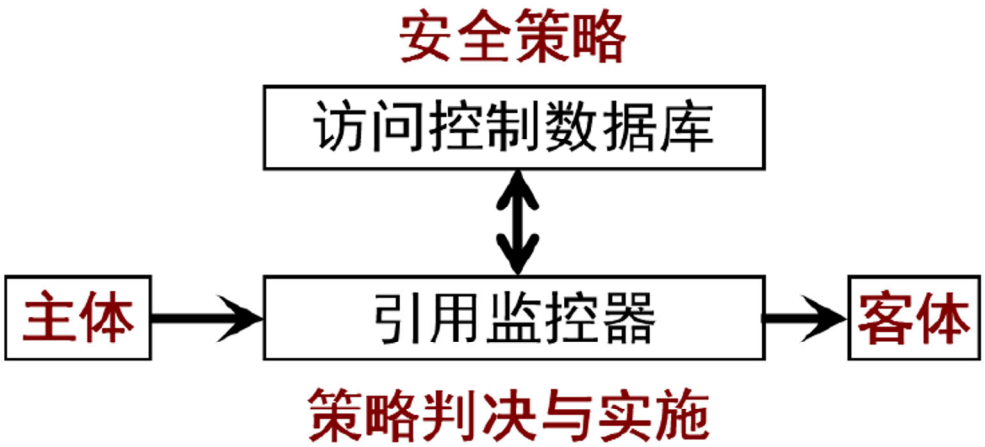
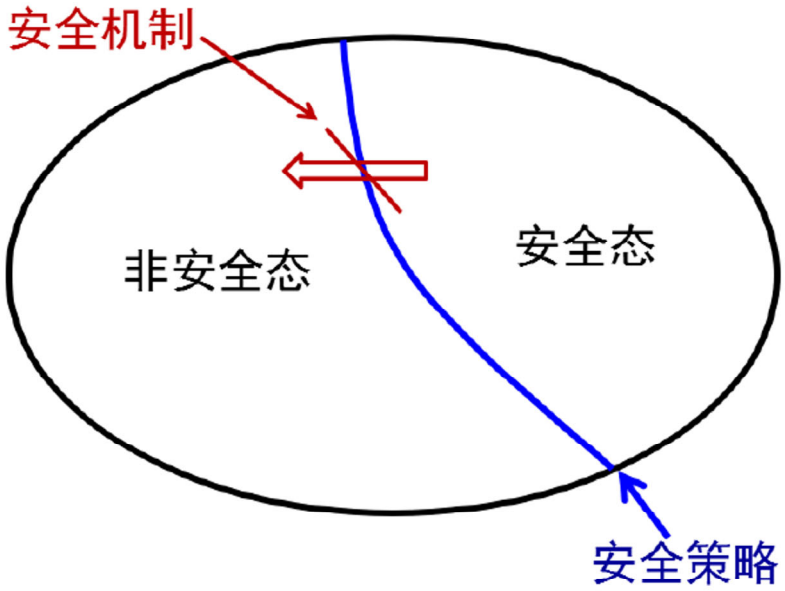
3 访问控制

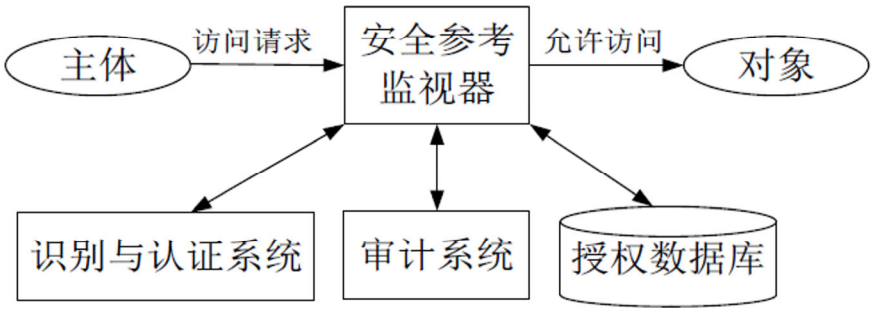
2019年4月10日　　23:35

1 概述

1.1 重要概念

- 客体（Object）
  - 定义：包含或接收信息的被动实体。
  - 例子：页面、文件、目录、套接字、进程、数据库、表、字段、...
- 主体（Subject）
  - 定义：可导致信息在客体间流动，或使系统状态发生变化的主动实体。
  - 例子：用户、进程、...
- 授权（Authorization）
  - 定义：规定主体可以对客体执行的动作。
  - 例子：读、写、执行、拒绝访问、...
- 引用监控器（ReferenceMonitor）
  - 不同译法：引用监督器、参照监视器、参考监视器等
  - 定义：对主体访问客体的行为进行仲裁的抽象装置。
  - 一个描述访问控制的抽象概念
  - 

- 安全策略（SecurityPolicy）
  - 由一整套严密的安全规则组成。
  - 决策的集合，集中体现了一个组织或信息系统对安全的态度。那些行为是可以接受的，那些行为是不能接受的，对于违规行为做出何种反应，需要明确界定。
  - 一种将系统状态划分为安全态（授权态）和非安全态（未授权态）的声明
    - 授权状态（安全）：系统可进入的状态
    - 未授权状态（不安全）：系统进入这些状态，则违背了安全性
  - 安全的系统
    - 初始于授权状态
    - 不会进入未授权状态
- 安全策略与安全机制
  - 

- 安全模型（SecurityMode）
  - 安全模型是对安全策略所表达的安全需求的一种简单、抽象、精确和无歧义的描述
  - 安全模型给出了安全系统的形式化定义，正确地综合系统的各类因素，包括系统的使用方式、环境类型、授权定义、共享资源和共享类型等。
  - 

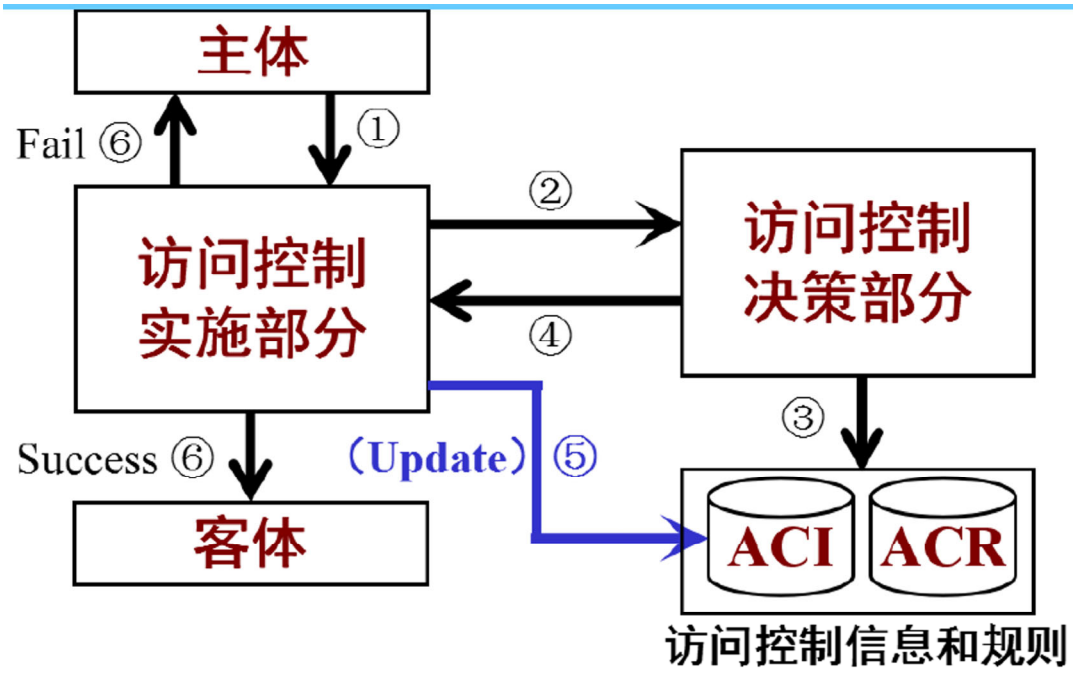
单机安全模型(Anderson)

1.2 访问控制策略

- 访问控制策略是对访问控制和相关授权的描述。
- 是保证信息系统安全最重要的核心策略之一。

- 目的：限制访问主体对访问客体的访问权限，从而使计算机系统在合法范围内使用。它决定用户能做什么，也决定代表一定用户利益的程序能做什么。

通用访问控制框架（GFAC）



2 自主访问控制

自主访问控制(DiscretionaryAccessControl,DAC)

- 自主访问控制策略下，每个客体有且仅有一个属主。
- 客体的属主（拥有者）可以按照自己的意愿精确地指定主体对客体的访问权限。
- 灵活性高，应用于UNIX，Windows等。

“自主”的含义

- 客体的属主可以自主地决定共享该客体的主体和方式。
- 具有授权能力的用户，可以自主地将访问权的部分或全部授予其他用户。

“自主”权的转交

- 严格的DAC(StrictDAC)：客体属主不允许其他用户代理客体的权限管理；
- 自由的DAC(LiberalDAC)：客体属主允许其他用户代理客体的权限管理，可以进行多次转交；
- 属主权可以转让的DAC：客体属主可以将作为属主的权利转交给其他用户。

自主访问控制的实现方式

- 访问口令
- 访问控制矩阵
- 访问能力表
- 访问控制表

- 授权关系表
- 属主/同组用户/其他用户

访问口令（PasswordsforAccess）

- 每个客体至少有一个访问口令，通过口令的分发，对主体进行访问授权；
- 主体在对客体进行访问前，必须向操作系统提供相应的访问口令；
- 注意：此口令不同于身份认证口令。

访问控制矩阵(AccessControlMatrix)

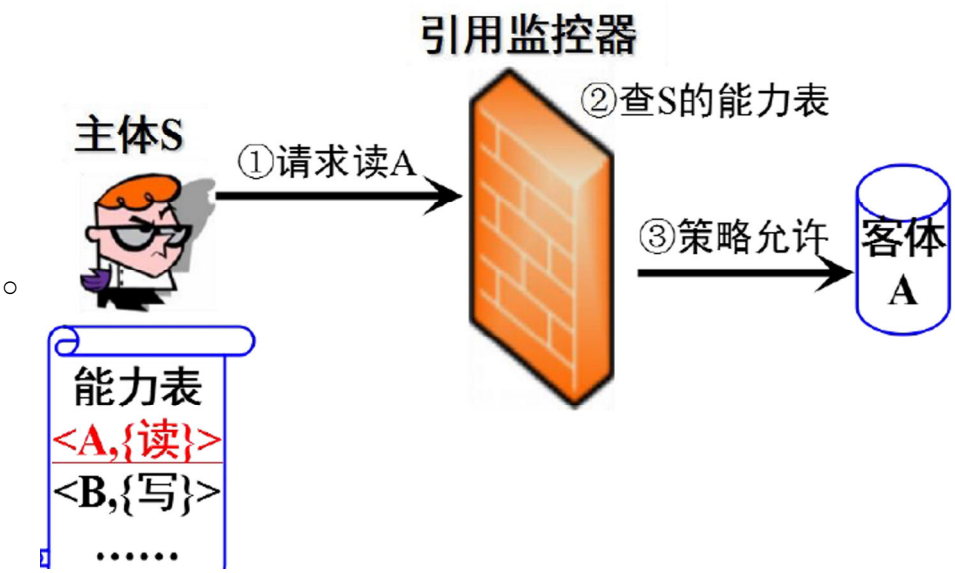
- 用矩阵完整地表示一个自主访问控制系统，规则如下：
  - 用主体的标识索引矩阵的行；
  - 用客体的标识索引矩阵的列；
  - 矩阵中的元素表示其所在行对应的主体，对该元素所在列对应的客体的访问权限。
- 访问控制矩阵实例
  -

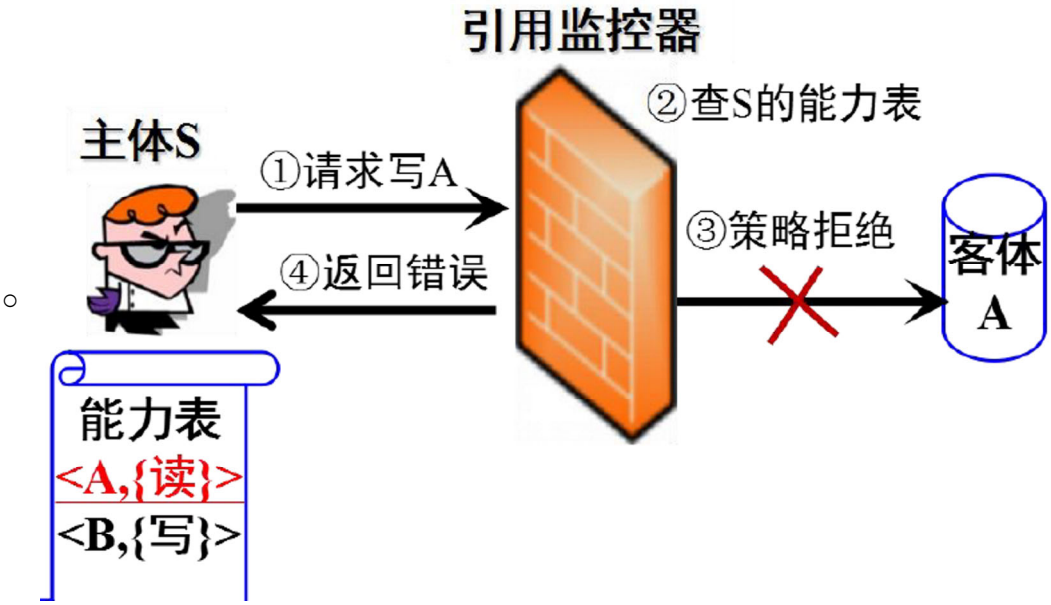
	客体1	客体 2	客体3	客体4
主体1	read, write, own	read	write	read, write
主体2	read, write	read, own	read	write

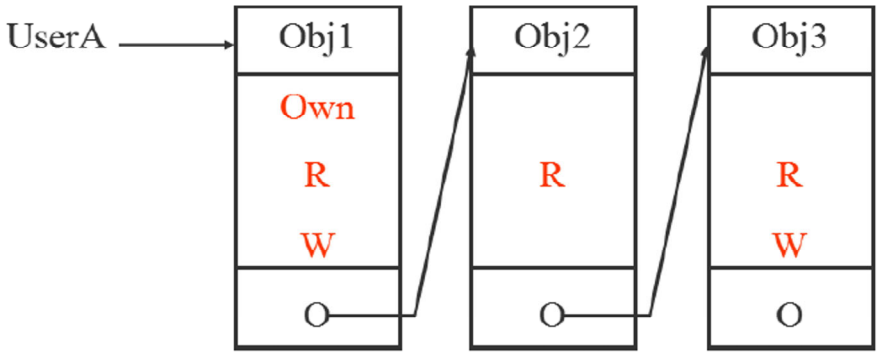
- 任何访问控制策略最终均可被模型化为访问矩阵形式。
- 最原始的访问控制方法。
- 开销大、不易扩展、管理。

访问能力表（CapabilityList）

- 主体可访问的客体的明细表，表中的每一项包含：客体的标识，主体对该客体的访问权限。
  - 能力：<客体标识，{访问方式}>
  - 能力决定用户能以何种方式访问客体
  - 拥有相应能力的主体可以按照指定的访问方式访问客体。
- 访问能力表的工作原理

- 

- 基于票据的访问控制
- 

- 访问能力表的实现
- 

-

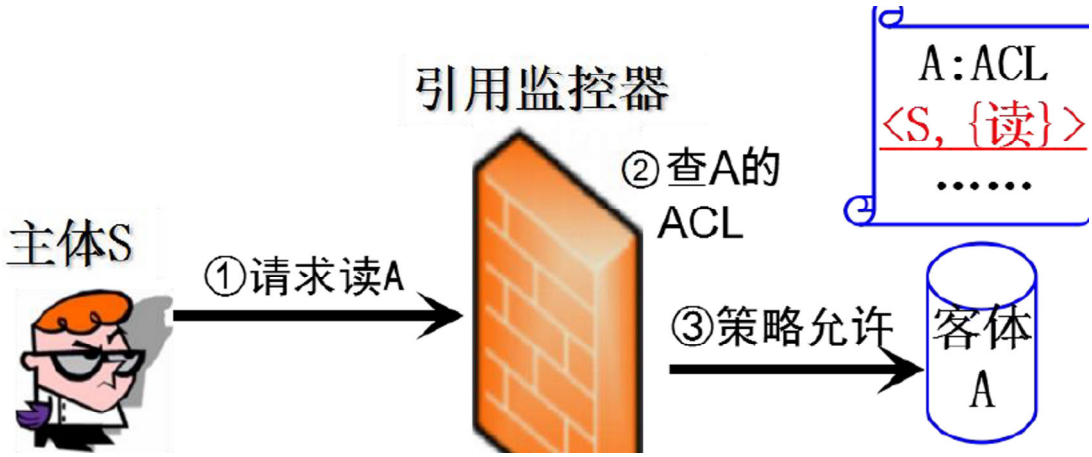
	客体1	客体2	客体3	客体4
主体1	read, write, own	read	write	read, write
主体2	read, write	read, own	read	write

访问控制表（AccessControlList）

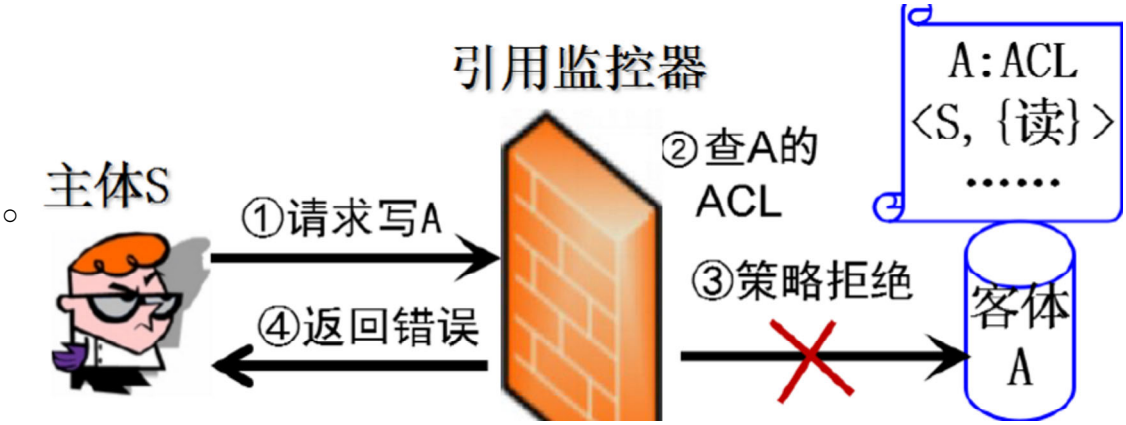
- 在每个客体上附加的主体明细表，表中每一项包括主体的身份和主体对该客体的访问权限。

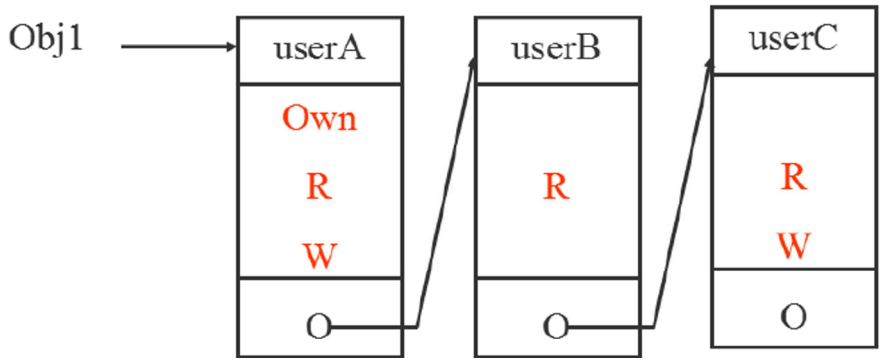
File1:	
	S_1, {R, W, X}
	S_2, {R}
	S_3, {R, W}
	. . .
	S_n, {W}

- 访问控制表的工作原理



- 基于列表的访问控制（ListBasedAccessControl）

- 

- 访问控制表的实现
  - 

- 应用最广泛的访问控制方法
- 是以客体为基准的
- 灵活、易用、直观
- Unix、Windows都使用了该方法进行访问控制

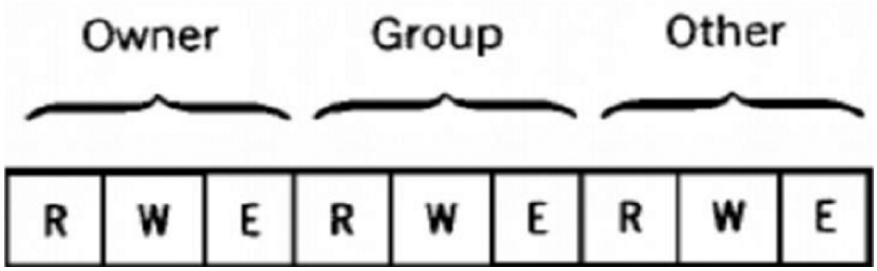
授权关系表(AuthorizationRelations)

主体1	R	客体1
主体1	W	客体1
主体1	Own	客体1
主体1	R	客体2
主体1	W	客体3

- 对应于访问控制矩阵中每一个非空元素的实现方法。既不对应于行，也不对应于列
- 授权关系表的特点
  - 授权关系表中的每一行表示了主体和客体的一个权限关系；
  - 如果这张表按主体进行排序，可以拥有访问能力表的优势；
  - 如果这张表按客体进行排序，可以拥有访问控制表的优势；
  - 特别适合关系数据库

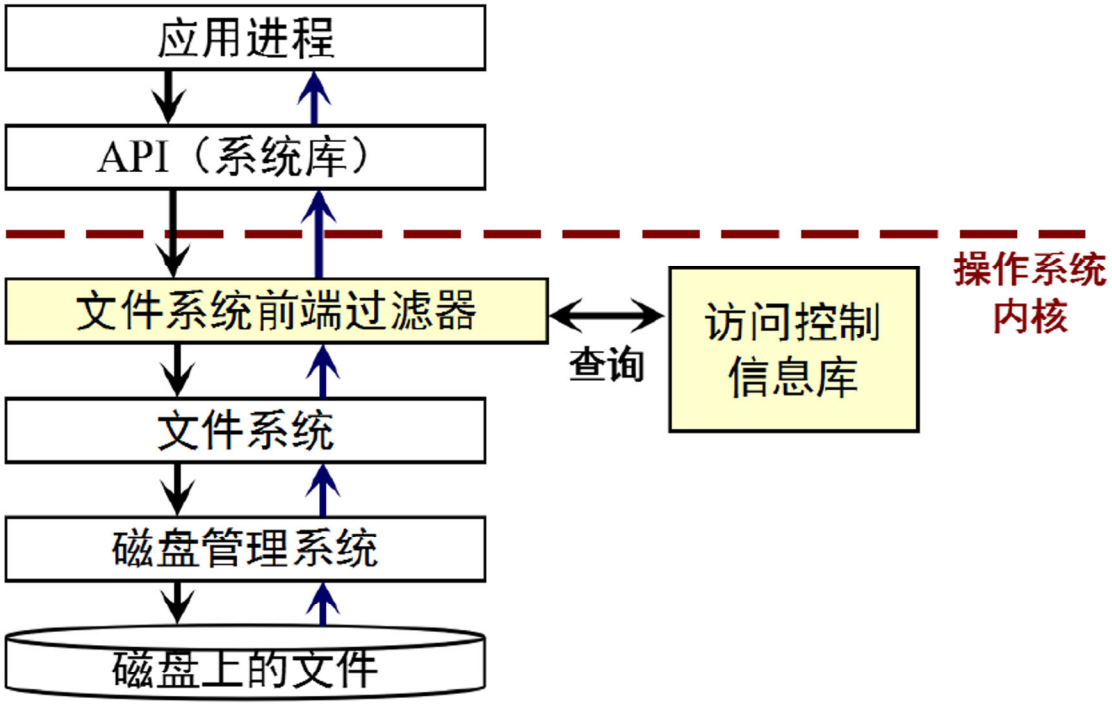
属主/同组用户/其他用户(Owner/Group/Other)

- 在每个文件属性中，附加一段有关访问控制信息的二进制位。这些二进制位反映了文件的主人、与文件主人同组的用户、剩下的其他用户等三类不同用户的访问权限。

- 应用于UNIX/Linux系统中
- 

  - Owner：客体属主的访问权限
  - Group：与Owner同组的用户对该客体的访问权限
  - Other：余下的其他用户对该客体的访问权限
- 缺点
  - 与ACL相比，基于Owner/Group/Other的访问控制粒度较粗
    - 客体的属主不能精确控制某个用户对其客体的访问权限
    - 系统只能将访问权限分配到同组用户或者所有其他用户

访问控制在操作系统中的实现

- 

自主访问控制的局限性

- 自主访问控制策略下，客体属主决定该客体的保护策略，即资源的拥有者对资源的访问策略具有决定权。让属主参与授权管理，存在安全缺陷。
- 允许在主体间传递访问权限，在传递过程中访问权限关系可能被改变，带来安全隐患。

3 强制访问控制

- 3.1 BellLaPadula(BLP)模型
- 3.2 Biba模型

强制访问控制(MandatoryAccessControl,MAC)

- 根据严格的安全规则、以行政方式生成并赋予主、客体一定的安全属性。
- 系统根据主、客体的安全属性决定是否授予主体访问权限。
- 客体的拥有者无权传播该客体的访问权限。
- 主体不能改变自己、自己所拥有客体、其他主体、以及其他所有客体的安全属性。

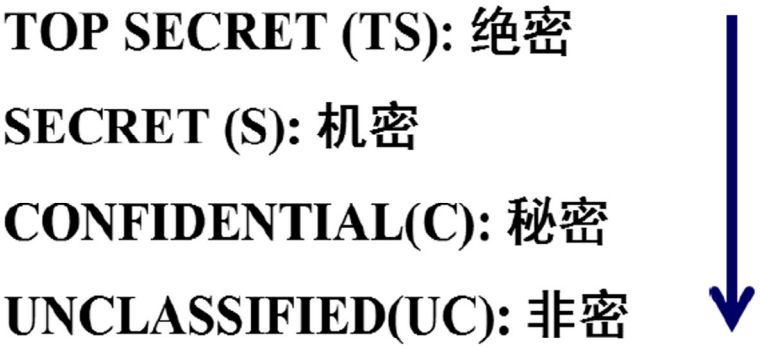
强制访问控制的特点

- 系统对授权集中管理。
- 管理部门按照严格的安全规则设置系统中的主体、客体的安全级别、安全范畴等安全属性。
- 系统运行时通过比较主、客体的安全属性，决定是否允许主体以其所请求的方式访问客体。

多级安全(MultilevelSecurity,MLS)

- 军事安全策略的数学描述
- 信息的安全级别（Classification）
- 人员的安全级别（Clearance）
- 在实施访问控制时，系统先对主体和客体的安全级别属性进行比较，当且仅当主体的安全级别高于或等于客体的安全级别时，主体才能访问客体。

安全级别（初始版）

- 

军事安全策略

- 保密规定
  - 保密规定1:如果某人的安全级别达不到信息的安全级别，那么就禁止把该信息传播给他
  - 保密规定2:如果某人的安全级别达到了信息的安全级别，则允许把该信息传播给他。
- 需知原则（NeedtoKnow）:用户仅能访问他履行职责所需的资源。
- 安全级别（正式版）：<密级，｛范畴｝>
  - 密级：绝密>机密>秘密>非密
  - {范畴}:
    - 对主体：用以表示可知悉的信息范畴
    - 对客体：用以限定信息的传播范围

    - 例子：
        - 文件A：<绝密，｛人事部｝>
        - 用户B：<机密，｛人事部,财务部｝>

支配（Dominate,dom）——偏序关系

- 定义:安全级别l <L，C>支配安全级别<L'，C'> ,当且仅当 L'<=L并且C'属于C。

(不考)3.1 BellLaPadula(BLP)模型

- 安全目标:机密性(Confidentiality)。
- 第一个经严格数学证明的安全模型。
- 最常用、最著名的多级安全模型，实际应用于多种安全操作系统、安全数据库的开发中

BLP模型中主体对客体有四种访问权限

- 只读（ReadOnly）
- 添加（Append）
- 执行（Execute）
- 读写（ReadWrite）

(不考)BLP模型的安全访问规则

- SS策略（SimpleSecurityProperty）：
    - 主体S可以对客体O进行读访问，仅当S的安全级支配客体O的安全级。
    - 不上读
- *策略（StarProperty）：
    - 主体S可以对客体O进行写访问，仅当O的安全级支配S的安全级。
    - 不下写
- DS策略（DiscretionarySecurityProperty）：
    - 主体S可以访问客体O，仅当S具备对O的自主访问权限。
    - 自主访问控制

BLP模型规则的另外一种表述

- SS策略：主体S可以对客体O进行读访问，当且仅当S的安全级支配客体O的安全级且S对O具有自主的读权限。
- *策略:主体S可以对客体O进行写访问，当且仅当O的安全级支配S的安全级且S对O具有自主的写权限。

BLP模型的基本安全定理

- 如果系统的初态是安全的，且系统状态的每次变化都能满足SS策略、*策略和DS策略的要求，那么系统将始终处于安全状态。

BLP模型的特性

- 强制特性
  - 强制规定主、客体的安全级别
  - 根据SS策略和*策略进行访问判决
  - 一般主体不能修改主、客体的安全级别
  - 主体不能传播访问权限
- 自主特性

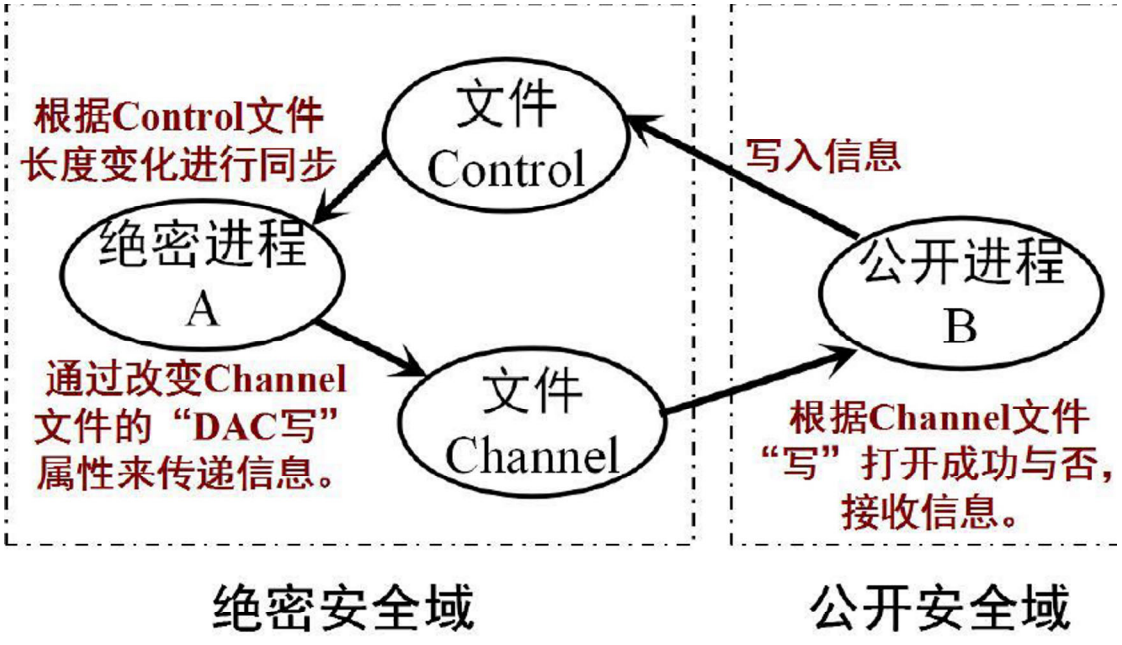
BLP模型的可用性问题

- *策略对系统可用性的影响：
  - 高安全级的主体只能生产高安全级的信息！
  - 有的系统可能无法正常运转。
- 例子：
  - 打印进程： TS （绝密）
  - 文件资料： S （机密）
  - 文件可以写到打印服务器。
  - 打印完成后，打印进程无法删除文件。
  - 原因：禁止下写

隐蔽通道

- 如果一个通信信道既不是设计用于通信的，也不是有意用于传递信息的，则称该通信信道是隐蔽的。
- TCSEC定义：允许进程以违反系统安全策略的方式传递信息的通道。

BLP模型的隐蔽通道

- 

敏感信息从高安全区域向低安全区域传递的过程

1. 进程A创建绝密信息文件data；
2. 进程B打开文件control，并写入一个字节，内容为“0”或“1”。进程A一直监控文件

control的长度，当它发现其变长时，则说明进程B已经做好了接受信息的准备，此时可以开始发送信息了；

3. 进程A改变文件data的DAC访问模式。进程A与进程B双方约定，若允许进程B写data文件，则表示进程A发送了一个二进制比特“1”；否则，表示进程A发送了二进制比特“0”；
4. 进程B试图以写方式打开文件data。若打开成功，则认为自己收到了比特位“1”；否则，认为自己收到了比特位“0”；
5. 进程B每接受一个二进制比特信息，则将其写入文件control。进程A则通过检查文件control的长度是否发生变化，确定信息传递是否正确；
6. 反复执行以上的第2~5步动作，直到敏感信息全部从进程A传给进程B。

3.2 Biba模型

- 安全目标：保护系统中数据的完整性。

(不考)数据完整性的四类（五种）定义

- 数据质量符合预期
- 防范对数据的不正确修改
- 防范对数据的非授权修改
- 禁止修改数据，或可检测对数据的任何修改。
- 限制信息单向流动

Biba模型元素

- 主体
- 客体
- 完整性级别
- 访问方式

完整性级别

- 定义： <C， S>
  - C：完整性等级（classification ）
    - C： Crucial
    - VI： V ery Important
    - I： Important
  - S：范畴集（set of categories）
- 含义：主、客体完整性等级的含义不完全相同
  - 对主体：级别越高，其可信性和可靠性就越高。
  - 对客体：级别越高，其重要性和安全性就越高。
- 比较：支配（dom）
- 表示： i（0）、 i（S）

Biba模型定义的访问方式

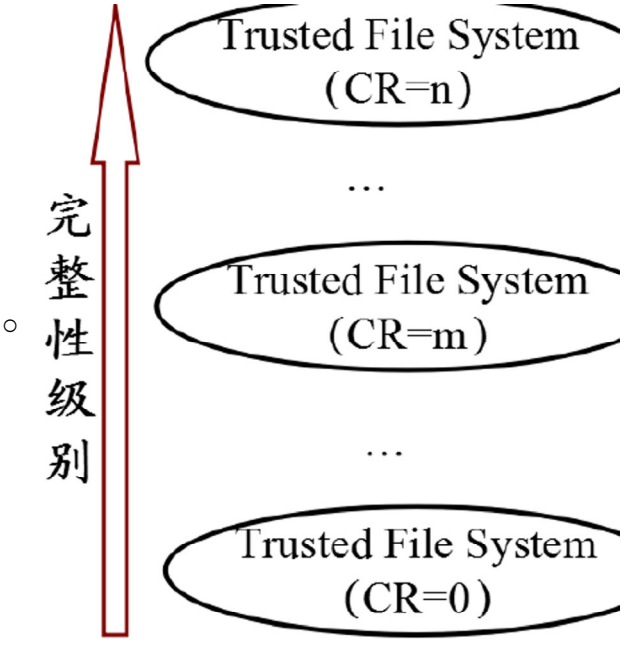
- 读（Observe）主体读取客体的信息
- 写（Modify）主体往客体中写信息
- 调用（Invoke）一个主体调用或执行另外一个主体

Biba模型的安全访问规则

(1)(不考)严格完整性策略（StrictIntegrityPolicy,SIP）

- 规则1：主体s可以对客体o进行“读”访问,当且仅当i(o)domi(s)(不下读)
- 规则2：主体s可以对客体o进行“写”访问,当且仅当i(s)domi(o)(不上写)
- 规则3：主体s1可以调用（invoke）主体s2,当且仅当i(s1)domi(s2)

(不考)SIP的实现举例：LOCUS

- 目标
  - 防止不可信软件修改数据或其他软件
- 客体的完整性级别(CredibilityRating,CR)
  - 0(Untrusted)~n(highlytrusted)
- 主体的完整性级别(RiskLevel)
- LOCUS的访问控制机制
  - 用户的初始完整性级别是他可以运行的所有程序的完整性级别的最大值；
  - 用户可以执行完整性级别比它低的程序，但须借助特殊命令。
  - 

(不考)SIPvsBLP

-

	“读”规则	“写”规则	信息流方向
SIP	禁止下读	禁止上写	从高完整性级别到低完整性级别的安全域
BLP	禁止上读	禁止下写	从低密级到高密级的安全域

BLP和Biba模型中，“上”、“下”的含义有何区别？

完整性级别vs机密性级别

- 存在如下几种可能性：
  - 完整性级别高、机密性级别也高；
  - 完整性级别高、机密性级别低；
  - 完整性级别低、机密性级别高；
  - 完整性级别低、机密性级别低。
- Biba模型中的完整性级别同BLP模型中的机密性级别之间没有必然联系，它们是独立的两套安全属性！

(2)环策略（RingPolicy,RP）

- 规则1：任何主体都可以对任何客体进行“读”访问。
- 规则2：主体s可以对客体o进行“写”访问，当且仅当i(s)domi(o)。
- 规则3：主体s1可以调用（invoke）主体s2，当且仅当i(s1)domi(s2)。

(3)针对主体的下限标记策略(LowWatermarkPolicyforSubjects,LWMPS)

- 规则1：允许主体s对任何客体o进行“读”访问，但访问完成后其完整性级别调整为min(i(s),i(o))。
- 规则2：主体s可以对客体o进行“写”访问，当且仅当i(s)domi(o)。
- 规则3：主体s1可以调用（invoke）主体s2，当且仅当i(s1)domi(s2)。

SIP、RP、LWMPS比较

策略名称	区别
SIP	主体s可以对客体o进行“读”访问，当且仅当i(o) dom i(s).
RP	任何主体都可以对任何客体进行“读”访问
LWMPS	允许主体s对任何客体o进行“读”访问，但访问完成后其完整性级别调整为min(i(s) , i(o)).

- 只有"读"访问规则不同

(4)针对客体的下限标记策略(LowWatermarkPolicyforObjects,LWMPO)

- 规则1：主体s可以对客体o进行“读”访问，当且仅当i(o)domi(s)
- 规则2：允许主体s对任何客体o进行“写”访问，但访问完成后把客体的完整性级别调整为min(i(s),i(o))
- 规则3：主体s1可以调用（invoke）主体s2，当且仅当i(s1)domi(s2)

(5)下限标记完整性审计策略(LowWatermarkIntegrityAuditPolicy,LWMIAP)

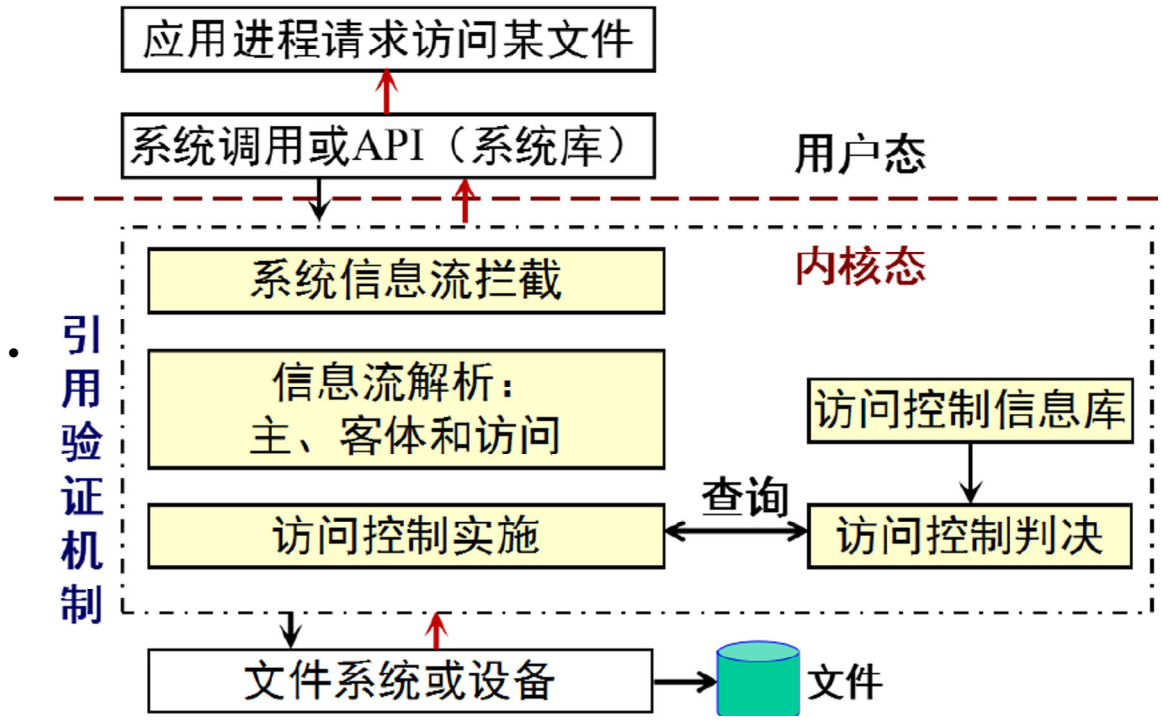
- 规则1：主体s可以对客体o进行“读”访问，当且仅当i(o)domi(s)
- 规则2：允许主体s对任何客体o进行“写”访问，但如果i(s)不支配i(o)，则对该操作进行审计
- 规则3：主体s1可以调用（invoke）主体s2，当且仅当i(s1)domi(s2)

SIP、LWMPO、LWMIAP比较

策略名称	区别
SIP	主体s可以对客体o进行“写”访问，当且仅当i(s) dom i(o).
LWMPO	允许主体s对任何客体o进行“写”访问，但访问完成后把客体的完整性级别调整为min(i(s), i(o)).
LWMIAP	允许主体s对任何客体o进行“写”访问，但如果i(s) 不支配 i(o)，则对该操作进行审计。

- 只有"写"访问规则不同

多级安全模型在操作系统中的实现



4 基于角色的访问控制(RoleBasedAccessControl,RBAC)

传统访问控制在授权管理方面的局限

- 授权工作量与用户数、客体数成正比；
- 用户职责变化时重新授权的工作量大；
- 用户离职时的权限撤销工作量大。

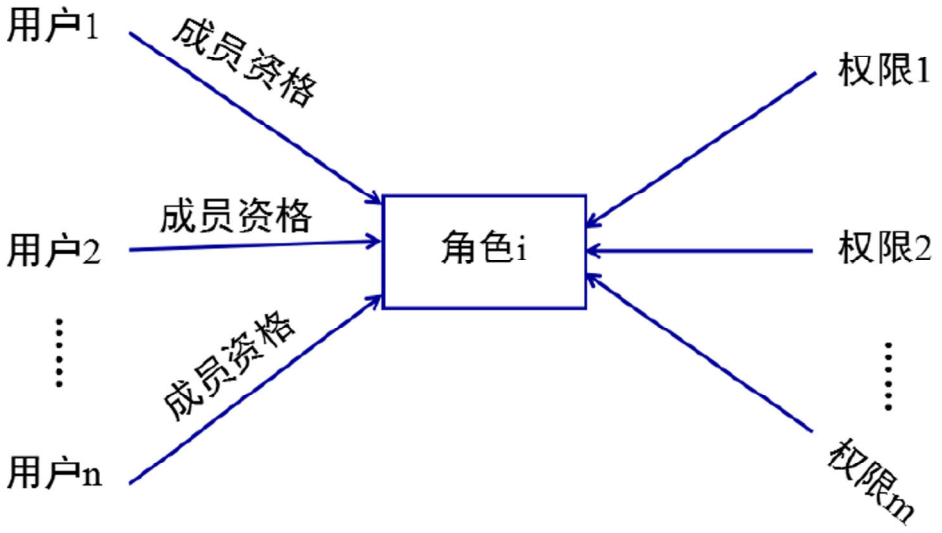
在用户与客体间引入较稳定的中间实体

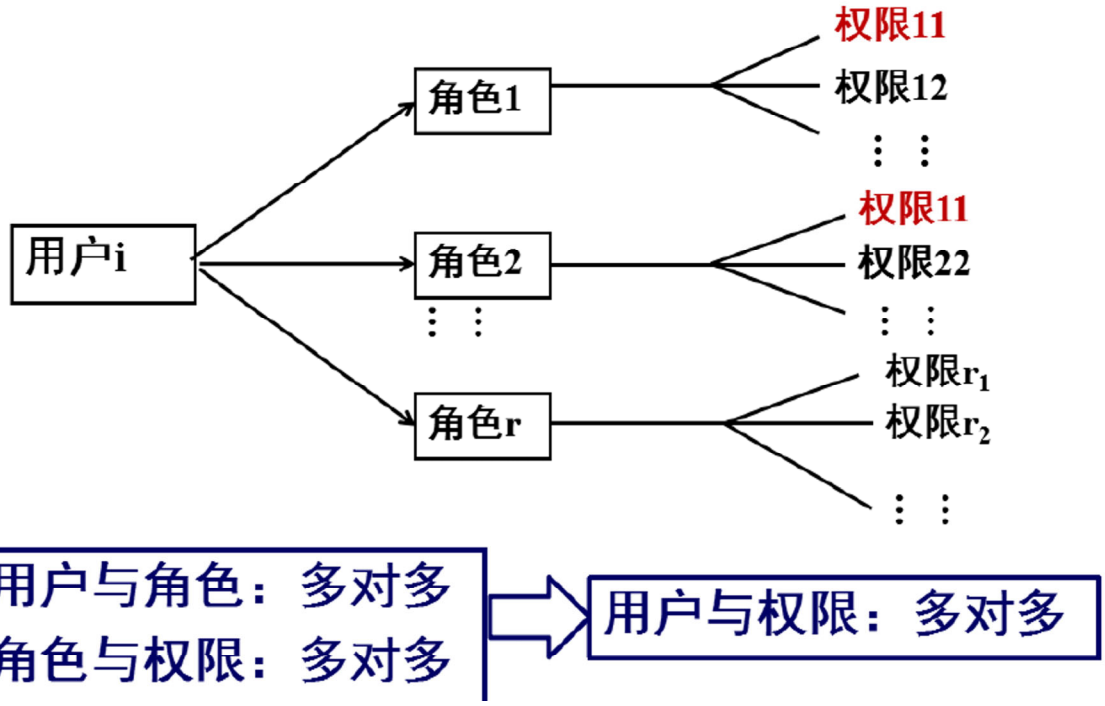
RBAC的思想

- 在一个组织机构里，为不同岗位创建对应的角色；
- 对每个角色分配不同的操作权限；
- 根据用户在组织机构中的职责或任务，为其指派相应的角色；
- 用户通过所分配的角色获得相应的权限，实现对信息资源的访问。

在用户与客体之间引入角色

用户、角色、权限的关系

- 

- 

角色的概念

- 理解1：
  - 与特定工作活动相关的一组动作和职责
- 理解2：
  - 组：用户集
  - 角色：用户集 + 权限集

RBAC模型的特点

- 中性，可以偏自主，也可以偏强制；
- 灵活，可以适应变化的需求；
- 支持最小特权原则；

- 支持职责分离原则。

特权的概念

- 定义：定义：不不受访问控制策略限制的权限受访问控制策略限制的权限
- 特权存在的原因特权存在的原因
  - 便于系统维护便于系统维护
  - 提高系统的可用性提高系统的可用性
- 特权对系统安全的危害特权对系统安全的危害
  - 被滥用
  - 被窃取
  - 被误用

最小特权原则

- 定义
  - 系统安全中最基本的原则之一。
  - 最小特权(LeastPrivilege)：主体只能被授予其完成任务所必需的特权。
  - 最小特权原则：应限定每个主体所必须的最小特权，确保可能的事故、错误等原因造成的损失最少。
  - “最小”：给予主体“必不可少”的特权，保证主体能在所赋予的特权之下完成需要完成的任务或操作；另一方面，只给予主体“必不可少”的特权，限制了主体所能进行的操作。
- 最小特权原则的内涵
  - 如果某个访问权限不是主体履行职责所必需的，那么就不应该把这个权限授予他
  - 如果主体在执行某项任务时确实需要额外的权限，那么必须在任务完成时撤销这个额外权限。
- 最小特权原则VS需知原则（NeedToKnow）
  - 主体只能访问他履行职责所需的资源。
- 最小特权原则的实现
  - 基本思想：分权
  - 把超级用户权限进行细分，分别授予不同的系统操作员或管理员，确保任何一个用户都没有足够的权限去破坏整个系统的安全策略。
  - 一种常用的特权分割方法
    - 安全管理员
    - 系统操作员
    - 安全操作员
    - 网络管理员
    - 审计管理员
- 最小特权的实现机制举例
  - 基于进程的特权机制
    - 对于系统中的每个进程，根据其所代表的用户，赋予相应的权限集。

    - 基于文件的特权机制
        - 对可执行文件赋予相应的权限集

职责分离原则SeparationofDuty（SOD）

- 角色的执行权限和角色的管理权限是相分离的，即主体不应同时拥有二类权限，否则会出现权限管理的失控。
- 指将不同的责任分派给不同的主体以期达到互相牵制，消除一个主体执行两项或多项不相容任务的风险。

核心RBAC模型（FlatRBAC）

- 用户通过角色获得权限
- 必须支持多对多的用户角色指派
- 必须支持多对多的权限角色指派
- 必须支持用户角色检查
- 用户可同时行使多个角色的权限

有角色继承的RBAC模型

在核心RBAC的基础上

- 支持角色层次（偏序）
- 增加角色继承操作
    - 受限继承：一个角色只能继承某一个角色；
    - 多重继承：一个角色可以继承多个角色，也可以被多个角色继承。

有约束的RBAC模型(ConstrainedRBAC)

- 在有角色继承的RBAC的基础上，支持权限角色检查，且较之用户角色检查，其性能是高效的。
    - 静态职责分离（StaticSeparationofDuty，SSD）
    - 动态职责分离（DynamicSeparationofDuty，DSD）

静态职责分离

- 如果两个角色直接存在SSD约束，则当一个用户分配了其中的一个角色后，就不能再获得另一个角色。
- 不能在有SSD约束关系的两个角色之间定义继承关系。

动态职责分离

- 引入的权限约束，作用于用户会话激活角色的阶段。
- 如果两个角色直接存在DSD约束，系统可以将这两个角色都分配一个用户，但该用户不能在一个会话中同时激活它们。