绪论

信息安全概念

信息就是数据 → 信息安全 = 数据安全 + 系统安全

信息安全学关注信息本身的安全（不丢、不坏、不冒、不泄），信息安全的任务是保护信息财产。

CIA模型：保密性（防止未授权信息泄露）、完整性（防止未经授权的信息修改和破坏）、可用性（防止未经授权的信息或资源被截留）

系统研究的方法：

还原论：把系统分解成它的组成部分， 通过对系统的

组成部分的研究去了解原有系统的情况

整体论：把一个系统看成一个完整的统一体， 一个完整的被观察单位， 而不是简单的微观组成元素的集合

整体特性：综合特性（可以分解为系统组成部分的特性 ）、 涌现性（不可分解为系统组成部分的特性）

安全性属于涌现性

系统安全工程

系统工程：涵盖系统生命周期的具有关联活动和任务的技术性和非技术性过程的集合

系统安全工程：把安全性相关活动和任务融合到系统工程的过程之中， 形成的一个系统工程专业分支，力求从系统生命周期的全过程去保障系统的安全性

系统安全思维

- 运用整体论思想分析安全问题
- 在系统的全生命周期中衡量系统的安全性
- 通过系统安全工程措施建立和维护系统的安全性

风险评估：按照标准和规范，对信息系统的资产、威胁、漏洞以及已采取的防护措施等进行分析，判断安全事件发生的概率以及可能造成的损失，提出风险管理措施的过程

风险 = 资产 × 威胁 × 漏洞

安全测评：

对信息系统的安全保障能力进行评估，分析系统当前的安全运行状况、查找存在的安全问题、提供安全改进建议，从而降低系统的安全风险

等级保护、风险评估和安全测评的关系：

等级保护是一项制度，风险评估、系统测评都是在等级保护制度下，评价信息系统安全性的方法

对于等级保护制度来说，风险评估是等级保护的出发点，安全测评等级保护的落脚点

对于一个信息系统来说，风险评估是安全建设的起点，系统测评是安全建设的终点

攻击树：树根、节点、边（权值，攻击成本等）

计算机安全基础

计算机安全内涵

PDRR——Protect、Detect、React、Restore

PDR——Prevent、Detect、React，去掉restore

PPDRR——在PDRR基础上加Policy（策略）

信息安全的3A：

- 认证 (Authentication)：验证身份
- 授权 (Authorization)：控制主体如何访问客体的策略
- 审计 (Auditing)：记录和跟踪与安全有关的事件

4A再加上记账 (Accounting)

计算机安全的五个设计原则：

1. 在一个给定的应用中，一个计算机系统中的安全保护机制应该聚焦于数据、操作还是用户?
2. 一个安全机制应该放置在计算机系统从硬件到应用的哪个层次?
3. 如何防止攻击者访问安全机制下面的层?
4. 定义和实施安全的任务应该交给一个中央实体，还是系统中的每个成员?
5. 相比富有特色的安全环境，你是否更偏爱简单性和更高的保障?

隐信道：

1. 定义：系统中不受安全策略控制的、违反安全策略的信息泄露途径
2. 特点：一次传递一个比特信息
3. 关键属性：存在性与带宽
4. 实现： 利用资源共享

隐存储信道：

1. 基础：利用资源是有限的并且是共享的这一特点
2. 实现：通过占用共享资源来实现信息的发送和接收
3. 必要条件：资源共享、发送信息、接受消息、同步或者协同

隐定时信道：

1. 基础：利用计数器的值是递增分配的的这一特点
2. 实现：通过改变共享计数器的值来传递信息

3. 必要条件：资源共享、基准时间、同步或者协同.判断是否是隐定时信道的重要依据：是否有基准时间

侧信道攻击： 也叫旁道攻击，指的是不攻击密码本身，而是攻击那些实现于不安全系统上的加密系统。利用的信息有声音、时间、电磁辐射、能量消耗等

身份识别与认证

口令空间计算：平均值等于暴力穷举的一半

身份认证基于的信息：秘密信息、物理令牌、生物特征、行为特征、位置信息

1. What you know（口令，PIN，令牌等）；
2. What you have（物理令牌如钥匙等，USB Key）；
3. Who are you（生物技术）；
4. How to do（签名，打字，手势等）；
5. Where are you（地理位置等）

口令认证机制面临的威胁：口令猜测、口令欺骗、口令文件泄露、遗忘

口令认证机制的缺点：口令在网络中传输时是很容易被窃取或攻击——网络数据流窃听、认证信息截取/重放、字典攻击、穷举尝试 、窥探口令 、骗取口令、垃圾搜索

1.用户长期使用同样口令，没有妥善保管，使用弱口令等不安全的使用措施，使得攻击者通过穷举或者猜测攻击破解口令；

2.欺骗攻击与重放攻击。攻击者伪造网页，或者监听获得口令或通信内容，重放通信内容冒充用户；

3.未加密的口令。口令文件泄露等情况

访问控制

访问控制基本原则：

1. 最小特权原则——只授予每个主体所必需的最小特权
2. 多人负责原则——授权分散化，保证没有任何个人具有完成任务的全部授权或信息
3. 职责分离原则——将不同的责任分派给不同的人员以期达到互相牵制

概念：在身份认证的基础上，依据授权对提出的资源访问请求加以控制

访问控制的核心： 授权控制，即控制不同用户对信息资源的访问权限

授权： 访问监控器决定访问是否被准许或拒绝

一个访问操作的活动实体，叫做**主体(subject)**或**主角(principal)**，使用某种特定的**访问操作(access operation)**去访问一个被动的实体，叫做**客体、对象、目标(object)**

主角和主体

主角和主体都是用来指代一个访问操作的活动实体

- 主角：能被授予访问对象的权限或做出影响访问控制决策的声明
- 主体：IT系统的活动实体

例如：主角- 用户ID；主体 – 用户进程

主角和主体的使用场合不同，在谈到安全策略时使用主角，在谈论执行安全策略的系统时使用主体。访问决策时，主体必须和主角绑定。

访问操作

在最基本的级别上，主体可以观察或者改变客体。可以定义两种访问模式(assess mode)：

- 查看(observe)：查看客体的内容
- 修改(alter)：改变客体的内容

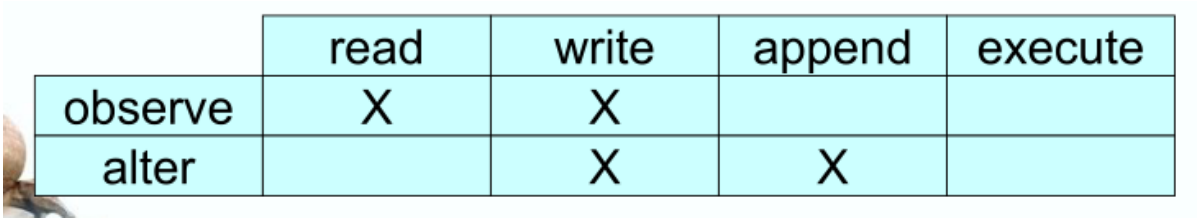
Bell-LaPadula安全模型的访问权限(access right)

Multics操作系统中的访问属性(access attribute)

访问权限： 主体对客体进行访问的具体形式。例如，读、写和执行等

BLP模型（详见第8章）有四种访问权限

- 读（read）
- 写（write）
- 添加（append），又叫做盲写（blind write）
- 执行（execute）



	read	write	append	execute
observe	X	X		
alter		X	X	

操作系统可以不打开文件而使用文件，因此引入“执行权限”，其查看和修改这两种访问模式都不包括

Multics系统，访问属性和访问权限之间的对应关系

数据段		**目录段**	
读	r	状态	r
执行	e, r	状态和修改	w
读和写	w	添加	a
写	a	搜索	e

DAC 和ACL 对应， MAC 和有着偏序关系的安全等级 (级别) 对应。二者直接赋予使用者权限。RBAC将访问权限分配给角色， 用户通过被指派为角色从而获得角色所拥有的访问权限 。

DAC自主访问控制(Discretionary Access Control)

为每个资源定义一个所有者，让所有者规定谁可以访问，这类策略被称为自主的，因为访问控制由所有者来处理。

MAC强制访问控制(Mandatory Access Control)

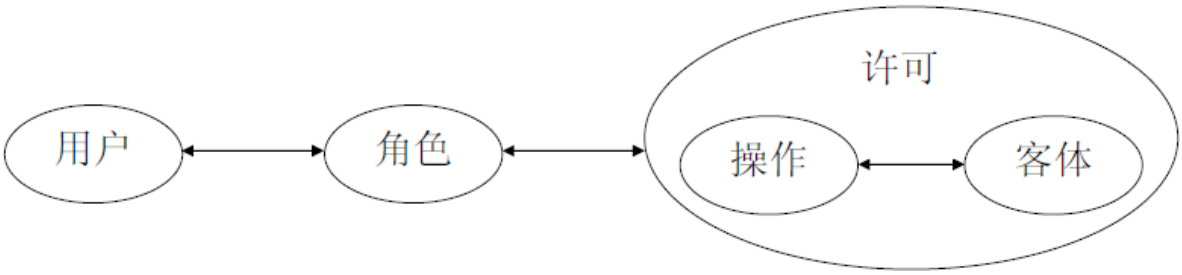
系统策略规定谁可以访问，这类策略被称为强制的。

RBAC 基于角色的访问控制（Role-based access control）RBAC主要用于应用系统，如数据库管理系统

RBAC0：

基本思想： 许可被授权给角色，角色被授权给用户， 用户不直接与许可关联。RBAC对访问权限的授权由系统管理员统一管理，用户不能自主地将访问权限传给他人。 管理员通过为用户分配角色、取消用户的某个角色等操作管理角色分配 。

用户users（一个可以独立访问计算机系统中的数据或者用数据表示的其他资源的主体 ）、 角色roles（一个组织或任务中的工作或者位置，它代表了一种权利、资格和责任 ）、 客体objects、 操作operations、许可permissions（允许对一个或多个客体执行的操作 ）



会话(sessions)： （用户 ，激活的角色）集合。一次会话是用户的一个活跃进程，它代表用户与系统交互

RBAC1：引入了角色间的继承关系 。角色间的继承关系可分为一般继承关系（绝对偏序，多继承）和受限继承关系（树结构，单继承）

RBAC2：添加了责任分离关系,RBAC2 的约束规定了所应遵循的强制性规则（权限被赋予角色时、角色被赋予用户时、当用户在某一时刻激活一个角色时 ）

RBAC3：RBAC3包含了RBAC1和RBAC2，既提供了角色间的继承关系，又提供了责任分离关系

TBAC基于任务的访问控制 ， 是一种基于实例（instance-based） 的访问控制模型——一个例子就是操作系统中的进程管理

组成：工作流(Work flow)、授权结构体(Authorization unit)、授权步(Authorization step)、受托人集(Trustee set)、许可集(Permissions set)、任务(Task)、依赖(Dependency)

在TBAC中， 对象的访问权限控制不是静止不变的， 而是随着执行任务的上下文环境发生变化 。在工作流环境中， 数据的处理与上一次的处理相关联， 相应的访问控制也如此， 因而TBAC是一种上下文相关的访问控制模型。

TBAC不仅能对不同工作流实行不同的访问控制策略， 而且还能对同一工作流的不同任务实例实行不同的访问控制策略

访问控制矩阵

为每个主体和客体的组合制定访问操作访问控制矩阵

$M = (M_{so})_{s\in S,o\in O}$，当$M_{so}\in A$

S - 主体集合； O - 客体集合； A - 访问操作集合

M_{so}表项规定了主体s可以施加于客体o上的访问操作的集合

	bill.doc	**edit.exe**	**fun.com**
Alice	-	{执行}	{执行,读}
Bob	{读,写}	{执行}	{执行,读,写}

- 访问能力表（Capabilities Lists， CL）：访问权限和主体保存在一起，矩阵的行
- 访问控制列表（Access Control Lists，ACL）：访问权限和客体保存在一起，一个ACL = 访问控制矩阵中的一列

中间结构

- 组
- 否定的许可
- 特权
- 角色：一组特定应用的操作成为角色，主体从他们履行的角色上获得访问权限

> 组和角色的不同点是什么，它们是否有本质的区别?
>
> 从定义上来看， 具有相同访问权限的用户集合形成组，然后再给授予访问对象的许可。而一组特定应用的操作（过程）称为角色，主体从它们履行的角色上获取访问权限。这两种本质上没有区别，都是结构化中间控制的实现方法，都有利于用一种简单的管理方式来实现访问控制

在主体和客体之间，可插入的中间层次有：

1. 角色：一个角色就是一组过程，角色被分配给用户。一个用户可以有多个角色，多个用户也可以有相同的角色
2. 过程：过程是语义比读或写更复杂的高层访问控制方法。过程只能被应用于某些数据类型的客体上
3. 数据类型：每个客体均属于一种特定的数据类型，只能通过为该数据类型定义的过程来访问。通过限制可以访问客体的过程来控制对客体访问

保护环：用于说明主体和客体之间中间层。主体和客体均依照“重要性”被赋予一个数字，通过比较主体与客体的数字进行访问控制决策。数字 → 偏序 → 多级安全策略

进程按照以下规则获得他们的数字：0 – 操作系统内核、1 – 操作系统、2 – 实用程序、3 – 用户进程

偏序关系

集合L上的一个偏序≤是L×L上的一个关系，它是：

- 自反 对于所有a∈L， a≤a成立
- 传递 对于所有a,b,c∈L， 如果a≤b且b≤c， 那么a≤c
- 反对称 对于所有a,b∈L， 如果a≤b且b≤a， 那么a=b
- 如果两个元素a,b∈L是不可比的， 记为a ⋦b

哈斯图是偏序集的图示， 有向图。 对a,b∈L，置一条a到b的边， 当且仅当a≤b且a≠b， 且不存在c∈L，使得a≤c≤b且a≠c，b≠c

安全级别的格

安全策略： 仅当主体的安全等级高于或等于客体的安全等级时，主体才可以查看客体

1. 给定两个不同安全级别的客体， 一个主体至少必须具有什么样的安全级别才能允许读这两个客体？对于主体s和客体o1、 o2: C(s)≥C(o1)且C(s)≥C(o2)，即最小上界
2. 给定两个不同安全级别的主体， 为使一个客体能被两个主体查看， 该客体能够拥有的最高安全级别是多少？对于主体s1、 s2和客体o: C(s1)≥C(o)且C(s2)≥C(o)， 即最大下界

格（L, ≤） 由集合L和偏序≤组成， 对于任意两个元素a, b∈L， 存在一个最小上界（Least upper bound） u∈L和一个最大下界（Greatest lower bound） l∈L，即

- a ≤ u, b ≤ u, 并且对所有v∈L: (a≤v ∧ b≤v) ➡️ (u ≤ v)
- l ≤ a, l ≤ b, 并且对所有k∈L: (k≤a ∧ k≤b) ➡️ (k ≤ l)

在安全中， 如果a≤b， 则说“a受b支配” ('a is dominated by b')或者“b支配a”

受所有其它级别支配的安全等级称为系统低（system low）

支配所有其它级别的安全等级称为系统高（system high）

系统低和系统高不一定都存在；但如果存在，则一定是唯一的

如果集合L是有限的，那么其系统高和系统低一定是存在的

pf：根据格的定义，对于任意两个元素a, b∈L， 存在一个最小上界（Least upper bound） u∈L和一个最大下界（Greatest lower bound） l∈L

证明：因为L是有限集合，不妨设L是n元格，$L=\{a_1,a_2,\ldots,a_n\}$.
则它有全下界 $a_1 \wedge a_2 \wedge \ldots \wedge a_n$，记为$b$. 全上界$a_1 \vee a_2 \vee \ldots \vee a_n$，记为$c$.
对于任意$x \in L$，有$b \leq x$，即b受其他所有级别支配，为系统低.
对于任意$x \in L$，有$c \geq x$，即c支配其他所有级别，为系统高.

安全标签

保护机密信息的安全策略：

- 文件制定了安全级别(security level)
- 一个用户的许可表明了该用户可以访问哪些文件

于是引入安全标签——

令H为一个密级集合，具有分等级（线性）的排序$\leq_H$

令C是一个种类集合，如工程名字、公司部门、学院的系等。一个间隔就是一个种类集

安全标签是一个二元组（h, c），其中h∈H是一个安全级别，而c∈C是一个间隔

安全标签的偏序≤定义为：(h1, c1) ≤ (h2, c2)

当且仅当h1 $\leq_H$ h2和c1 ⊆ c2

使用控制

UCON的ABC模型：

主体（属性）——权限——客体（属性）

权限相关：授权规则（A）、义务（B）、条件（C），三种条件可以选择在使用前（pre）和使用中（on）满足

属性可变性：不改变(0)、使用前改变(1)、使用中改变(2)、使用后改变(3)

不存在：preA2、preB2、以及对preC OnC，只能0。故一共4×4-2+2=16

决策因素	0(不可改变)	1(使用前改变)	2(使用中改变)	3(使用后改变)
PreA	Y	Y	N	Y
OnA	Y	Y	Y	Y
PreB	Y	Y	N	Y
OnB	Y	Y	Y	Y
PreC	Y	N	N	N
OnC	Y	N	N	N

- 若决策因素是“preA”或“preB”，则属性改变只能在使用前或使用后，而不可能在使用中
- 以条件为决策因素(preC、 onC)的访问模型中所有更新属性的组合都标为“N”，因为“条件”只与环境或系统状态有关，不能改变任何主客体属性

基本模型：比如UCONpreA1，表示以授权规则为决策因素，并在使用前改变主客体有关可变属性的基本模型

用ABC描述DAC、MAC、RBAC

DAC、MAC、RBAC由$UCON_{preA0}$模型支持

DAC:

(1)S表示主体即用户或用户组，O表示客体，N表示身份标记集合，R为操作权限集

(2) id表示用户到标记集合N之间的一对一映射关系

(3) ACL表示客体与N×R之间的映射关系，其中N×R为用户身份与操作权限的组合关系

(4) ATT(S) = {id}

(5) ATT(O) = {ACL}

(6) allowed(s，o，r) => (id(s)，r) ∈ ACL(o)。其中allowed(s，o，r)表示主体s对客体o具有执行r操作的权限

MAC

(1) L表示具有偏序关系的安全级别集合；

(2) clearance: S → L，是访问主体S和安全级别L之间的映射函数

(3) classification: O → L，表示客体和安全级别L之间的映射函数

(4) ATT(S)={clearance}

(5) ATT(O)={classification}

(6) allowed(s，o，read) => clearance (s) ≥ classification (o)，即主体可以“下读”客体

(7) allowed(s，o，write) => clearance (s) ≤ classification (o)，即主体可以“上写”客体

RBAC1:

(1) P = {(o，r)}，P表示许可集合，(o，r)为客体-权限对

(2) ROLE表示角色层次的偏序关系

(3) actRole表示激活角色，实现“用户-角色”分配

(4) pRole表示授权角色，实现“角色-许可”分配

(5) ATT(S) = {actRole}；

(6) ATT(O) = {pRole}

(7) allowed(s，o，r) = role ∈ actRole(s) ∧ role' ∈ pRole(o，r) ∧ role ≥ role'，即如果存在授权角色(pRole(o，r))，其偏序关系≤激活角色(actRole(s))，则访问请求被允许

实际例子：

(1) credentials是工作证集合

(2) cert: S → $2^{credentials}$，是用户与工作证的映射关系

(3) groupID: O → $2^{credentials}$，是客体资源与工作证的映射关系

(4) ATTS(S): {cert}，即主体持有的工作证

(5) ATTS(O): {groupID}，即客体资源允许持有特定工作证的用户访问

(6) allowed(s，o，read) => cert(s) ≠ Φ，即组织内员工可读组织内的全体客体资源

(7) allowed(s，o，write) => (cert(s) ≠ Φ) ∧ (groupID(o) ∈ cert(s))，即组织内客体资源只允许某些持有特定工作证的用户修改

访问控制器RM

访问监控器RMReference Monitor，一个访问控制概念，指所有主体对客体访问进行仲裁的抽象机器。RM可放置在分层系统的任何地方

访问控制机制的理论基础是访问监控器

安全内核SK：实现 RM 的可信计算基的硬件、固件和软件的总和，必须处理所有的访问控制，必须保护安全内核，使之不被修改，并能够证明是正确的。

可信计算基TCB：是计算机系统中全部的保护机制的总和. 操作系统的安全依赖于一些具体实施安全策略的可信的软件和硬件. 这些软件、硬件和负责系统安全管理的人员一起组成了系统的可信计算基 (Trusted Computing Base， TCB)

橘皮书定义：SK是RM的实现，TCB包括了SK

访问控制机制（RVM）的三个核心要求：

1. 具有自我保护能力：以保证访问验证机制即使受到攻击也能保持自身的完整性
2. 总是处于活跃状态，是所有访问请求的唯一入口(不可旁路)：以保证程序对资源的所有访问都得到访问验证机制的仲裁
3. 设计得足够小：以利于分析和测试，保证访问验证机制实现的正确性和符合要求

受控调用：系统在超级用户模式只执行一组预先定义的操作，而在控制交回给用户前返回用户模式。这个过程称为受控调用。

这里涉及了保护环的概念. 对于所有应用程序，按照其属性分为不同的安全级别，对应在保护环上就是，从外到内安全级别逐渐升高。从安全出发，必须对那些请求较高特权的操作进行访问管理。当外层级别低的程序需要调用内层级别高的程序时，需要通过门，其访问是受到限制的，我们称之为，受控调用。

门：一个指向某个程序（在某个代码段中）的系统客体，门具有与它指向的代码不同的特权级别，门允许对内环的程序进行只执行，但仍限制对外的调用

如果一个过程要使用门，门必须和过程位于同一个环内

调用程序不能写入内环

可信路径

交互安全敏感数据时 ，需要建立一条从输入/输出设备到 TCB 的可信路径

计算机实体安全

可信计算：如果实体的行为总是以预期的方式，朝着预期的目标，则一个实体是可信的 （计算机终端是安全的源头 ）

用途：风险管理、数字版权管理、电子商务、安全监测和应急

TPM

“可信计算”技术的核心是称为TPM(**可信平台模块**)的安全芯片。

TPM实际上是一个含有密码运算部件和存储部件的小型片上系统(System on Chip， SOC)，由CPU、存储器、 I/O、密码运算器、随机数产生器和嵌入式操作系统等部件组成。

TPM技术最核心的功能在于对CPU处理的数据流进行加密，同时监测系统底层的状态。

软件狗

电子设备“软件狗”，也称为电子“锁”、 USB-Key。软件运行前要把这个小设备插入到一个端口上， 在运行过程中程序会向端口发送询问信号，如果“软件狗”给出响应信号，则说明该程序是合法的。

实体安全

机房的安全考虑：场地选择、结构防火、机房内部装修、活动地板、供配电系统、空调系统、其它设备和辅助材料、火灾报警及消防设施、防水安全计算机房、防静电计算机房、防雷击、防鼠害、电磁波的防护

计算机工作环境：温度10-35度，湿度40%-60%

Unix/Linux安全

Unix实现了自我访问控制，**粒度是owner(属主)，group(组)，other**

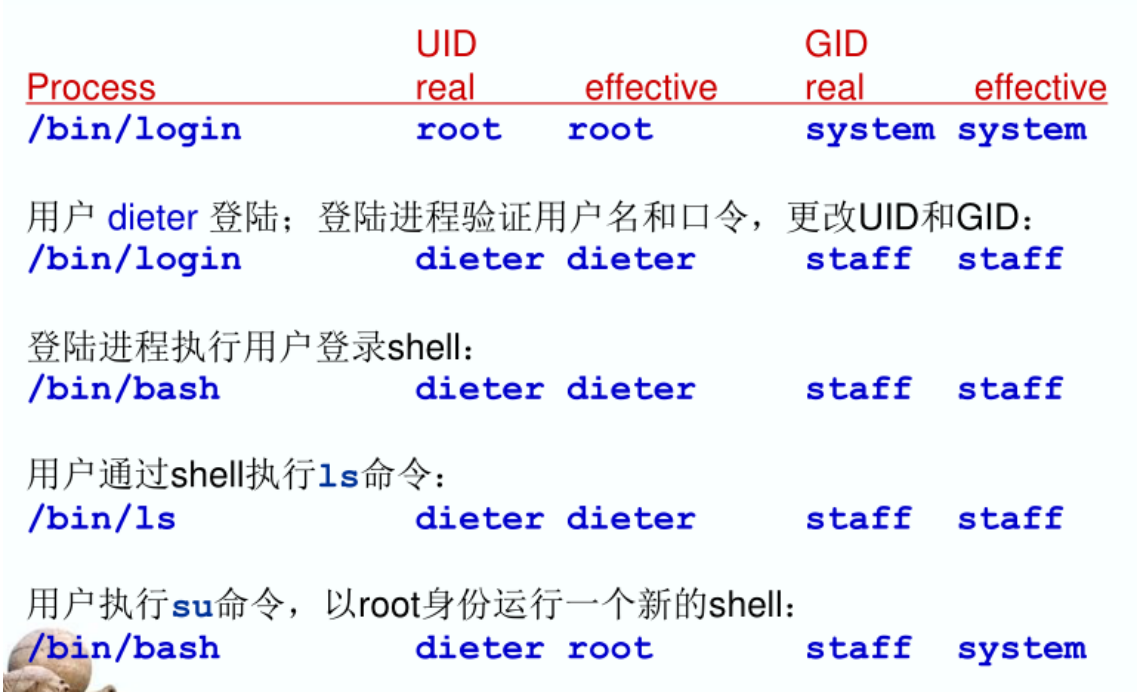
主角：用户标识符(UID) 和组标识符(GID) ，都是16位数字。超级用户(root)的UID总是0 。

组：用户属于一个或多个组，每个用户属于一个基础组(primary group)

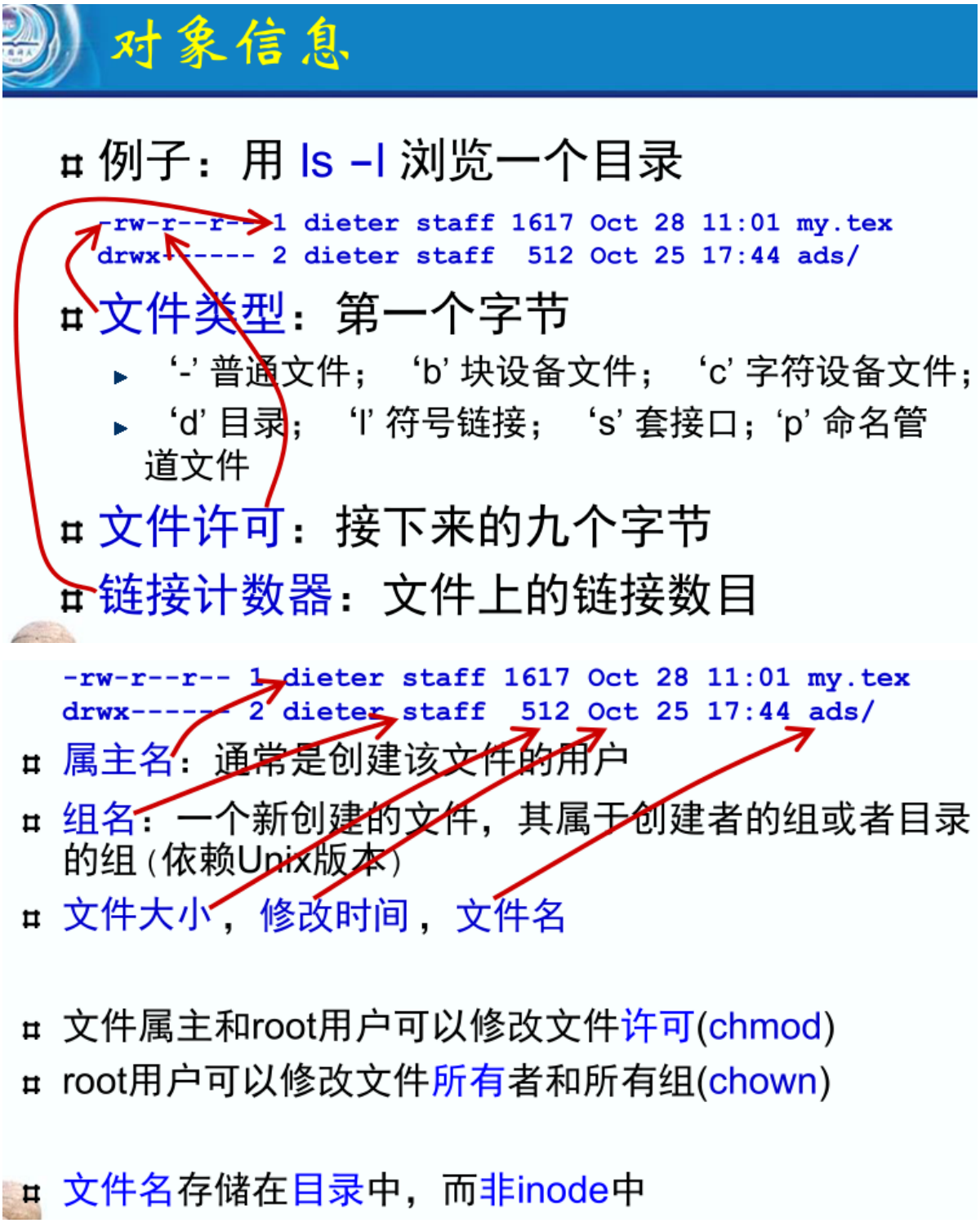
主体：主体是进程。每一个进程都有一个进程标识符(PID) 。

每个进程都有一个真实(Real)UID/GID和一个有效(Effective)UID/GID

- RUID/RGID:继承于父进程，通常是登录用户的UID/GID。进程为谁工作?
- EUID/EGID:继承于父进程或正在被执行的文件。进程拥有的权限与谁相同?



客体：文件、目录、存储设备以及I/O设备被统一视为资源(resource) 。这些资源是访问控制的对象 。资源被组织成树型结构的文件系统。目录中的每一个文件表项都是一个指针，指向名为i节点(inode，索引节点)的数据结构 。



文件许可位：文件许可位分为三组，分别定义了属主(owner)、 属组(group)以及其他人(other)的读、写和执行的权限，每组3bit（八进制可以用0~7表示），分别代表读（Read）、写（Write）、执行（Execute）权限。也可简单表示为rwx。

新建文件默认权限666，新建程序默认777，可以用Umask调整。例：默认666，umask 077，先按位取反为700，再与666按位与为600。umask 777拒绝所有访问，000不做任何限制

三个附加位：

SUID：设置UID为文件属主(也就是使得当前进程的有效 UID 为属主 UID)

SGID：设置GID为文件属组(也就是使得当前进程的有效 GID 为属组 GID)

粘滞位(Sticky bit，防删除)：只有在用户是文件的属主、目录的属主并且拥有写权限的情况下或者用户为超级用户的情况下，才能被删除或重命名。

文件许可位中， SUID、 SGID、粘滞位在最前面依次排列. 对于 SUID 程序，文件许可位的八进制表示最高位就是 4 。例子：

文件	uid	gid	set uid标志位	set gid标志位	文件许可位
Program1	10	12	1	0	4551
File1	10	12	0	0	0600

访问控制：如何用 UID/GID 和文件许可位进行访问控制.

Unix 中使用有效UID/GID 进行访问控制，访问控制时文件许可位检查顺序为：

1. 检查 UID：如果是文件属主，则用户的权限就是文件许可位中表示的文件属主的权限；如果不是属主，进行第 2 步
2. 检查 GID：如果是文件属组，则用户的权限就是文件许可位中表示的文件属组的权限；如果不是属组，进行第 3 步
3. 其他人访问许可位决定用户是否可以获取访问权限

从上面的检查顺序可以发现，如果用户为文件属主，那么拥有的权限就是文件许可位中属主的权限。 此时如果用户同时也是属组，属组的权限与其无关，也就是说，即使属组的权限比属主更多，用户也无法拥有属组权限 (和属主权限比多出来的部分)——属主优先

chmod：访问权限可以用chmod命令修改

只有文件属主或超级用户root才有权限执行

两种命令格式：

1. 绝对模式：
   • 格式： chmod [-fR] [修改后的文件许可位 (八进制表示)] filename
   • 解释： f 强制 (force)； R 递归 (Recursion)
   • 示例： chmod 0754 filename 表示将该文件的许可位修改为0754(属主可读、写、执行，属组可读、执行，其他人可读)
2. 符号模式：chmod u+wrx,g+rx,g-w,o+r,o-wx filename
   • 格式： chmod [-fR] [who] [+-=] permission filename
   • 解释： f 强制 (force)； R 递归 (Recursion)； who——u(属主)、 g(属组)、 o(其他人)、 a(所有人)； permission——r(读)、 w(写)、 x(执行)、 s(SUID)、g(SGID)、 t(粘滞位)
   • 示例： chmod u+wrx,g+rx,g-w,o+r,o-wx filename 表示将该文件的许可位修改为 0754(属主可读、写、执行，属组可读、执行，其他人可读)

用chown命令可改变文件的属主：chown nOwner:nGroup filename

SUID/SGID 受控调用：

在执行某些系统调用时， Unix需要超级用户权限

Unix解决方法：通过SUID(set userID)程序和SGID(set groupID)程序，SUID(SGID)程序与属主或属组的有效UID或有效GID一起运行， 拥有暂时的或者受限制的访问权限。这些访问权限通常情况下是不赋予普通用户的。

当root是SUID程序的属主，这时运行该程序的用户就会在执行期间获得超级用户的状态。 （即sudo ）

强制访问控制（SELinux 优点、类型、域、域切换）

自主访问控制(DAC)问题：文件/目录的所有者可以对文件进行所有的操作，这给系统整体的管理带来不便。

SELinux(Security Enhanced Linux) 是美国国家安全局（NSA）对于强制访问控制的实现。

优点：

- MAC(Mandatory Access Control) --对访问的控制彻底化。
- TE（Type Enforcement，类型强制）--对于进程只赋予最小的权限
  - 对所有的文件file（客体）都赋予一个叫type的文件类型标签
  - 对于所有的进程process（主体）赋予各自的一个叫domain的标签
  - domain标签能够执行的操作是由access vector在策略里定好的

> DTE模型（Domain， Type， Entity），把主体分到不同的域(Domain)， 客体设定不同的类型(Type)，主体和客体都被称为实体(Entity)，根据域和类型综合判断进行访问控制。
>
> SETE -- SELinux Type Enforcement

- domain迁移 -- 防止权限升级
- RBAC（role base access control) --对于用户只赋予最小的权限

域切换：allow、type_transition

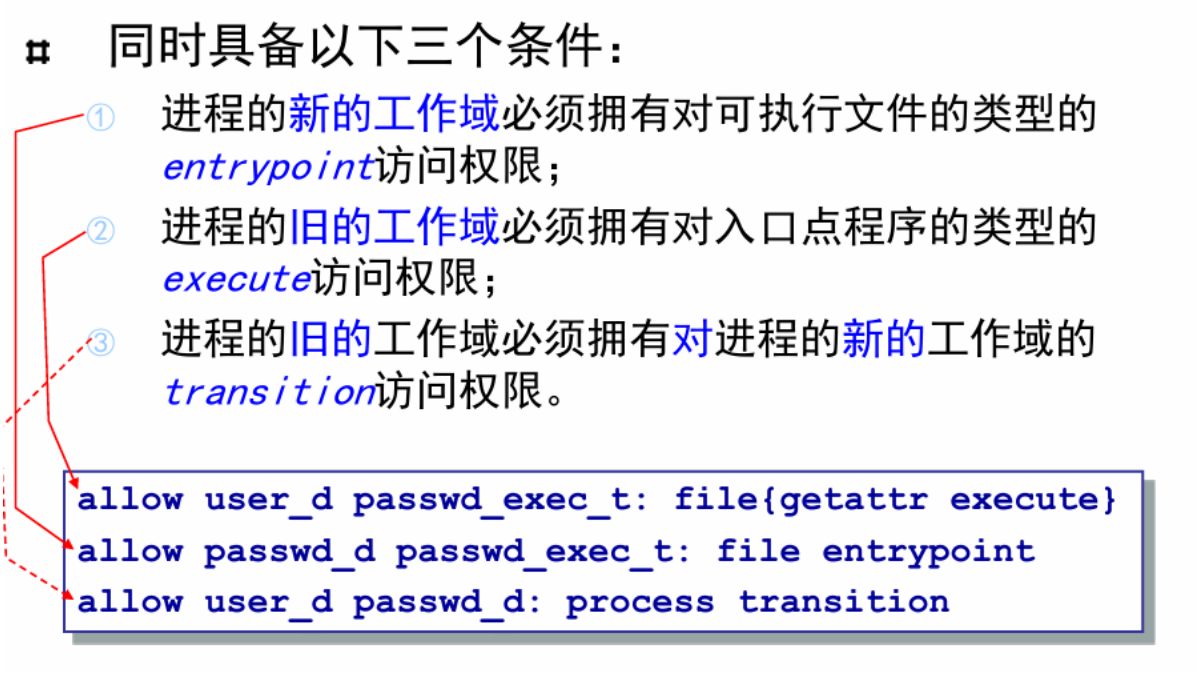
*allow*规则

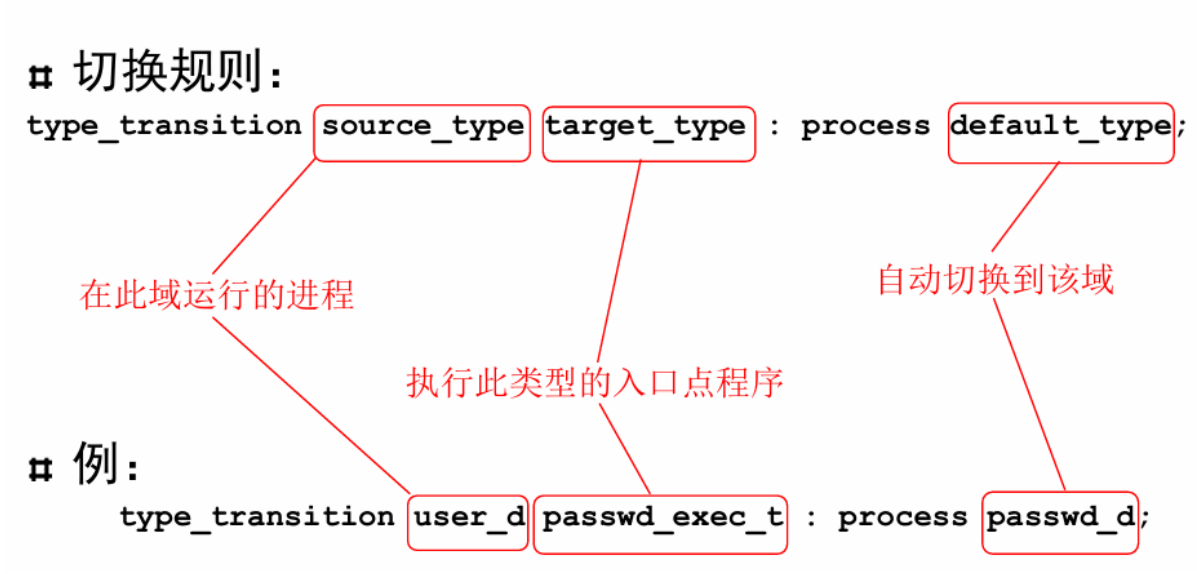
```
allow source_type target_type : object_class perm_list;
```

例：

```
allow user_d bin_t : file {read execute getattr};
```

在SETE模型中，进程为实现工作域切换必须满足的条件





Android安全

Android系统架构：分层架构

应用程序层：Java程序

应用程序框架层：Dalvik虚拟机

系统运行库层：C/C++程序

Linux核心层：内核、驱动

应用开发语言：Java，C/C++

Dalvik： Google为Android系统设计的JVM，并不遵循Java官方的JVM规范，执行的是特有的DEX文件格式

指令集基于寄存器架构（Java基于栈），执行其特有的文件格式：dex字节码（dex字节码将多个文件整合成一个，提高查找速度，减少整体文件尺寸等） 来完成对象生命周期管理、堆栈管理、线程管理、安全异常管理、垃圾回收等重要功能。

Dalvik是一个应用，一个虚拟机实例，一个进程，即每个应用都运行在一个虚拟机中，每个虚拟机都是一个独立的进程空间。虚拟机关闭或中止不会影响其他虚拟机，保护应用的安全和独立运行。

核心内容是实现库（libdvm.so），大体由C语言实现，高性能，编译时间更短 。

ART： Android Runtime，安卓4.4版本ART取代Dalvik。使用Ahead-Of-Time (AOT) 编译（在应用安装时就预编译字节码到机器语言）

优点： Android程序运行的根本机制改变了，程序启动更快，更省资源 → Android更流畅了，同时续航能力显著增加

缺点： 增加程序安装所需的时间；安装后的文件占用更多的空间

安卓系统的安全机制:

1. 进程沙箱隔离机制
2. 用户ID机制
3. 权限机制
4. 签名机制
5. SEAndroid机制

安卓最重要安全设计：沙箱和权限（就记这两个）

最主要安全机制：用户ID、权限、签名

沙盒

进程沙箱隔离机制，使得Android应用程序在安装时被赋予独特的用户标识（UID），并永久保持。应用程序及其运行的Dalvik虚拟机运行在独立的Linux进程空间，与其它应用程序完全隔离。除非显示声明premissions，来获得额外能力。是静态在程序中声明，所以会在程序安装时被知晓并不会再改变。

如果其他程序拥有该应用程序的共享的UID，即设置自己的sharedUserID为该应用程序的UID，其他程序就可以突破沙盒的限制访问其数据

权限

权限：应用能做的事，在安装时确定，而不在运行的时候。

一个权限主要包含三方面的信息：名称、属于的权限组、保护级别

- 权限组：把权限按功能分成不同集合
- 保护级别：权限通过protectionLevel来标识保护级别，分为：普通、危险（在安装应用时会提醒用户）、签名（具有同一签名的应用才能访问）、系统/签名（不推荐使用）

普通级别和危险级别属于比较低的级别，申请即可授予，签名级别和系统/签名级别需要使用者的应用程序和系统使用同一个数字证书才可以申请成功 。

Windows安全

- 微内核(microkernel)：Window NT, Minix, QNX，鸿蒙
  - Minix建立在分模块之上，模块间以信息传递联系
  - 微内核是一个信息中转站，自身完成很少功能，主要是传递一个模块对另一个模块的功能请求（消息传送与处理）
  - 理论上，微内核的思想更好些，微内核把系统分为各个小的功能块，降低了设计难度，系统的维护与修改也容易，可靠性更高，但通信带来的效率损失是个问题
- 宏内核(macrokernel)：Unix, Linux
  - Linux内部也分模块，但在运行时是一个独立的二进制大映像，模块间通讯是通过直接调用其他模块中的函数实现的
  - 宏内核是一个大主管，把内存管理、文件管理等全部接管（函数调用）
  - 宏内核的功能块之间的耦合度太高造成修改与维护的代价太高，但因为是直接调用，效率是比较高的

Windows两种模式：用户模式和内核模式

内核模式安全组件：安全引用监控器

用户模式安全组件：

1. **登录进程 (WinLogon)** 机器总是运行着一个以 SYSTEM 为主角的登录进程(winlogon.exe)
2. 本地安全权威 (Local Security Authority, **LSA**)：用户登录时，检查有用户账户并创建访问令牌，负责审计

3. 安全账户管理员 (Security Account Manager, **SAM**)：维护用户账户数据库；在本地用户认证期间， LSA 将使用该数据库；存储口令 (hash 值)

注册表：Windows配置数据的中央数据库，表项为键key、指向特定可执行文件位置。修改注册表可以改变操作系统的行为。

域 (domain) 是共享公用用户账户数据库和安全策略的计算机集合，类似于我们前面介绍的访问控制中间结构的组。使用域来实现**一次签到以及集中式的安全管理。**要有一台服务器当域控制器（DC），其他计算机加入域，域管理员在DC上创建并管理域用户和组。域——组——用户（别名）

活动目录：对象被组织在活动目录中，可以看成一个有特定类型对象构成的树，每一种对象类型有一个特性的属性和唯一的GUID，每一种属性都有GUID。 可以通过添加新的对象类型、为已经存在的对象类型添加新属性来实现动态扩展。是Windows平台的核心组件。

活动目录服务的功能：基础网络服务、计算机管理、用户服务（管理）、资源管理、桌面配置、应用系统支撑

主角：

可以是本地用户、本地组（别名）、域用户、组、机器。

username（人可读）、SID（安全标识符，机器可读）。

每个windows机器拥有内置权威LSA，其创建的用户成为本地用户（local user）。

域控制器 (Domain controller, DC) 为一个域提供安全服务，每个域有其对应的域控制器权威 (domain controller authority,DCA) ，认证时DCA充当可信第三方，设计原则是集中式认证（口令管理）（域内），分布式服务（域间）

关于主角的作用域：

本地主角 (local principal)：本地管理，只对本地计算机可见. 如，本地用户 ；域主角 (domain principal)：域控制器上的管理员管理，对域中的所有计算机都是可见的. 如：域用户、组、别名

主体：在 Windows 中，进程和线程是主体. 进程或线程的安全凭证存放在访问令牌(access token) 中。

令牌：令牌包括一系列主角和其他安全属性。一个新的进程，在一定的限制条件，可获得其父进程的访问令牌

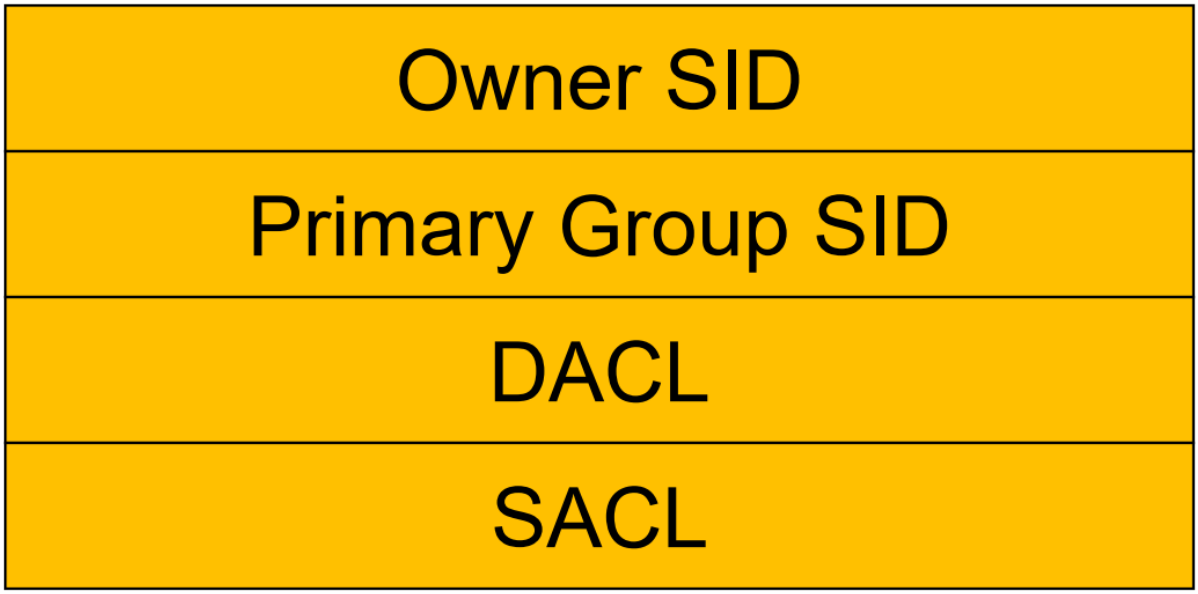
对象、客体

对象是访问操作中的被动实体：文件系统对象（文件目录）、执行对象（进程线程）、其他对象（注册表键、打印机）

安全描述符

安全对象具有安全描述符，内置对象的安全描述符由操作系统管理，私有对象的安全描述符必须有应用程序自己管理。

安全描述符包括：属主SID、主组、DACL、SACL



1. 属主SID：属主是一个主角，对象在创建时就有属主，通常有读写权利
2. 主组：为了和POSIX兼容
3. **DACL** 自主访问控制列表：定义了谁将被给予或拒绝访问。访问控制表项（ACE）的列表，ACE包括主角SID。

   访问检查：从属主得到许可；遍历DACL；当主体的令牌包含一个配对的SID时检查ACE。

   - 允许访问：当所请求的所有许可都已经获得；
   - 拒绝访问，找到否定的ACE或者到达ACE末尾
   - 否定ACE在优先于肯定的ACE，必须被放置在DACL的顶端
   - 一个ACE可以同时是肯定的和否定的
   - Empty DACL：访问一直被拒绝
   - NULL DACL：一直被允许，即如果一个对象没有DACL，那么就是说这个对象是任何人都可以拥有完全的访问权限。
4. **SACL** 系统访问控制列表：定义了审计策略，规定了哪些用户的哪些操作应该被记录到审计日志中

练习2 Windows系统中，DACL和SACL有什么区别？

- DACL（Discretionary Access Control List），其指出了允许和拒绝某用户或用户组的存取控制列表。 当一个进程需要访问安全对象，系统就会检查DACL来决定进程的访问权。如果一个对象没有DACL，那么就是说这个对象是任何人都可以拥有完全的访问权限。其中ACE的类型是肯定(允许)或否定（拒绝） 。 DACL中的ACE格式包括： 类型：肯定(允许)或否定(拒绝) 标志；Object Type； Inherited Object Type；访问权限；主角SID： ACE应用的主角。
- SACL（System Access Control List），其指出了在该对象上的一组存取方式（如，读、写、运行等）的存取控制权限细节的列表。审计规则在SACL中定义， SACL中的ACE格式包括： Type：肯定的（审计允许的许可）或否定的（审计拒绝的许可）； Trustee：一个 SID（个人，组，别名）； Mask：许可（32-bit 掩码）。一个ACE可以同时是肯定的和否定的 。

受限上下文

以受限令牌运行的进程，设计原则：最小特权

受限令牌可以从一个给定的访问令牌构建，可以为每一个进程创建受限SID，并将受限SID添加到程序所请求的资源(对象)中

DEP 数据执行保护（Data Execution Prevention）

一套软硬件技术，能够在内存上执行额外检查以帮助防止在系统上运行恶意代码

原理：将数据所在内存页标识为不可执行，当程序溢出成功转入shellcode时，程序会尝试在数据页面上执行指令，此时CPU就会抛出异常，而不是去执行恶意指令

练习7.2 在UNIX和Windows中，访问权限是为用户和组定义的。为了便于更好的安全管理，用户被放到不同的组中。当用户拥有的权限少于用户所在组的权限时，两个操作系统会如何决定访问权限呢？如何拒绝给予用户那些已经给予用户所在组的访问权限？

	UNIX	Windows
1）当用户权限少于用户所在组权限	Unix按owner,group,other顺序检查权限。如果用户为属主则按属主权限，如果用户不为属主则按属组权限。	检查用户帐号或者账号所属的组是否在访问控制列表 ACL 中。如果在，则进一 步检查访问控制项 ACE，然后根据控 ACE 中的权限来判断用户最终的权限。（即使用户没有授权也会访问所在组）。
2）拒绝给予用户那些已经给予用户所在组的访问 权限	更改权限。更改属主、属组的访问权限。	在DACL中添加否定权限的 ACE，否定 ACE 添加在许可ACE的前面。

练习7.3 讨论“中间层控制”是如何在Windows中使用的？

- 1）主角
  - 域（domain）、组（group）、别名（alias）。域是共享公用用户账户数据库和安全策略的计算机集合，用来实现一次签到和集中式管理；组是一组由与域管理器管理的SID；别名被用于实现逻辑角色。
- 2）对象
  - 活动目录。活动目录可以通过添加新的对象类型以及为已存在的对象类型添加新属性动态扩展，是Windows平台的核心组件。
- 3）访问控制规则
  - 通用访问权限。按照中间描述层的设计原则，不需要记忆文件和目录等对象的特定类型的访问许可。

数据库安全

关系数据库：表的集合

基本关系：就是实际的关系，不是派生于其他关系，拥有自己存储的数据

视图：命名的导出关系；由其他已命名的关系定义， 本身并不存储数据(这一点是和快照的区别，也需要掌握)

快照：命名的导出关系；根据已经命名的好的其他关系定义；拥有自己独立的存储的数据

查询结果：执行查询语句 (select 语句) 的返回值，可能没有名字；并不是常驻在数据库中

保证数据完整性的几种常用方法

- 约束 (最常用)：表、列级的强制规定、是防止那些无效或有问题的数据输入到表中
- 存储过程：一组为了完成特定功能的SQL 语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行它
- 函数
- 触发器：一种特殊类型的存储过程。通过事件进行触发而被执行的（而存储过程可以通过存储过程名字而被直接调用 ）

  当对某一表进行诸如Update、 Insert、 Delete 这些操作时， DBMS就会自动执行触发器所定义的SQL 语句，从而确保对数据的处理必须符合由这些SQL 语句所定义的规则。

自主访问控制：基于视图、基于角色

访问控制三要素：

1. 用户 (主体)：登陆时用户身份要得到认证
2. 对象 (客体)：表、视图、表和视图的列 (属性)
3. 行为 (操作)：包括 SELECT, UPDATE, DELETE 和 INSERT

特权：使用 GRANT 和 REVOKE 操作来管理特权

特权授予 (grant)： grant [行为] on[对象] to[用户](with grant option)

特权撤销 (revoke)： revoke[行为]on[对象]from[用户]

其中用户、对象、行为为上一小节介绍的访问控制三要素； with grant option表示可以使已经获得特权的用户授权给其他用户，是可选项

委托授权：一个主体把对客体的授权的发放和回收传递给另一个主体，使得另一个主体能够发放和回收相应客体的授权

递归回收：A回收B的授权时，有时也需要回收C的授权

否定式授权： GRANT (S, O, NO-A)禁止主体S对客体O进行A类型的访问。GRANT NO-DELETE ON emp TO bob

解决授权冲突的原则：

1. 否定优先原则： 如果一个主体既拥有在某个客体上进行某种操作的肯定式授权， 也拥有在该客体上进行同样操作的否定式授权， 那么， 否定式授权发挥作用， 肯定式授权不起作用
2. 个体优先原则： 如果某个小组及其组中的某个用户都拥有同一个客体上的同一种访问类型的授权，并且， 两者所拥有的授权是冲突的， 那么， 用户拥有的授权发挥作用， 小组拥有的授权不起作用

基于角色的自主访问控制

用户发放授权和角色发放授权的区别

用户拥有的授权：直接发放给用户的授权、用户从角色中继承的授权（间接发放授权）

角色拥有的授权：直接发放给角色的授权、角色从其它角色继承的授权

由用户执行授权操作时， 回收用户的授权引起递归回收；由角色执行授权操作时， 回收用户的授权不会引起递归回收。

基于视图的访问控制

使用特权和视图机制。

视图实际上是一个或多个表上的预定义查询，这些表称为基表。视图并不存储数据，在查询视图时实际要访问基表 。

创建视图的 SQL 语句： CREATE VIEW < 视图名 > [(< 列名 1>[， < 列名2>.....])] AS < 子查询 >

```
create [or replace] [{force|noforce}]
  view view_name
[(alias_name[,alias_name...])]
as subquery
[with {check option|read only}
  constraint constraint_name]
```

- or replace: 如果视图存在就替换它
- force: 即使基表不存在也要建立该视图
- noforce: 若基表不存在就不建立此视图，默认值
- view_name: 视图名称

- 创建视图

```
CREATE VIEW young_clerks
   AS SELECT ename, sex, deptno, phone FROM emp
    WHERE age <= 30 AND job = 'CLERKS';
```

- 授权用户访问视图

```
GRANT SELECT ON young_clerks TO tom
```

- 确保操作符合视图约束

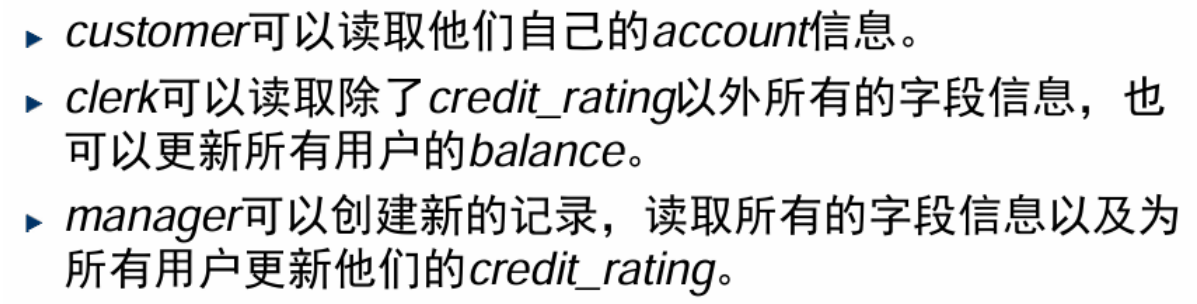
```
CREATE VIEW emp_rw
  AS SELECT * FROM emp
    WHERE ename =
      SYS_CONTEXT ('userenv', 'session_user')
       WITH CHECK OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE
 ON emp_rw TO bob, alice, tom;
```

SYS_CONTEXT: Oracle的DBMS提供的系统函数，该函数的以上调用返回当前会话用户的用户名。基于环境参数的访问控制

视图的优点:

- 简化操作：减少用户查询的复杂度
- 提高了安全性：限制某个视图只能访问基表中的部分数据
- 视图能够增强上下文依赖和数据依赖的安全策略
- 视图能够实施受控调用
- 安全的视图可以代替安全标签
- 数据可以很容易地重新分类

题目：假设有一个accounts数据库，他的记录是(customer_name,account_number, balance, credit_rating)以及以下一些用户类型： customer, clerk, manager。试定义一个访问结构，比如通过视图，来实现如下一些功能：



• CREAT VIEW customer_View AS SELECT * FROM Accounts WHERE customer_name=current_user();

• GRANT SELECT ON customer_View TO GROUP customer;

• CREAT VIEW clerk_View AS SELECT customer_name,account_number,balance FROM Accounts ;

• GRANT SELECT,UPDATE(balance) ON clerk _View TO GROUP clerk;

• CREAT VIEW manager_View AS SELECT * FROM Accounts ;

• GRANT SELECT,INSERT,UPDATE(credit_rating) ON manager _View TO GROUP manager;

强制访问控制

三元组安全标签

<等级，类别，组别>

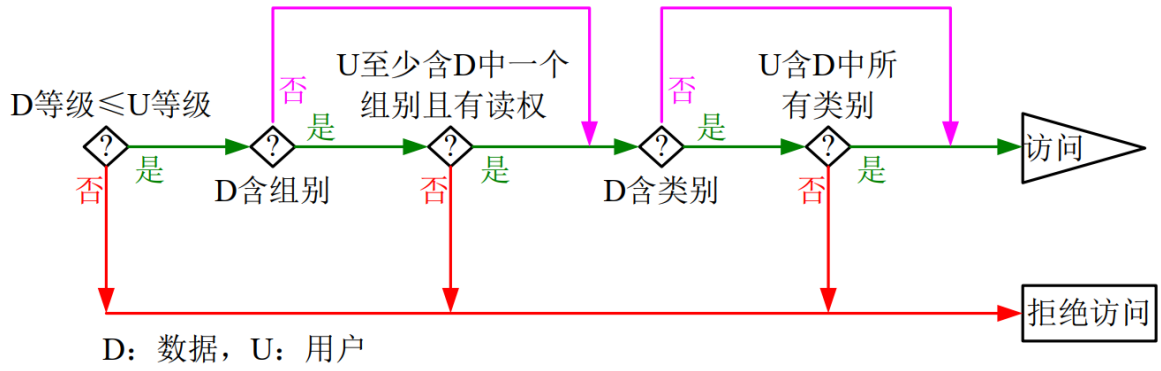
“等级”为数值型，“类别”和“组别”为集合型

有效形式：(等级::)、 (等级:类别:)、 (等级:类别:组别)、(

等级::组别)

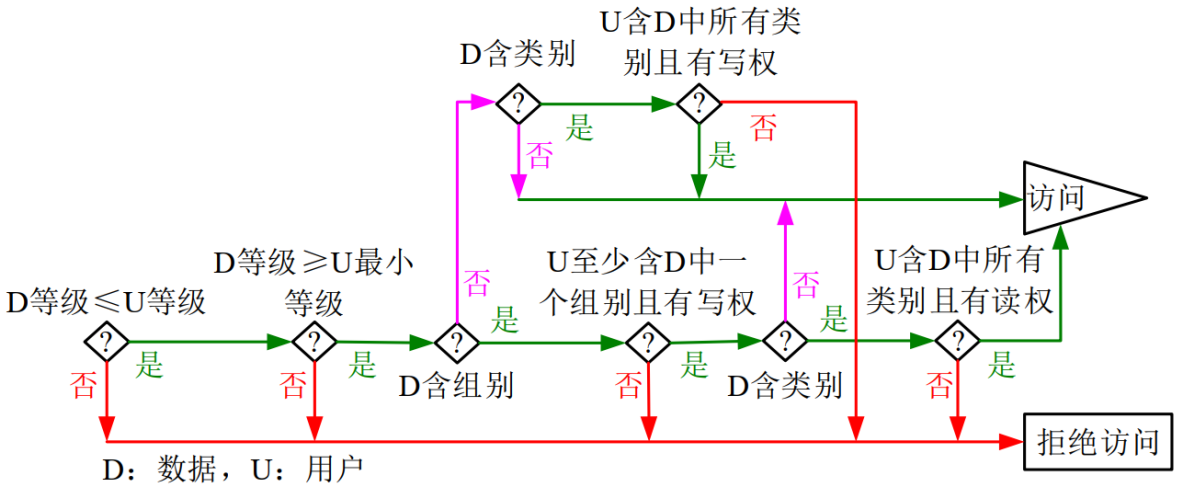
示例：(绝密::)、 (机密::)、 (秘密::)、 (非密::)、(绝密:海军:)、 (绝密:海军,空军:)、(机密:海军,空军:纽约)、(秘密:陆军:纽约,夏威夷)

基于标签“读”的访问判定



- 用户标签中的等级必须大于或等于记录标签中的等级
- 用户标签中的组别至少必须包含记录标签中的一个组别，并且用户拥有对该组别的读权限
- 用户标签的类别必须包含记录标签中的所有类别

基于标签"写"的访问判定



- 记录标签中的等级必须小于或等于用户会话标签中的等级
- 记录标签中的等级必须大于或等于用户的最小等级
- 用户会话标签中的组别至少必须包含记录标签中的一个组别，并且，用户拥有对该组别的写权限
- 用户会话标签中的类别必须包含记录标签中的所有类别
- 若记录标签不含组别，则用户必须对记录标签中的所有类别拥有写权限；若记录标签含有组别，则用户必须对记录标签中的所有类别拥有读权限

例题

练习3 在OLS机制中，设ca、cb、cc是三个类别，ga、gb、gc是三个组别，而且ga是gb和gc的父组。用户U的安全等级取值范围是100~200，他对ca拥有读和写权限，对cb拥有读权限，对cc无访问权限，对ga拥有读和写权限。已知若干安全标签如下：

- 用户U的默认会话标签：Id=(150:ca,cb:ga)
- 用户U的默认记录标签：Ir=(100:ca,cb:gb)
- 记录1的安全标签：Ir1=(150)
- 记录2的安全标签：Ir2=(100)
- 记录3的安全标签：Ir3=(150:ca)
- 记录4的安全标签：Ir4=(200:cb)
- 记录5的安全标签：Ir5=(150:cc)
- 记录6的安全标签：Ir6=(100:ca)
- 记录7的安全标签：Ir7=(100:cb)
- 记录8的安全标签：Ir8=(150:cb:ga)
- 记录9的安全标签：Ir9=(150:cb:gb)
- 记录10的安全标签：Ir10=(100:cb,cc:gc)

假如系统设置了LABEL_DEFAULT选项，并且未临时改变过会话标签，用户U正在使用系统，请分析并回答以下问题：

(1) 用户U对哪些记录拥有读权限？对哪些记录拥有写权限？

(2) 用户U插入了一个新记录但未指定其标签，该记录的标签值等于什么？

可读：1，2，3，6，7，8，9
可写：1，2，3，8，9

插入新纪录的标签值等于默认记录标签值：（100：ca，cb：gb）

统计数据库的推理方法：

借助求和结果进行推理

借助记录个数进行推理

借助平均值进行推理

借助中位数进行推理

借助智能填充进行推理

借助线性特性进行推理

跟踪攻击

差分隐私

数据安全

三种数据备份方式：

• 完全备份：定期进行包括系统和数据的完全备份

• 增量备份：只备份上次数据库完全备份后发生更改的部分数据库

• 差分备份：每次备份的数据只是相当于上一次备份后增加的和修改过的数据

BLP模型——针对保密性

Tip：BLP模型中读/写就是"读"，添加/写就是"写"。

在 BLP 模型中， 安全级别就是**保密性**级别

- 符号
  - 主体集合S
  - 客体集合O
  - 访问操作集合A={execute, read, append, write}
  - 具有偏序≤的安全级别集合L (例如级别可以是绝密、机密、秘密、内部、公开等)
- 用系统状态来检查系统安全，所以模型的状态集必须包括当前所有的许可和当前所有主体访问客体的实例。这导致了一个相当复杂的状态集B×M×F

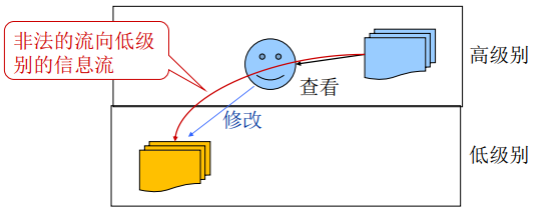
- B = P(S×O×A)是当前访问的集合
  - 元素b∈B是一个三元组（s, o, a），表示主体s当前正在客体o上执行操作a，例如(Alice, fun.com, read)
  - B = P(S×O×A)，例如{(Alice, fun.com, read), (Bob, fun.com, write), …}
- M表示所有获得访问许可的矩阵的集合，每个访问许可矩阵$m = (m_{so})$ s∈S, o∈O
  - M = {$\{read, write\}_{Alice, fun.com}$, $\{read, write, execute\}_{Bob, fun.com}$}

- F⊆ $L^S \times L^S \times L^O$是安全级别分配的集合。元素$f$∈F是一个三元组($f_S$, f_C, f_o)，其中
  - f_S: $S \rightarrow L$给出了每个主体可以拥有的最高安全级别
  - f_C: $S \rightarrow L$给出了每个主体当前的安全级别
  - f_o: $O \rightarrow L$给出了所有客体的安全级别
- 一个主体的当前级别不能高于他的最高级别，因此$f_C \le f_S$，读作“f_S控制f_C”或“f_S支配f_C”
- 高安全级别有时称为主体的许可(clearance)，其它文献只用许可来表示用户的安全级别

- 状态集合V=B×M×F
- 一个独立的状态由三元组(b, m, f)决定
- 定义状态集是BLP的主要问题，不需要描述输入、输出或状态迁移的精确结构来给出BLP的安全特性

简单安全性ss-property：如果对于每一个元素(s, o, a) ∈ b，访问操作 a 是读/写，则主体 s 的安全级别控制客体 o 的安全级别，即 $f_s(s) \geqslant f_o(o)$，那么那么状态 (b, m, f) 满足简单安全性。

- ss-property不足以防止一个低级别的主体读一个高级别客体的内容
  - 可以创建一个高级别的特洛伊木马，读高级别的客体并将它拷贝（写它的内容）到一个低级别的客体
- 需要一个策略来控制写访问



星特性：简单安全性只实现了**无向上读**，星特性在此基础上增加了**无向下写**的策略。星特性的内容由两条陈述组成，注意的是二者缺一不可，如果遇到简答题，两条都需要答上。

1. 如果对于每一个元素(s, o, a) ∈ b，访问操作 a 是添加或写，则主体 s 的当前级别受客体 o 的当前级别控制，即 $f_c(s) \leqslant f_o(o)$，那么状态 (b, m, f) 满足星特性；
2. 如果存在一个元素，有 (s, o, a) ∈ b，访问操作 a 是添加或写，那么对于所有客体o'， b = (s, o', a') ∈ B， a' 是读或写，则必须有 $f_o(o') \leqslant f_o(o)$。

> 信息只能向上流动，安全等级高的信息无法流动到下层

简单地在低级别阻止主体修改客体将产生一个问题：高级别的主体不能发消息给低级别的主体

有两种方法可以避开这个限制：
- 第一种方法：临时地降低高级别主体的级别，这是引入当前安全级别f_C的原因
- 第二种方法：确定一组允许违背*-property的主体，这些主体称为可信主体(trusted subjects)

自主安全性ds-property：自主安全性对应 DAC，也就是对应访问控制矩阵 (或者说 ACL)。所以不难记忆其内容：

如果对于每一个元素，均有 (s, o, a) ∈ b， a ∈ m_{so}，那么状态 (b, m, f) 满足自主安全性。

基本安全定理：如果系统中所有的状态迁移都是安全的，并且系统的初始状态也是安全的，那么不管输入情况如何，其后的每一个状态也都是安全的。

基本安全定理是由状态机模型理论产生的。首先定义状态安全和状态转移安全的定义：

• 如果状态 (b, m, f) 满足简单安全性、星特性、自主安全性，那么称状态 (b, m, f)是安全的

• 如果状态 v1 = (b1, m1,f1) 和状态 v2 = (b2, m2, f2) 都是安全的，那么称从状态 v1到状态 v2 的迁移是安全的

状态迁移安全的形式化描述：

如果状态(b, m, f)满足ss-, *-, ds-property策略，那么称状态(b, m, f)是安全的

如果状态$v_1 = (b_1, m_1, f_1)$和状态$v_2 = (b_2, m_2, f_2)$都是安全的，那么称从状态v_1到状态v_2的迁移是安全的

• 每个 (s, o, a) ∈ $b_2 \backslash b_1$ 满足 f_2 的简单安全性 (星特性、自主安全性)， ($b_2 \backslash b_1$ 表示集合 b_2 和 b_1 的差集)

• 如果 (s, o, a) ∈ b_1 不满足 f_2 的简单安全性 (星特性、自主安全性)，那么(s, o, a) ∉ b2.

练习1 用基本访问模式术语alter(修改)和observe(查看)描述*-property

- 如果对于每一个元素，均有(s,o,a)∈b，访问操作a是添加或写，主体s的当前级别受客体o的当前级别控制，即$f_C(s) \le f_O(o)$，那么状态(b, m, f)满足*-property
- 此外，如果存在一个元素，有(s,o,a)∈b，访问操作a是添加或写，那么对于所有的客体o'，(s,o',a')∈b，a'是读或写，必须有$f_O(o') \le f_O(o)$

- 如果主体可以对客体进行alter，那么主体s的当前级别受客体o的当前级别控制，也就是$f_C(s) \le f_O(o)$。
- 如果主体可以alter客体o，可以observe客体o'，那么$f_O(o') \le f_O(o)$。

练习2 在一个控制对病历和处方访问的医疗信息系统中：

- 医生可以读写病历和处方；
- 护士只能读写处方，但是不应当知道病历的内容。

你如何在格模型中得到这个策略，阻止信息从病历流到处方？依你看，哪一种安全模型最适合这种策略？为什么？

将医生作为可信主体，最高安全等级为{High，{病历，处方}}，当前安全等级为{Low，{病历，处方}}，护士的安全等级为{Low，{处方}}，病历的安全等级为{High，{病历}}，处方的安全等级为{Low，{处方}}。这样是指可以实现护士对处方的读写，对病历的不可见；医生基于当前安全等级可以对处方进行读写操作，基于最高安全等级可以对病历进行读写。

BLP模型只考虑主体在查看或改变一个客体时发生的信息流动，可以实现无向上读，无向下写。因此可以用来实现该权限控制策略，能够防止信息从高安全级别流向低安全级别。

Biba模型——针对完整性

Biba 模型中，安全级别是完整性级别 。用于衡量完整性等级的术语是“可信度” . 高级别的客体比低级别的客体更“可信”

因此， 对应的安全策略：**无向上写**（禁止“干净的”高级别实体被“脏的”低级别实体污染 ）

注意这里和 BLP 模型的区别. 其中这里只讨论写，因为这里关心的不是保密性而是完整性，只有写操作才能影响完整性，而读操作并不影响完整性，所以其安全策略中并不关心读的问题。

- 完整性级别格$(L, \leq)$的元素标注主体和客体，具体由函数$f_S: S \to L$和$f_O: O \to L$给定
- 在完整性格中，信息只能向下流动；只关心由访问操作直接引发的信息流
- Biba模型：类似于BLP的状态模型；但没有唯一的高级别的完整性策略

静态完整性级别

简单完整性:

无向上写：如果主体 s 可以修改客体 o，则 $f_S(s) \geqslant f_O(o)$

星特性:

如果主体 s 可以读客体 o，那么仅当 $f_O(o) \geqslant f_O(o')$ 时， s 可以写另一客体 o'。因为要避免信息从高等级流向低等级。

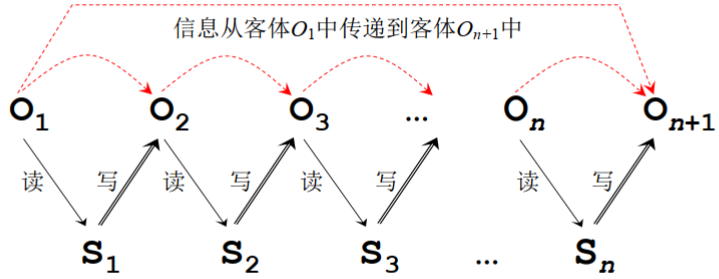
动态完整性级别：如果一个实体已经接触了低级别的信息， 低水印性策略自动调整实体的完整性级别（如同Chinese Wall模型）

1. 主体低水印性：主体 s 可以读 (查看)任何完整性级别上的客体 o， 主体的新的完整性级别是 $\inf\{f_S(s), f_O(o)\}$——主体降级
2. 客体低水印性：主体 s 可以修改任何完整性级别上的客体 o， 客体的新的完整性级别是 $\inf\{f_S(s), f_O(o)\}$ ——客体降级

   其中， $f_S(s)$和$f_O(o)$是操作前的完整性级别，$\inf(f_S(s), f_O(o))$是$f_S(s)$和$f_O(o)$的最好的低限。防止间接的非法修改行为的发生。

信息传递路径

一个信息传递路径是一个客体序列$O_1, O_2, \ldots, O_{n+1}$和一个对应的主体序列$S_1, S_2, \ldots, S_n$，其中，对于所有的$i$（$1 \le i \le n$），有$S_i \underline{r} O_i$和$S_i \underline{w} O_{i+1}$



在Biba低水印性策略的控制下，如果系统中存在一个从客体O_1到客体O_{n+1}的信息传递路径，那么，对于任意的$k(1 \le k \le n)$，必有$f_O(O_k) \ge f_O(O_{k+1})$

调用性——只能向下

主体可以调用另一个主体来访问客体，这是向在中间层上表达的访问控制迈出的第一步。但是要确保完整性策略

调用性：一个“脏的”主体s_1不能通过调用主体s_2来间接访问一个“干净的”的客体：主体s_1可以调用主体s_2，仅当$f_S(s_1) \ge f_S(s_2)$

- 主体只允许在较低级别上调用工具，否则一个脏的主体可使用一个干净的工具去访问并可能污染一个干净的客体

环属性——只能向上

- 环属性：一个“脏的”主体s_1可以通过调用一个“干净的”工具s_2来间接访问一个“干净的”的客体：主体s_1可以读任何完整性级别上的客体o，它只能在$f_S(s_1) \ge f_O(o)$时修改客体o，仅当$f_S(s_1) \le f_S(s_2)$时可以调用主体s_2
- “环属性”和“调用性”是不一致的，必须根据应用来决定哪种特性更合适
- 操作系统通过“保护环”来进行完整性保护的

Chinese Wall模型

为了理解中国墙模型，我们可以先了解一下其背景.

它模拟了咨询公司的访问规则：咨询公司的分析员必须保证他们与不同客户的交易不会引起利益冲突，即一定不能有引起利益冲突的信息流。

主体：分析员 (集合用 S 表示)；客体：信息条目 (集合用 O 表示)；公司集合： C.

安全标签： (利益冲突类, 公司数据集)，即 (x(o), y(o))

- 利益冲突类：相互竞争的公司. 函数 x : O → P (C) 给出了客体 o 的利益冲突类
- 公司数据集：有关同一公司的所有客体的集合. 函数 y : O → C 给出了每个客体的公司数据集

简单安全性

仅当客体请求属于以下情形时访问才被允许：

• 用户已经拥有的一个公司数据集

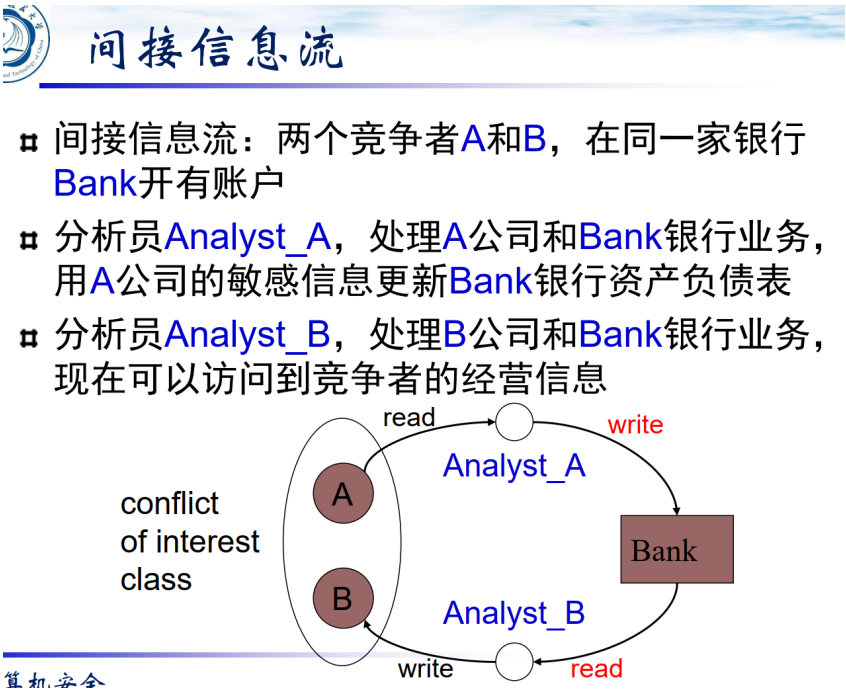
• 一个完全不同的利益冲突类

用矩阵 N = $(N_{so})_{s\in S,o\in O}$ 记录访问情况， (N_{so}) = true 表示主体 s 访问过客体 o

我们给出简单安全性的形式化描述：

主体 s 允许访问客体 o，仅当对于所有 $(N_{so'})$ = true 的客体 o' 有 y(o) = y(o') 或y(o) ∉ x(o')

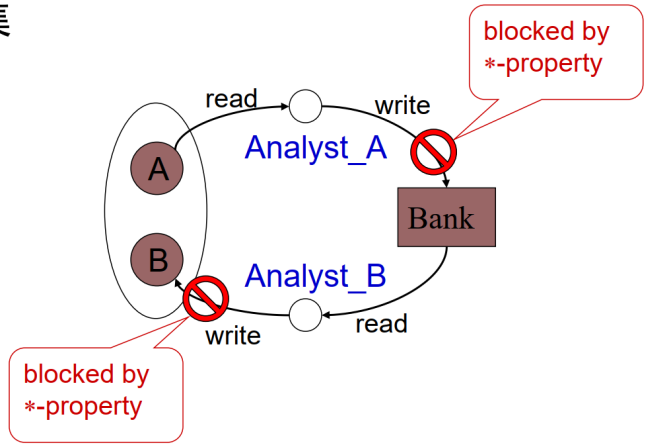
星特性



此时Analyst_A又能读A，又能写Bank，造成冲突。

阻塞间接信息流

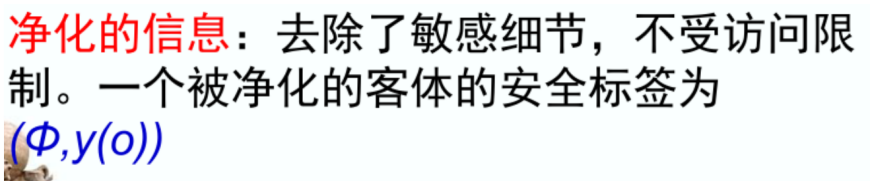
- * - Property阻止未净化的信息流出公司的数据集



***-property定义**:

主体 s 被允许对客体 o 进行写访问，仅当 s 对 y(o') ≠ y(o) 和 x(o') ≠ ϕ 的客体 o'没有读访问权。访问权限动态变化，每次访问操作，主体的访问权限都要重新赋值。

仅当没有其他的位于不同公司数据集，且包含非净化信息的客体不能被阅读时，对一个客体的写访问才是允许的。



信息流模型

信息流控制策略：描述信息流动的合法路径的安全策略

保密性：信息流不流向未授权用户

完整性：信息流流向的进程可信度不高于数据可信度（防止高可信度进程被污染，只能高流向低）

信息从客体x->客体y，记为y<-x，x是来源，y是目标

强信息流(显式信息流):

• x 的值直接影响 y 的值，例如赋值

• 形式化描述： y = f(x)

弱信息流 (隐式信息流):

• 信息从 x 流向 y，但 x 的值不直接影响 y 的值

• 弱信息流不是因为使用 x 值进行了赋值，而是语句中基于 x 值的控制流产生了弱信息流。

不存在显式xy函数，x的值不直接影响y，弱信息流中x的值一般作为条件，比如：if(f1(x1)==0) then y=f2(a) else y=f3(b)，其中ab是客体

隐式信息流的信息量计算

x 到 y 的信息流用给定 y 的值后的 x 的信息量平均值的变化 (条件熵) 来度量

$$H_y(x) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(x_i|y_j)$$

考虑赋值“IF *x=0* THEN *y:=1*”

设*x*和*y*是二进制变量，*y*的初始值设为0，*x*的两个值很可能相等，有

$p(0|0)=p(1|1)=0, p(1|0)=p(0|1)=1$,因此$H_y(x)=0$

事实上，在执行赋值后观察*y*，可知道*x*的确切值，*x*中的所有信息流入了*y*

如果*x*为0, 1, 2的概率相等，将得到

q(0)=2/3, q(1)=1/3

$p(0|0)=p(1|1)=p(2|1)=0, p(1|0)=p(2|0)=\frac{1}{2}, p(0|1)=1$,因此$H_y(x)=\frac{2}{3}$

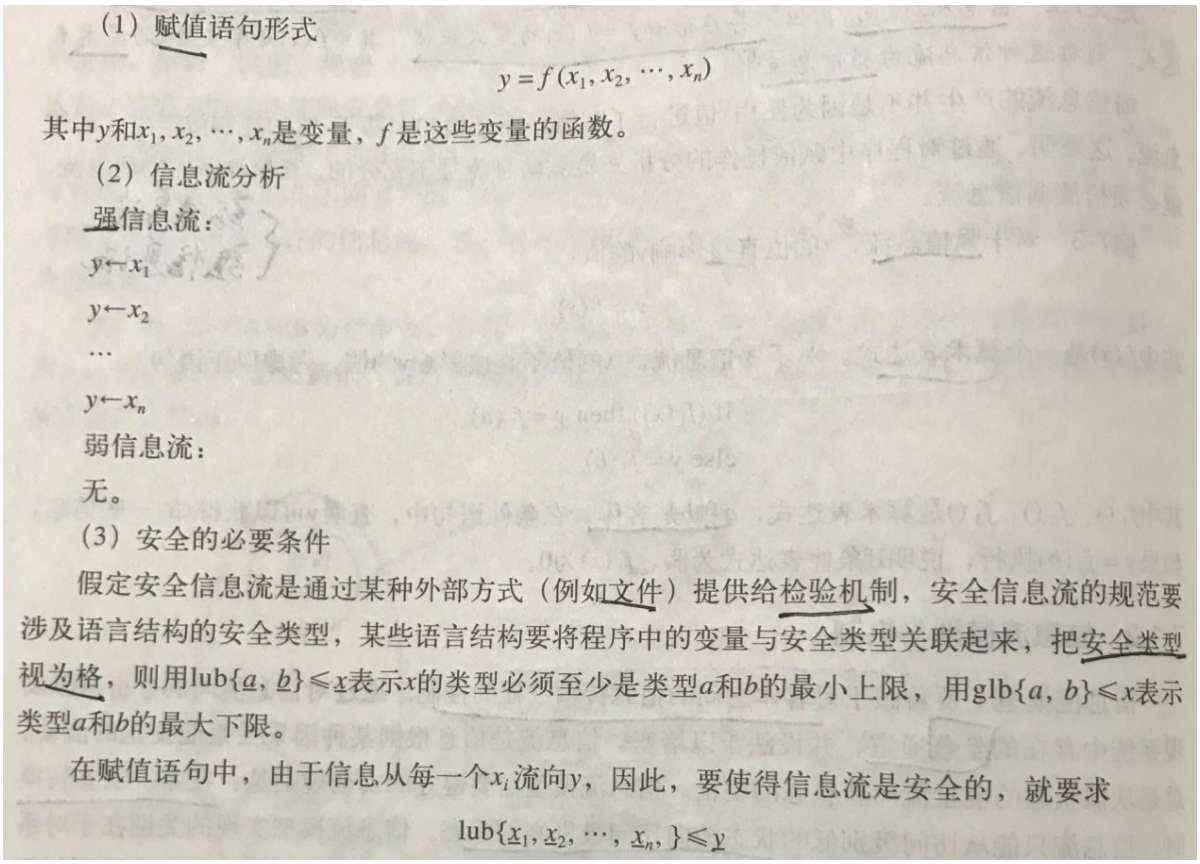
将程序中的变量与安全类型关联起来，把安全类型视为格

lub：最小上限（least upper bound）

glb：最大下限（greatest lower bound）

lub{a, b}≤ x，表示x的类型必须至少是类型a和b的最小上限

glb{a, b}≤ x，表示x的类型必须至少是类型a和b的最大下限



（1）赋值语句形式

$$y = f(x_1, x_2, \cdots, x_n)$$

其中y和$x_1, x_2, \cdots, x_n$是变量，f是这些变量的函数。

（2）信息流分析

强信息流：

$y \leftarrow x_1$

$y \leftarrow x_2$

…

$y \leftarrow x_n$

弱信息流：

无。

（3）安全的必要条件

假定安全信息流是通过某种外部方式（例如文件）提供给检验机制，安全信息流的规范要涉及语言结构的安全类型，某些语言结构要将程序中的变量与安全类型关联起来，把安全类型视为格，则用lub{a, b}≤x表示x的类型必须至少是类型a和b的最小上限，用glb{a, b}≤x表示类型a和b的最大下限。

在赋值语句中，由于信息从每一个x_i流向y，因此，要使得信息流是安全的，就要求

$$\text{lub}\{\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n, \} \leqslant \underline{y}$$

例7-4 考虑赋值语句的例子。赋值语句如下：

```
a=b+c-d;
```

试进行信息流分析及说明安全的条件。

强信息流：

$a \leftarrow b$

$a \leftarrow c$

$a \leftarrow d$

弱信息流：

无。

安全的必要条件：

使信息流安全的条件为$\text{lub}\{b, c, d\} \leqslant a$。

2. 复合语句

（1）复合语句形式

begin

S_1;

...

S_n;

end;

其中，每一个S_i是一条语句，$1 \leqslant i \leqslant n$。

（2）信息流分析

信息流包括语句S_i中的所有信息流，$1 \leqslant i \leqslant n$。

（3）安全的必要条件

如果每条语句的信息流都是安全的，则复合语句中的信息流就是安全的，因此复合语句中信息流安全的必要条件是：

S_1是安全的；

...

S_n是安全的。

例7-5 考虑复合语句的例子。复合语句如下：

```
begin
      x = a + b;
      y = x + z;
end;
```

试进行信息流分析及说明安全的条件。

强信息流：

$$x \leftarrow a, x \leftarrow b, y \leftarrow x, y \leftarrow z$$

弱信息流：

无。

安全的必要条件：

使信息流安全要求对S_1有$\text{lub}\{a, b\} \leqslant x$以及对$S_2$有$\text{lub}\{x, z\} \leqslant y$，所以安全信息流的必要条件是$\text{lub}\{a, b\} \leqslant x$以及$\text{lub}\{x, z\} \leqslant y$。

(1) 条件语句形式

if $f(x_1, x_2, \cdots, x_n)$then

S_1;

else

S_2;

end;

其中$x_1, x_2, \cdots, x_n$是变量，f是关于这些变量的（布尔）函数，y是复合语句S_1或S_2的赋值语句的一个赋值对象。

(2) 信息流分析

信息流包括复合语句S_1和S_2中的所有信息流。

强信息流：

无。

强信息流：

$$y \leftarrow x_1, y \leftarrow x_2, \cdots, y \leftarrow x_n$$

(3) 安全的必要条件

根据f的值，S_1和S_2中有一个要被执行，所以S_1和S_2都必须是安全的。如前面所讨论的，选择S_1或者S_2将泄露变量$x_1, x_2, \cdots, x_n$的值，所以信息必须能够从这些变量流向S_1和S_2中的任意赋值目标。此条件满足当且仅当赋值目标的最低级类型支配变量$x_1, x_2, \cdots, x_n$的最高级类型。这样信息流安全的必要条件就是：

S_1是安全的；

S_2是安全的；

$\text{lub}\{\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_n\} \leqslant \text{glb}\{\underline{y}\}$，其中$y$是复合语句$S_1$或$S_2$中赋值语句的一个赋值对象。若语句$S_2$是空语句，它是平凡安全的而且没有任何的赋值。

例7-6 条件语句的例子。条件语句如下：

```
if  x+y≤z then
    a = b;
else
    c = b;
end;
```

试进行信息流分析及说明安全的条件。

强信息流：

$$a \leftarrow b, c \leftarrow b$$

弱信息流：

$$a \leftarrow x, a \leftarrow y, a \leftarrow z$$

$$c \leftarrow x, c \leftarrow y, c \leftarrow z$$

安全的必要条件：

对于S_1，要求$\underline{b} \leqslant \underline{a}$，对于$S_2$，要求$\underline{b} \leqslant \underline{c}$，具体执行哪一条语句由$x$、$y$和$z$的值决定，因此，信息从$x$、$y$和$z$流向$a$和$c$，要求$\text{lub}\{\underline{x}, \underline{y}, \underline{z}\} \leqslant \text{glb}\{\underline{a}, \underline{c}\}$。

隐信道

信息沿着三种类型的信道流动：

隐蔽通道：一个不受安全机制控制的信息流，信息通过一个隐蔽通道传输是可能的。一个状态变量：一次传递一个比特信息

存储通道：如果一个进程直接或间接写一个存储单元，另一个进程直接或间接读该存储单元

定时通道：如果一个进程通过调解它对系统资源的使用，适应到另一个进程的真实响应时间，实现一个进程向另一个进程传递信息

安全评估

安全评估框架（评估对象、 评估目标、 评估方法）

评估对象TOE：产品、系统

> ITSEC适合对系统的评估

评估目标：（层层递进）

- 评估(Evaluation)：评定一个产品是否具有它所声明的安全属性
- 证明(Certification)：评定一个（已评估的）产品是否适合给定的应用场合
- 鉴定(Accreditation)：确定一个（已证明的）产品是否可以在给定场合里应用

评估方法：

- 评估不应该漏过问题；不同评估方法对同一产品的评估结果应该相同；
  - 面向产品的：检查和测试产品；更加适合查找问题
  - 面向过程的：检查文档和产品开发的过程；代价小，而且容易达到可重复的结果
- 可重复性(Repeatability)和可再现性(Reproducibility)是评估方法必须满足的要求

橘皮书**TCSEC**：第一个评估安全产品（OS）的指导方针。将计算机安全从高到低A、B、C、D四类七个级别，共27条准则，安全级别定义是递增的

D：最小保护级

> 是为那些经过评估，但不满足较高评估等级要求的系统设计的，只具有一个级别。该类是指不符合要求的那些系统，因此，这种系统不能在多用户环境下处理敏感信息。

C：自主保护级（C1自主安全保护级，C2控制访问保护级）

> 具有一定的保护能力，采用的措施是自主访问控制和审计跟踪。一般只适用于具有一定等级的多用户环境，具有对主体责任及其动作审计的能力
>
> C1级TCB通过隔离用户与数据，使用户具备自主安全保护的能力；具有多种形式的控制能力，对用户实施访问控制。适用于处理同一敏感级别数据的多用户环境
>
> C2级计算机系统比C1级具有更细粒度的自主访问控制，通过注册过程控制、 审计安全相关事件以及资源隔离，使单个用户为其行为负责。一般C2级被认为**对商业应用是最合理的安全级别**，大多数厂商都提供经C2评估过的操作系统或数据库管理系统的版本

B：强制保护级（基于标签）（B1标记安全保护级，B2结构化保护级，B3安全域级）

> 主要要求是TCB应维护完整的安全标记，并在此基础上执行一系列强制访问控制规则。B类系统中的主要数据结构必须携带敏感标记

> B1级系统要求具有C2级系统的所有特性在此基础上，还应提供安全策略模型的非形式化描述、 数据标记以及命名主体和客体的强制访问控制，并消除测试中发现的所有缺陷
>
> 在B2级系统中， TCB建立于一个明确定义并文档化形式化的安全策略模型之上。要求将B1级系统中建立的自主和强制访问控制扩展到所有的主体与客体，包括对物理设备使用的控制。对于登录和初始认证，必须有可信路径。在此基础上，应对隐蔽信道进行分析。B2级系统应具备相当的抗渗透能力
>
> 在B3级系统中， TCB必须满足访问监控器需求：访问监控器对所有主体对客体的访问进行仲裁、访问监控器本身是抗篡改的、访问监控器足够小从而能够分析和测试。

A：验证保护级（验证设计级A1）

> 使用形式化的安全验证方法，应提供丰富的文档信息

安全级别	定　义
A1	验证设计（Verified Design）
B3	安全域（Security Domains）
B2	结构化保护（Structural Protection）
B1	标记安全保护（Labeled Security Protection）
C2	受控的存取保护（Controlled Access Protection）
C1	自主安全保护（Discretionary Security Protection）
D	最小保护（Minimal Protection）

ITSEC：应用于安全产品和安全系统，打破功能性和保证性的关系，方便兼容新的安全需要，针对完整性。应用于安全产品的标准也被用于安全系统。评估的发起人决定操作性需求和可能的威胁

评估对象（TOE）的安全目标进一步取决于法律和其他规则，这些形成了所需的安全功能和评估级别。安全目标（ST）定义了评估相关的TOE的所有方面，TOE的安全功能可以独立定义，也可以引用预定义的功能类，E0-E6七个评估级别表示了安全功能执行的正确性的保证级别

以超越TCSEC为目的，将安全概念分为功能和功能评估：

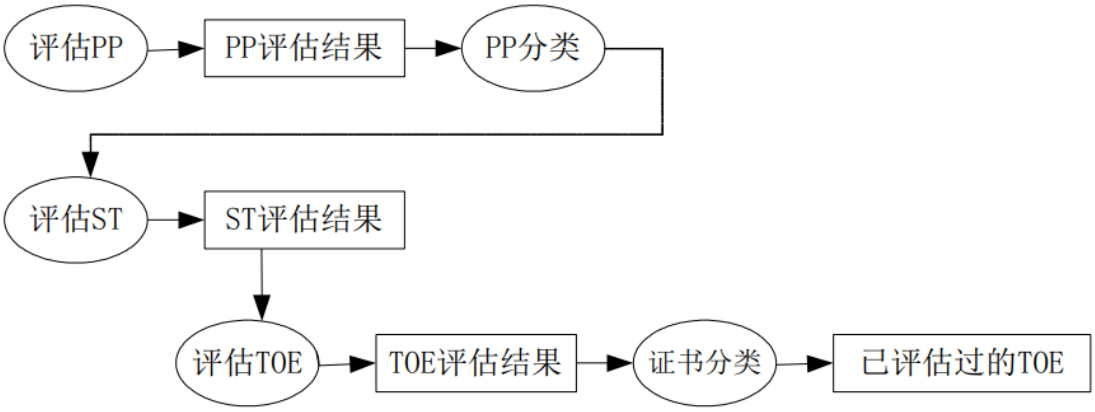
- 功能标准：F1-F10共10级，1-7级对应TCSEC的D到A，6-10还对应：

  F6：数据和程序的完整性，F7：系统可用性，F8：数据通信完整性，F9：数据通信保密性，F10：包括保密性的完整性的网络安全

- 评估准则：七级，在评估准则中首次提出CIA

  E0：最低，E1：测试，E2：配置控制和可控的分配，E3：能访问详细设计和源码，E4：详细的脆弱性分析，E5：设计与源码明显对应，E6：设计与源码在形式上一致

CC（通用准则）



使用对象：最终用户、开发者、测评者

对产品和系统(评估对象TOE)进行安全评估的标准

CC框架下的评估类型:

- 保护框架（PP）：适合特定用户需求的安全需求（包括一个EAL）的（可重用的）集合，用户应该开发自己的PP来捕获自己的典型安全需求
- 安全目标（ST）：表示一个特定的TOE的安全需求，例如对一个PP的引用，是任何安全评估的基本要素
- 评估保证级别（EAL）：定义了一个TOE的开发者和安全评估者的责任，有七个递增的级别：EAL1~EAL7

EAL1：功能测试

EAL2：结构测试

EAL3：系统测试和检查

EAL4：系统设计、测试和复查

EAL5：半形式化设计和测试

EAL6：半形式化验证的设计和测试

EAL7：形式化验证的设计和测试

区别与联系： ITSEC吸取了TCSEC的经验教训, 与TCSEC相比, 打破了功能性和保证性之间的联系; CC则是国际标准化组织统一现有多种准则的努力结果, 放弃了ITSEC采用的对功能类和保证等级的严格分离, 主要思想和框架取自ITSEC和FC, 仍然在不断发展和更新。三者都有七个安全等级，是不断吸取经验教训进行改善和标准化的结果。

CC	TCSEC	ITSEC
-	D	E0
EAL1	-	-
EAL2	C1	E1

CC	TCSEC	ITSEC
EAL3	C2	E2
EAL4	B1	E3
EAL5	B2	E4
EAL6	B3	E5
EAL7	A1	E6

- 练习2参照TCSEC标准，Windows 11和Linux 6.x的安全级别分别如何？试简述理由。

  都是C2或B1安全等级 （linux系统安全级别一般不会低于Windows系统，言之有理即可）

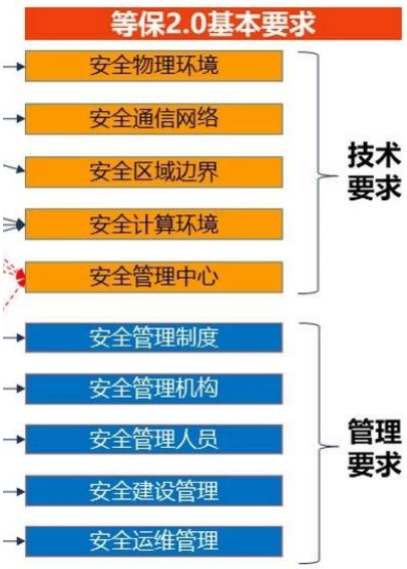
  C2
  - C2级计算机系统比C1级具有更细粒度的自主访问控制
  - C2级通过注册过程控制、审计安全相关事件以及资源隔离，使单个用户为其行为负责
  - C2被认为对商业应用是最合理的安全级别；大多数厂商都提供经C2评估过的操作系统或数据库管 理系统的版本

  B1
  - B1级系统要求具有C2级系统的所有特性
  - 在此基础上，还应提供安全策略模型的非形式化描述、数据标记以及命名主体和客体的强制访问控 制并消除测试中发现的所有缺陷

网络安全等级保护

等级保护的内涵：根据信息系统应用业务重要程度及其实际安全需求，分等级、分类、分阶段实施保护，保障信息安全和系统安全正常运行，维护国家利益、公共利益和社会稳定以及公民、法人和其他组织的合法权益。

等保 2.0 的十大安全类：



等级保护制度的主要内容(从信息、信息系统、安全产品、安全服务资质、安全事件等方面)

1. 对国家秘密信息，公民、法人和其他组织的专有信息、公开信息分类分等级进行管理和保护
2. 对信息系统按业务安全应用域、区，实行分等级保护
3. 对系统中使用的信息安全产品实行分等级许可管理
4. 对等级系统的安全服务资质实行分等级许可管理
5. 对信息系统中发生的信息安全事件分等级响应、处置

等级保护的主要工作：定级、备案、建设/整改、定期等级测评、定期监督检查

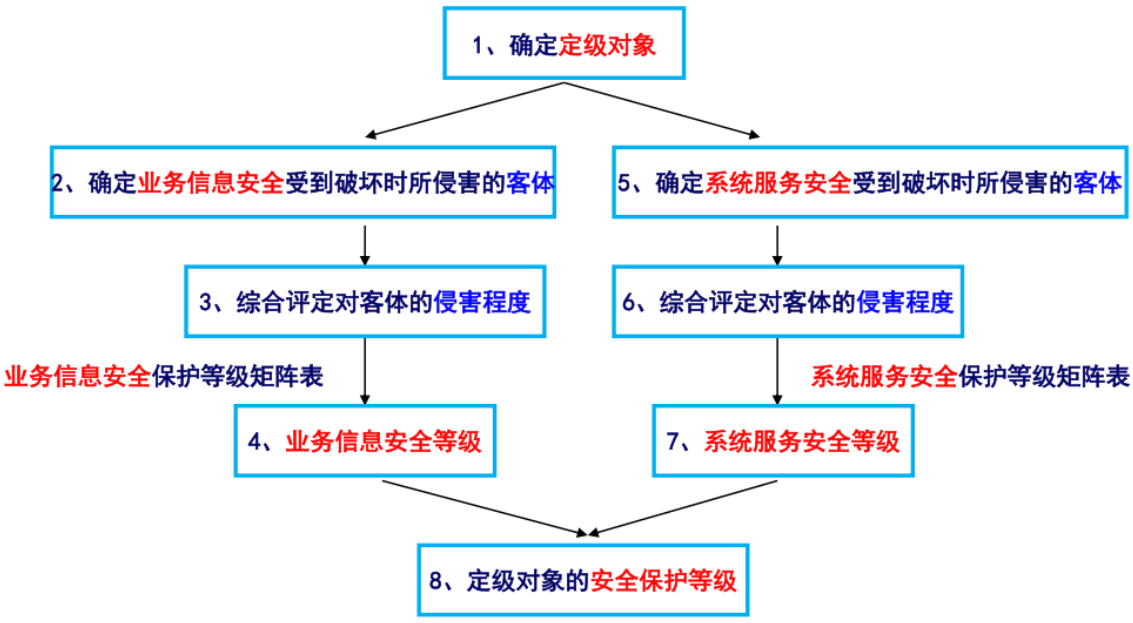
等级保护对象的定级方法：业务信息/系统服务/受侵害客体/侵害程度

- 业务信息和系统服务：相对应，分别从静态和动态体现了信息系统的重要作用
- 受侵害客体：国家安全；社会秩序、公共利益；公民、法人和其他组织的合法权益
- 侵害程度：一般损害、严重损害、特别严重损害

  信息系统的安全保护等级由业务信息安全保护等级和系统服务安全等级较高者决定

业务信息安全被破坏时所侵害的客体	对相应客体的侵害程度		
	一般损害	严重损害	特别严重损害
公民、法人和其他组织的合法权益	第一级	第二级	第二级
社会秩序、公共利益	第二级	第三级	第四级
国家安全	第三级	第四级	第五级

系统服务安全被破坏时所侵害的客体	对相应客体的侵害程度		
	一般损害	严重损害	特别严重损害
公民、法人和其他组织的合法权益	第一级	第二级	第二级
社会秩序、公共利益	第二级	第三级	第四级
国家安全	第三级	第四级	第五级



等级测评流程：测评准备、方案编制、现场测评、报告编制

练习1 业务信息安全和系统服务安全有什么区别？

- 业务信息安全和系统服务安全是定级对象的安全保护等级时的两个不同的方面，保护侧重点不同。
- 业务信息安全：保护数据在存储、传输、处理过程中不被泄漏，破坏和免受未授权的修改的信息安全类要求(简记为S，Security)。
- 系统服务安全：保护系统连续正常运行，免受对系统的未授权修改，破坏而导致系统不可用的服务保证类要求(简记为A，Available)。
- “业务信息”与“系统服务”相对应，分别从静态和动态两个方面体现信息系统的重要作用。

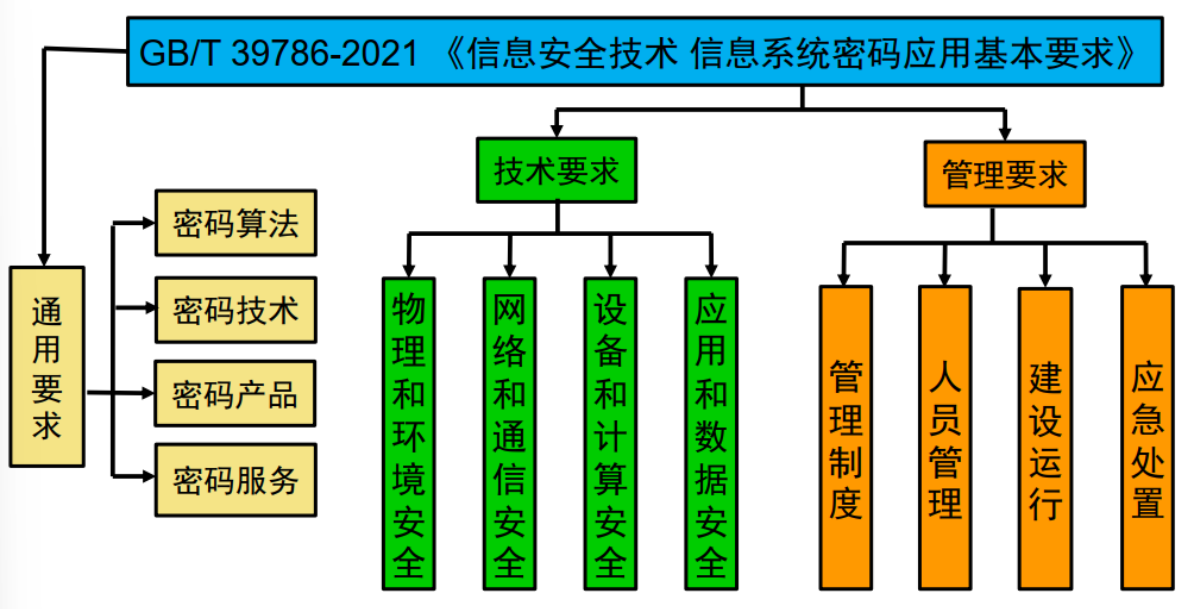
密码应用安全性评估

密码应用安全性评估的含义（合规性、正确性、有效性）

对网络与信息系统使用商用密码技术、产品和服务的合规性、正确性、有效性进行检测分析和评估验证

合规性	使用的密码算法、密码协议、密钥管理符合国家法律法规和标准规定，密码产品或服务经过国家密码管理局核准和认证机构认证合格。
正确性	密码算法、密码协议、密钥管理、密码产品和服务使用正确，即按密码相关的国家和行业标准进行正确设计和实现，密码产品和服务的部署、使用正确。
有效性	密码保障系统在系统运行过程中能够发挥实际效用，满足了信息系统的安全需求，解决信息系统面临的安全问题。

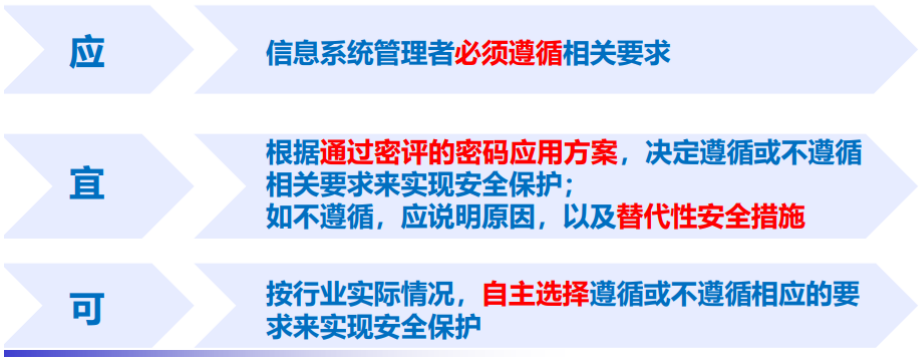
密码应用基本要求的八大安全类



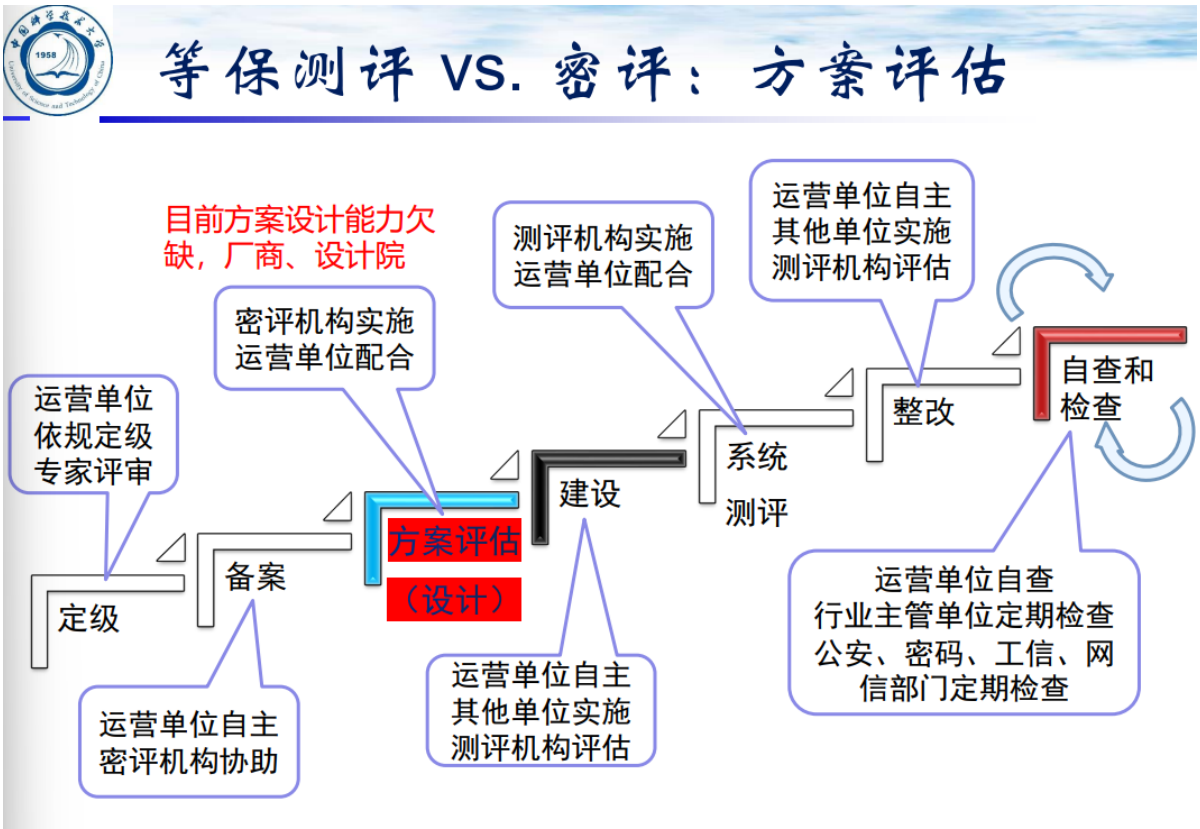
技术标准中“应”、“宜”、“可”的区别

标准的技术应用要求四个层面，采用逐级增强原则：

- “应”表示应该、要求。是要求型描述，表明符合标准需满足的要求
- “宜”表示推荐、建议。是推荐型描述，表示该条款是首选但不是必须要求
- “可”表示可以、允许。是陈述型描述，表示在标准的界限内所允许的条款



等保和密评的区别（方案评估）



密钥管理：包括密钥的产生、分发、存储、使用、更新、归档、撤销、备份、恢复和销毁等环节

云计算安全

云计算的主要特性

1. 按需自主服务：客户根据需要，获得所需计算资源
2. 泛在接入：通过各种终端随时随地使用服务
3. 资源池化：将资源提供给多个客户使用，这些物理的、虚拟的资源根据客户的需求进行动态分 配或重新分配
4. 快速伸缩性：根据需要快速、灵活、方便地获取和释放计算资源。对于客户来讲，这种资源是 “无限”的，能在任何时候获得所需资源量
5. 服务可计量：云计算可按照多种计量方式自动控制或量化资源

云计算的三种服务模式

SaaS：软件即服务，应用层

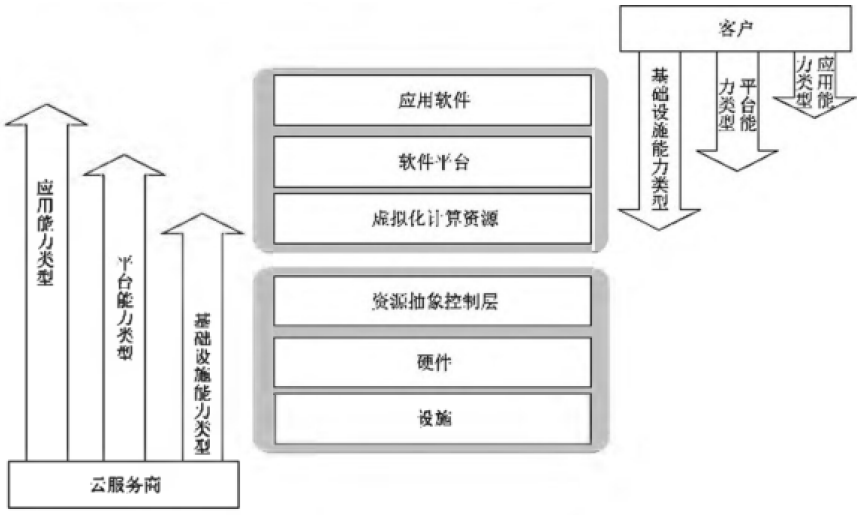
PaaS：平台即服务，平台层

IaaS：基础设施服务

云计算的部署模式（公有云、私有云、社区云、混合云）

1. 私有云：对某个特定客户，分场外（云服务商拥有管理运营）场内（客户自己搞的）
2. 公有云
3. 社区云：特定的客户群体（分场外场内）
4. 混合云 （私有链、公有链、联盟链）

云服务商和客户之间的安全责任划分（设施、 硬件、 资源抽象控制层、 虚拟化计算资源、 软件平台、 应用软件）



在图 1 中，云计算的设施层（物理环境）、硬件层（物理设备）、资源抽象控制层都处于云服务商的完全控制下，所有安全责任由云服务商承担。应用软件层、软件平台层、虚拟化计算资源层的安全责任则由双方共同承担，越靠近底层的云服务（即基础设施能力类型的云服务），客户的管理和安全责任越大；反之，云服务商的管理和安全责任越大。

考虑到云服务商可能还需要其他组织提供的服务，如 SaaS 或 PaaS 服务提供商可能依赖于 IaaS 服务提供商的基础资源服务。在这种情况下，某些安全措施由其他组织提供。

系统可信检查机制

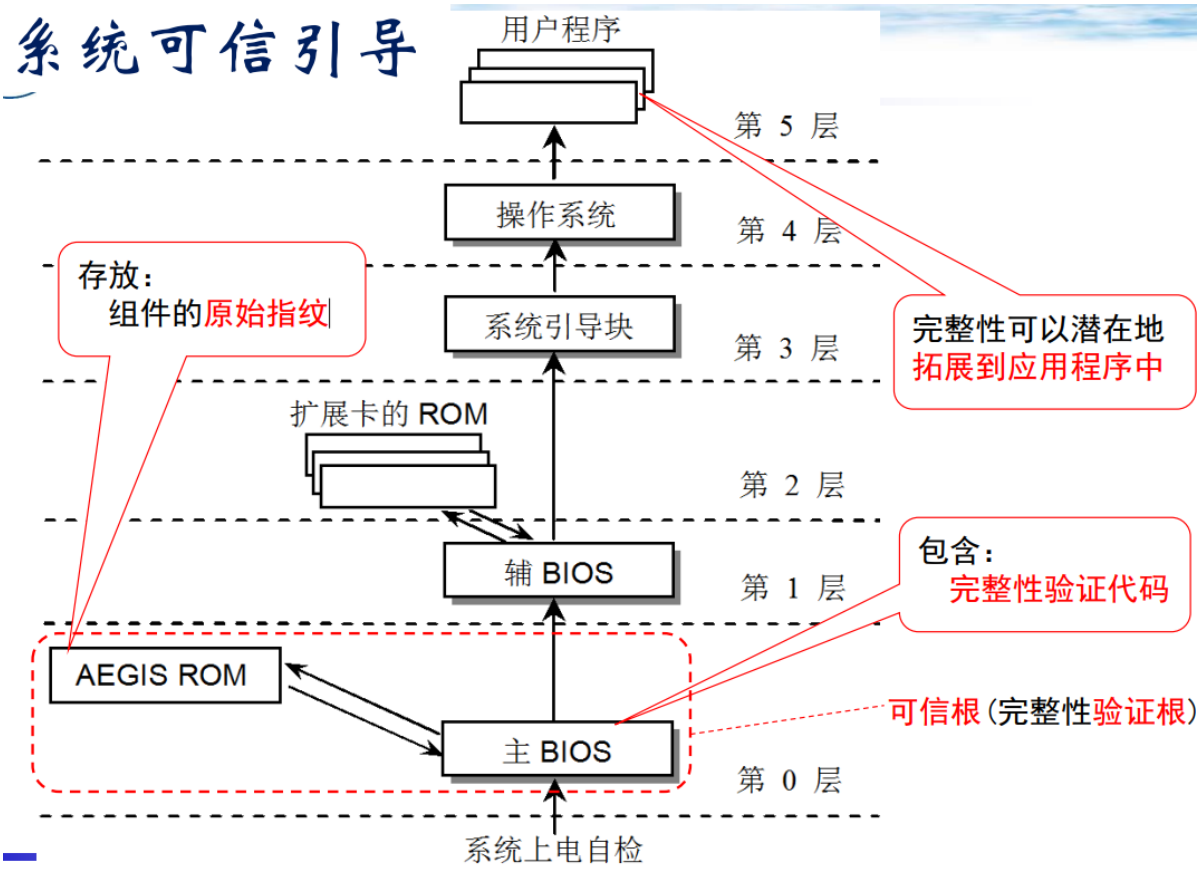
系统可信引导和系统安全引导的异同

可信引导:

在系统引导的一般过程中，组件A把控制权交给组件B时，它并不了解组件B的完整性情况。就算组件B的完整性已经受到破坏，组件A依然会把控制权交给组件B，组件B依然能够运行。这样引导结束，无法判定操作系统的完整性是否已经受到破坏

系统可信引导的目标： 确保引导过程中获得控制权的所有组件的完整性都没有受到过破坏，进而确保引导起来的操作系统的完整性是有保障的

需要对组件的完整性进行验证，组件的哈希值可以作为指纹

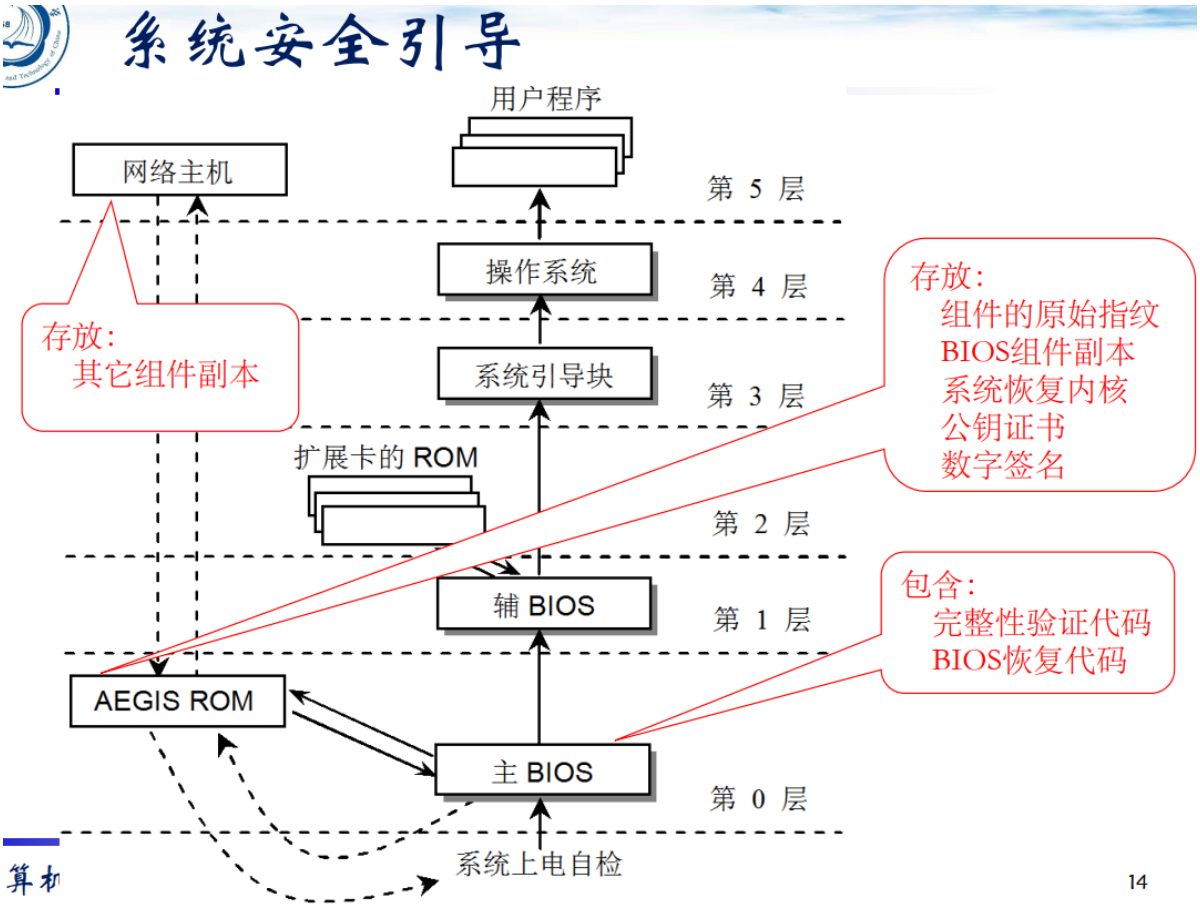


安全引导:

采用可信引导策略的系统可信引导不能抵御拒绝服务攻击DDOS

系统安全引导的目标是，不但要确保顺利引导起来的操作系统内核一定是完整性良好的，还要确保操作系统内核一定能够顺利地引导起来

为了实现系统安全引导的目标，需要在系统可信引导的基础上，增加系统恢复的功能



⌗ 练习1 对比第10页和第15页的图，说明系统安全引导与系统可信引导的共性与区别

- 共性：层次结构类似，均为0~5层结构；包含的组件大部分相同，如主BIOS、辅助BIOS、系统引导块等；运行的基本功能类似
- 区别：系统可信引导注重系统的完整性保障，但安全性薄弱，不能抵御DDOS攻击；系统安全引导在前者的基础上，使用公钥证书和数字签名来增强原始指纹的抗篡改性和真实性，同时通过添加相关副本和恢复代码增加系统恢复的功能，确保操作系统内核一定能够顺利地引导起来

进程完整性体现在哪些方面：

- 在初始状态中的完整性

  > 进程模式：普通模式（不需进行完整性验证）、篡改响应模式（需要进行完整性验证）
  >
  > 普通模式→篡改响应模式或初始状态开始：
  >
  > * 计算并检查程序哈希值
  >
  > * 检查运行环境

- 在中断过程中的完整性

  > 环境切换：
  >
  > * 保护寄存器信息 →保护环境信息的完整性

- 呈现于存储介质（片上 Cache/片外存储）的完整性

  > 读： 片外 → 片内：验证完整性------ 完整性验证机制
  >
  > 受验证内存位置：
  >
  > * 只允许一个进程修改
  >
  > * 最高地址位 = 1

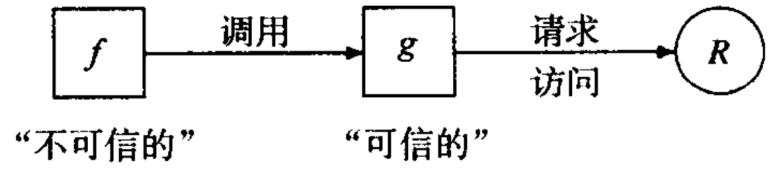
- 输出结果的完整性

  > 结果信息------数字签名
  >
  > 例：程序Prog， 结果M， CPU私钥$K_{PRV-CPU}$，
  >
  > 签名： $\{H(Prog), M\}K_{PRV-CPU}$
  >
  > 断定：特定的系统（系统认证）运行特定的程序（程序认证）得到特定的结果信息（信息认证）

基于代码的访问控制

当一个方法调用另一个方法时，什么特权是有效的?

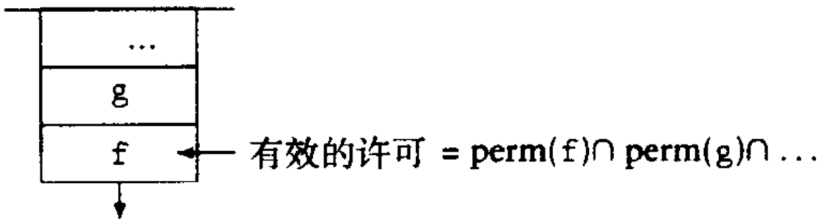
例如：函数g有访问资源R的许可，但是函数f却没有； g调用f， f请求对R的访问：这个访问应当被授权吗?



为了计算出授予代码的当前许可， 在做访问决策时需要知道整个访问链。Java VM和.NET CLR使用调用栈去管理代码的执行。每发生一个函数调用，就创建一个函数栈帧，其中包含函数的本地状态，例如直接授予给它的许可。

堆栈遍历

当函数请求访问一个受保护的资源， 堆栈遍历用来确定调用者是否有需求的许可。调用者有效许可的计算，取决于调用栈上所有函数的许可的交集。所以上面的例子中，访问不会被授权。



惰性计算(Lazy evaluation)：只有当访问一个资源需要一个许可时， 才去评估授予的许可 ，反之为**热情计算**

许可断言

允许不可信的调用者调用可信的方法，许可断言附着在当前栈帧上，当从调用部件返回时撤销，堆栈遍历在遇到许可断言的帧并授予这个许可时终止，至此被检查的所有帧都有这个许可

- 练习1 考虑一个调用它自己N次的递归函数。比较没有尾调用清除和有尾调用清除时堆栈遍历的性能。
    - 尾递归可以把堆栈中的数据清楚，把空间留给最后的递归调用，因此没有为诋毁调用的空间复杂度为O(N)，有尾递归的空间复杂度为O(1)。
    - 在时间复杂度上，由于它们都需要清除堆栈，传递参数，因此时间复杂度均为O(N)。
    - 但尾递归可以直接返回结果，而不需要层层向上传递，因此要优于没有尾递归时调用。

入侵检测

入侵检测系统（IDS）：进行入侵检测的软硬件组合

IDS基本功能部件：信息收集->信息分析->结果处理

IPS（入侵防御系统）：IDS（发现）和防火墙（阻断）联动，能够监视网络或网络设备的网络资料传输行为，能够即时的中断、调整或隔离一些不正常或是具有伤害性的网络资料传输行为

DPI（深度包检测）：基于应用层的流量检测和控制技术（识别IP包载荷）

DFI（深度/动态流检测）：基于流量行为，不同应用类型在会话连接或数据流上的状态不同，建立流量特征模型来鉴别应用类别

态势感知：基于环境的，动态、整体地洞悉安全风险的能力，以安全大数据为基础，从全局视角提升对安全威胁的发现识别、理解分析、相应处置能力的一种方式，是安全能力的落地。

入侵检测方法（异常检测、 误用检测）

- 异常检测：入侵是异常活动的子集。首先总结正常操作应该具有的特征（用户轮廓），当用户活动与正常行为有重大偏离时即被认为是入侵。

  指标：漏报率低，误报率高。不需要对每种入侵行为进行定义，因此能有效检测未知的入侵

- 误用检测：所有的入侵行为都有可被检测到的特征。收集非正常操作的行为特征（特征提取），建立相关的攻击特征库（要经常更新），当监测的用户或系统行为与库中的记录相匹配时，系统就认为这种行为是入侵。

  指标：误报低，漏报高。攻击特征的细微变化，会使得误用模式无能为力

入侵检测系统分类（按照数据来源，基于主机、基于网络、混合型）

1. 基于主机：系统获取数据的来源是系统运行的主机，保护目标也是主机
2. 基于网络：系统获取的数据是网络传输的数据包，保护的是网络的运行
3. 混合型

应急响应与灾备恢复

信息保障技术框架(IATF)

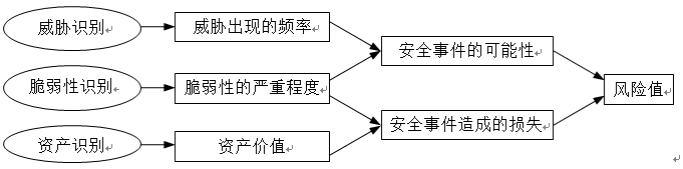
- 一个核心思想：纵深防御（深度防护）
- 三个核心要素：人（管理）、技术、操作（运维）
- 四个焦点领域（信息安全保障区域）：网络和基础设施、 区域边界、 计算环境、 支撑性基础设施

应急响应的概念：指一个组织为了应对各种网络安全事件的发生所做的准备以及在事件发生后所采取的措施。其目的是尽可能减少和控制安全事件的损失，提供有效的响应和恢复指导，并努力防止安全事件的发生。重点并不是应急，而是预防

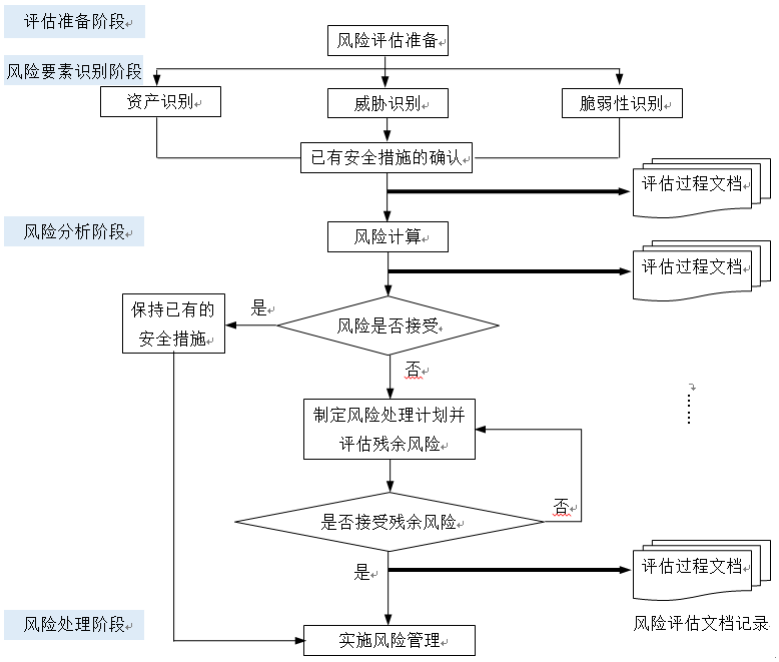
应急响应过程：应急准备（包括风险评估）、监测与预警、应急处置、总结与改进

风险评估(Risk Assessment) 过程：

1. 评估准备阶段
2. 风险评估要素识别阶段：资产、威胁、脆弱性（漏洞）识别
3. 风险分析阶段：使用风险分析模型——



4. 风险处理阶段



灾难恢复的概念

为了将信息系统从灾难造成的故障或瘫痪状态恢复到正常运行状态，并将其支持的业务功能从灾难造成的不正常状态恢复到可接受状态，而设计的活动和流程

容灾备份是灾难恢复的基础。

容灾备份的概念

利用技术、管理手段以及相关资源确保既定的关键数据、关键数据处理信息系统和关键业务在灾难发生后可以恢复和重续运营的过程。

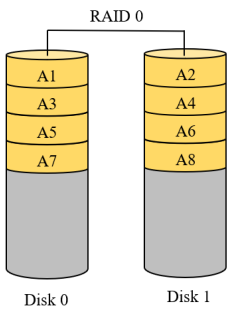
分为数据容灾、 应用容灾。

一个完整的容灾备份系统通常主要由数据备份系统、 备份数据处理系统、 备份通信网络系统和完善的灾难恢复预案(计划)所组成。

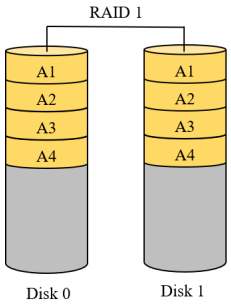
容灾备份与恢复关键技术——主流存储技术：RAID（独立冗余磁盘阵列）

RAID 0、 RAID 1、 RAID 10 和 RAID 5

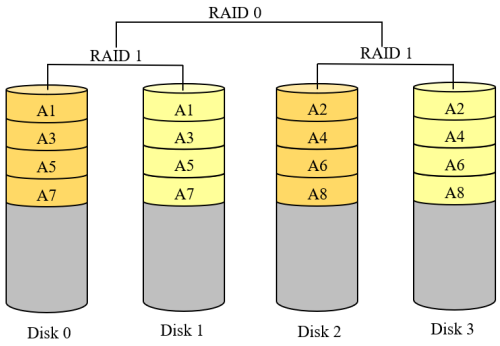
RAID 0：把多块硬盘连成一个容量更大的硬盘群，可以提高磁 盘的性能和吞吐量 ，但没有冗余或错误修复能力。成本低，要求至少两个磁盘。



RAID 1称为磁盘镜像：把一个磁盘的数据镜像到另一个磁盘上，磁盘利用率为50%，成本最高。



RAID10：镜像阵列条带，像RAID0一样，数据跨磁盘抽取；像RAID1一样，每个磁盘都有一个镜象磁盘，所以RAID 10又叫 RAID 0+1。提供100%的数据冗余，支持更大的卷尺寸，价格高



RAID5：1+0的折衷方案，但没有完全使用RAID 1镜像理念，而是使用了“奇偶校验信息”来作为数据恢复的方式 ，至少需要三块硬盘。奇偶校验信息是异或前两个数据块得到的

数据写入会根据算法分成3部分，然后写入这3块硬盘，写入的同时还会在这3块硬盘上写入校验信息。当读取写入的数据的时候会分别从3块硬盘上读取数据内容，再通过检验信息进行校验。当其中有1块硬盘出现损坏的时候,就从另外2块硬盘上存储的数据可以计算出第3块硬盘的数据内容。只允许有一块硬盘出现故障，出现故障时需要尽快更换。

是一种存储性能、数据安全和存储成本兼顾的存储解决方案。