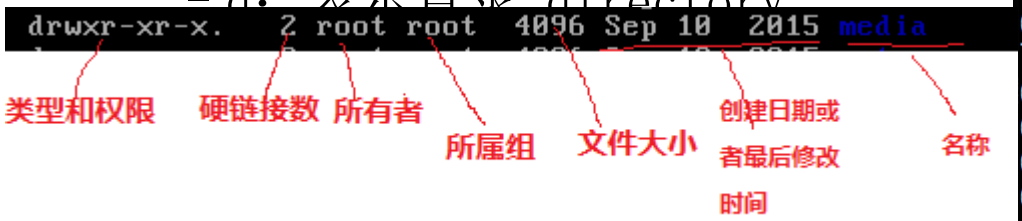
/lib/security

```
[root@localhost security]# ls
pam_access.so     pam_krb5afs.so    pam_permit.so       pam_timestamp.so
pam_chroot.so     pam_krb5.so       pam_pwdb.so         pam_unix_acct.so
pam_console.so    pam_lastlog.so    pam_rhosts_auth.so  pam_unix_auth.so
pam_cracklib.so   pam_ldap.so       pam_rootok.so       pam_unix_passwd.so
pam_deny.so       pam_limits.so     pam_securetty.so    pam_unix_session.so
pam_env.so        pam_listfile.so   pam_shells.so       pam_unix.so
pam_filter        pam_localuser.so  pam_smb_auth.so     pam_userdb.so
pam_filter.so     pam_mail.so       pam_stack.so        pam_warn.so
pam_ftp.so        pam_mkhomedir.so  pam_stress.so       pam_wheel.so
pam_group.so      pam_motd.so       pam_tally.so        pam_xauth.so
pam_issue.so      pam_nologin.so    pam_time.so
```



用户权限管理

- Linux访问控制实例-用户权限管理
  - bin文件夹的权限
    - drwxr-xr-x（d代表的文件类型）
      - d：表示目录 directory



    - l w x
      - 文件所有者拥有的权限是读写执行
    - r-x
      - 文件所属组对文件的权限是可读不可写可执行
    - r-x
      - 其他人的权限可读不可写可执行

```
root@apt:/home# ls -l /
total 88
drwxr-xr-x   2 root root  4096 Dec  7 07:38 bin
drwxr-xr-x   4 root root  4096 Dec  7 05:51 boot
drwxr-xr-x   2 root root  4096 Nov  2 05:45 datapool
drwxr-xr-x  16 root root  4200 Nov 16 06:10 dev
drwxr-xr-x 108 root root  4096 Dec 11 21:49 etc
drwxr-xr-x   3 root root  4096 Dec 11 21:48 home
lrwxrwxrwx   1 root root    33 Oct 18 08:27 initrd.img -> boot/initrd.img-3.19.0-25-generic
drwxr-xr-x  24 root root  4096 Oct 18 22:38 lib
drwxr-xr-x   2 root root  4096 Oct 18 22:38 lib64
drwx------   2 root root 16384 Oct 18 08:25 lost+found
drwxr-xr-x   3 root root  4096 Oct 18 08:26 media
drwxr-xr-x   2 root root  4096 Apr 10  2014 mnt
drwxr-xr-x   7 root root  4096 Oct 25 21:47 opt
dr-xr-xr-x 646 root root     0 Nov 16 06:10 proc
drwx------  20 root root  4096 Dec 11 19:42 root
drwxr-xr-x  22 root root   800 Dec 11 02:17 run
drwxr-xr-x   2 root root 12288 Oct 18 21:56 sbin
drwxr-xr-x   2 root root  4096 Aug  5  2015 srv
dr-xr-xr-x  13 root root     0 Nov 16 06:10 sys
drwxrwxrwt   7 root root  4096 Dec 11 21:50 tmp
drwxr-xr-x  10 root root  4096 Oct 18 08:26 usr
drwxr-xr-x  13 root root  4096 Oct 18 08:55 var
lrwxrwxrwx   1 root root    30 Oct 18 08:27 vmlinuz -> boot/vmlinuz-3.19.0-25-generic
root@apt:/home#
```

用户权限管理命令

- chmod（change the permission mode of a file）
  - 所在路径：/bin/chmod
  - 执行权限：所有用户
  - 功能描述：改变文件或目录权限
  - 语法：chmod [mode] 文件或目录
    - r（100）用十进制4表示
    - w（010）用十进制2表示
    - X（001）用十进制1表示

用户权限管理命令

- chmod（change the permission mode of a file）

chmod u=rwx，g=rx，o=rx abc：同上u=用户权限，g=组权限，o=不同组其他用户权限

chmod u-x，g+w abc：给abc去除用户执行的权限，增加组写的权限

chmod a+r abc：给所有用户添加读的权限

用户权限管理命令

- Chown改变所有者权限命令
- Chgrp改变用户组的权限命令

chown xiaoming abc：改变abc的所有者为xiaoming

chgrp root abc：改变abc所属的组为root

chown root ./abc：改变abc这个目录的所有者是root

chown －R root ./abc：改变abc这个目录及其下面所有的文件和目录的所有者是root

Linux系统审计-dmesg

```
type=AVC msg=audit(1453278081.311:18139): avc:  denied  { fork } for  pid=549 comm="test_
fork" scontext=unconfined_u:unconfined_r:testexec_t:s0-s0:c0.c1023 tcontext=unconfined_u:
unconfined_r:testexec_t:s0-s0:c0.c1023 tclass=process
```

谢谢！