Computer Security

Chapter 1 绪论

信息安全概念

- 信息安全学关注信息本身的安全，可以分为数据安全和系统安全。信息安全的任务是保护信息财产，防止遭到恶意泄露、修改和破坏等

CIA

- 保密性
- 完整性
- 可用性

系统研究的方法

- 还原论
  - 把大系统分解为小系统，然后通过对小系统的研究去推知大系统的行为
- 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

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

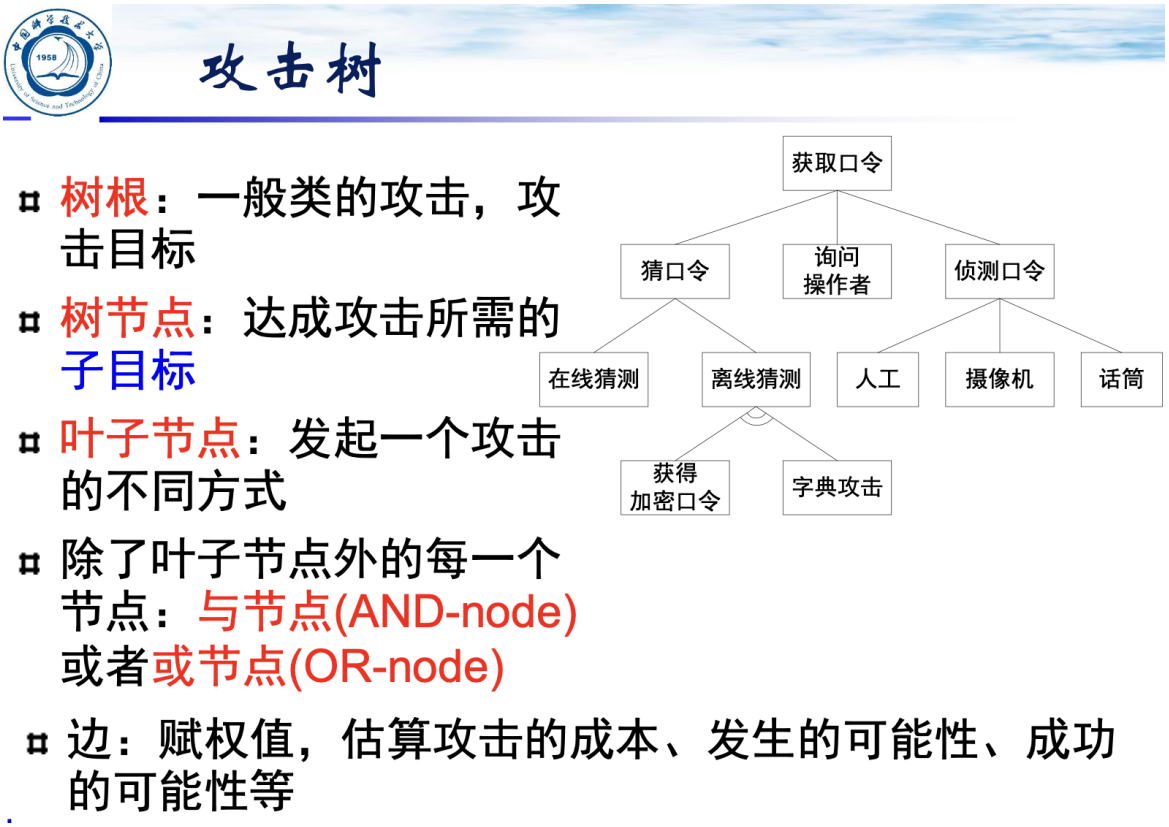
系统安全思维

- 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

风险分析的基本方法

- 风险 = 资产×漏洞×威胁
- 漏洞
  - 系统的脆弱性
  - 攻击面：由系统中可到达的和可被利用的脆弱点构成
    - 网络攻击面
    - 软件攻击面
    - 人为攻击面
- 威胁

- 利用漏洞去损坏资产的行为
- 攻击树：定量分析攻击



- .

- 社会工程学
  - 利用人的薄弱点，通过欺骗手段而入侵计算机系统的一种攻击方法

Chapter 4 访问控制

RBAC

- 将访问权限分配给角色
  - RBAC0
    - 用户 角色 客体 操作 许可
    - 用户指派 许可指派
    - 权限被赋予角色而不是用户
    - 会话：用户 <-> 激活的角色集合
    - 一个会话构成一个角色到多个角色的映射，激活了用户授权角色集的某个子集 -- 活跃角色集
  - RBAC1
    - 角色继承
      - 一般继承
        - 绝对偏序关系，允许多继承
      - 受限继承
        - 树结构，单继承
  - RBAC2
    - 责任分离

        - 动态
        - 静态
      - 约束规定了强制性规则
    - RBAC3
      - 角色继承 责任分离
- 特点
  - 最小权限原则 责任分离原则
  - 数据抽象原则和继承概念

Chapter 2 3 4 5

- 未整理

Chapter 5+ 计算机实体安全

可信计算

- 如果它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
- 以 TPM 为基础的可信计算
  - 用户的身份认证

TPM 可信平台模块

- 具有安全存储和加密功能的 TPM，可地址篡改的独立计算引擎
- 2003 年 1.2 标准 103 版本 为 ISO 标准
- 含有密码运算部件和存储部件的小型片上系统
- 核心功能
  - 对 CPU 处理的数据流进行加密，同时监测系统底层的状态
  - TPM 和可信计算软件栈 TSS 结合构建可信计算体系结构

软件狗

- 电子设备，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，则说明程序合法

Chapter 6 Unix/Linux 安全

SUID/SGID 受控调用

- SUID SGID 程序于属主或属组的有效 UID 或者有效 GID 一起运行，拥有暂时的或者受限制的访问权限

强制访问控制机制

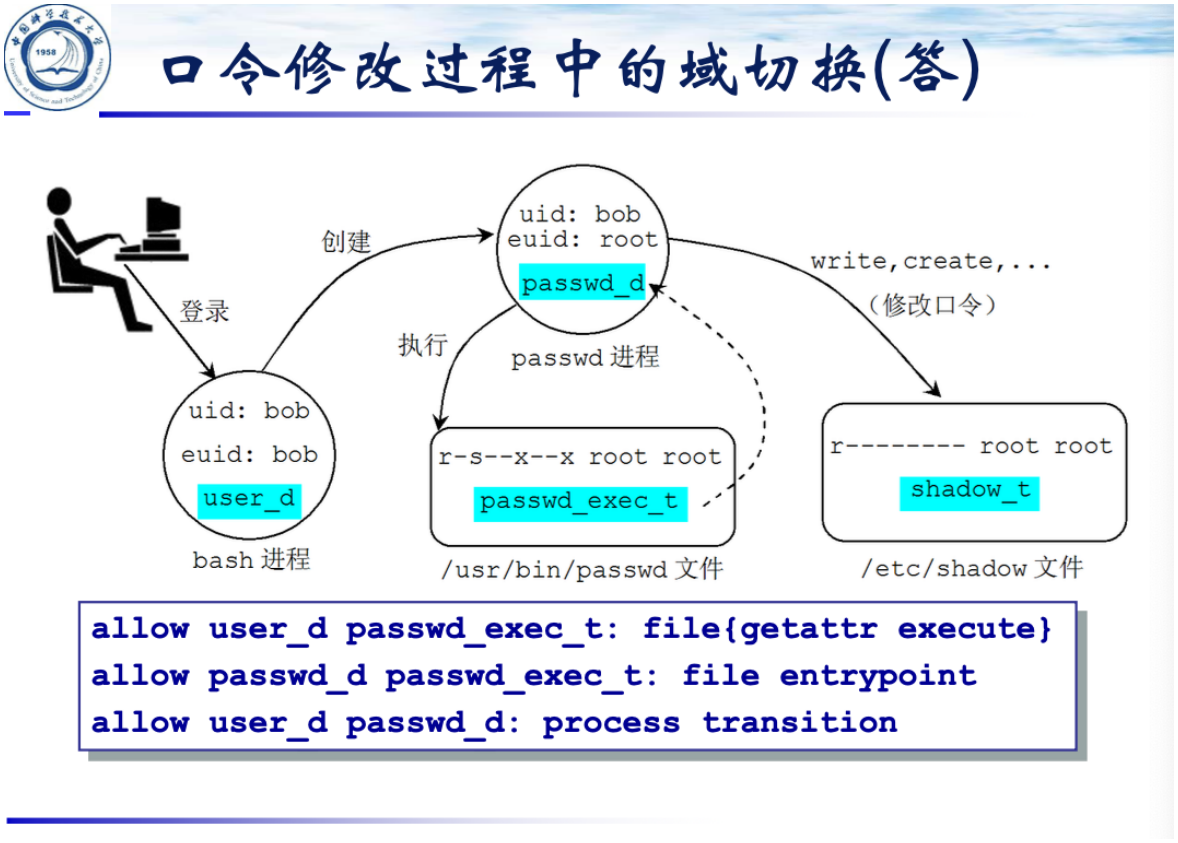
- SELinux 是强制访问控制的实现

SELinux 优点

- 强制访问控制 MAC -- 对访问的控制彻底化
  - 基于策略设定
- 类型强制 TE -- 对于进程只赋予最小的权限
  - 访问控制模型的核心 -- DTE 模型
    - 主体分到域，客体设定类型，主体客体都被称为实体
    - SETE
    - 类型细分、权限细化
- Domain 迁移 -- 防止权限升级
- 基于角色的访问控制 RBAC -- 对于用户只赋予最小的权限
  - ROLE 也可以迁移，但只能按规定

SETE 模型域切换

- 条件
  - 进程的新的工作域必须拥有对可执行文件的类型的 entrypoint 访问权限
  - 进程的旧的工作域必须拥有对入口点程序的类型的 execute 访问权限
  - 进程的旧的工作域必须拥有对进程的新的工作域的 transition 访问权限
- 口令修改过程中的域切换



Chapter 6+ Android 安全

Android 系统架构

- 分层架构

- Linux 核心层
  - 内核、驱动
- 系统运行库层
  - C/C++ 程序
- 应用程序框架层
  - Dalvik 虚拟机
- 应用程序层
  - Java 程序

Dalvik 虚拟机

- Google 为 Android 系统设计的 JVM，不遵循 Java 官方的 JVM
- 指令集基于寄存器架构
  - 区别于 Java 基于栈
  - dex 字节码
  - 核心内容是实现库 libdvm.so，大体由 C 实现
- Dalvik VM vs JVM
  - Dalvik 基于寄存器，JVM 基于栈
  - Dalvik VM 对提前优化提供更好支持
  - 基于寄存器的 VM 对于大程序来说编译时花费时间更短
  - JVM 运行 .class 文件格式，文件间会有不少冗余
  - dex 字节码将多个文件整合为一个
  - 一个应用，一个 VM 实例，一个进程
    - 每个应用都运行在 Dalvik VM 实例里，每个实例都是一个独立进程
    - IPC 通信
    - 不同的应用在不同的进程空间里运行

Android 问题

- JVM 先天不足
  - JIT 占用硬件资源且无法最大发挥软件运行性能
- 额外 Java 原生接口(JNI) 开销
- 代码优化空间有限
- 内存回收机制容易造成间歇性卡顿

ART

- 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：启动更快，更省资源
  - 缺点：增加程序安装时间，占更多空间
- 混合模式

        - AOT + JIT + 解释执行
        - 安装时 dex 不会被预编译
        - 运行时 dex 通过解释器直接执行，热点函数被识别并用 JIT编译存储在 profile 文件
        - 设备进入 idle 状态或充电状态，系统扫描 profile 文件进行 AOT 编译

Android 安全机制

- 重要安全设计
    - 沙箱
        - **进程沙箱隔离机制**
    - 权限
- 主要安全机制
    - **用户 ID**
        - 一个应用程序就是一个用户，拥有独立的 UID
        - 根据其UID为之分配一个独立的虚拟机
    - **权限**
        - 权限是在程序安装的时候确定
        - 权限组：把权限按照功能分成不同的集合
        - 用 protectionLevel 标识保护级别
            - 普通级别 危险级别 签名级别 系统/签名级别
    - **签名**
        - 仅拥有相同签名的两个应用程序会给相同的 UID
- 进程级实施安全限制措施

Chapter 7 Windows 安全

Winlogon

- Winlogon 机器总是运行着一个以 SYSTEM 为主角的登录进程 winlogon.exe

本地安全权威 LSA

- 用户登录时，检查有用户账户并创建访问令牌
- 负责审计功能

安全账户管理员 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

注册表

- Windows 配置数据的中央数据库，表项称为键，指向操作系统搜索特定可执行文件的位置

域

- 共享公用用户账户数据库和安全策略计算机的集合
- 实现 一次签到 以及 集中式安全管理

活动目录

- Windows 数据组织方式，可以被看作一颗由特定类型对象构成的树
- 功能
  - 基础网络服务
  - 计算机管理
  - 用户服务
  - 资源管理
  - 桌面配置
  - 应用系统支撑

主角

- 作用域
  - 本地主角
    - 本地管理，只对本地计算机可见
  - 域主角
    - 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

主体

- 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**中

受限上下文

- 受限令牌可以从一个给定的访问令牌构建
- 受限SID的创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，为每一个对象类型的ACL，添加到所要访问的主体的受限令牌中

任意访问控制列表 DACL

- 访问控制表项 (access control entries, ACE) 的列表
- ACE 格式：
  - 类型：肯定(允许)或否定(拒绝)
  - 访问权限
  - 主角SID：ACE应用的主角
  - ObjectType：对象类型

    - 标志：例如继承标志
    - InheritedObjectType：继承对象类型

系统访问控制列表 SACL

- SACL是非自主的，由管理员设置
- ACL中的ACE格式
    - 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

数据执行保护 DEP

- 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
- 基本原理：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

Chapter 8 数据库安全

关系数据库

- 关系类型
    - 基本关系(实际关系)：命名的自主的关系；独立存在，不是派生于其他关系，并且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
    - 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：可能没有名字；并不是常驻在数据库中
- 存储过程 vs 函数
    - 存储过程（Stored Procedure）是在大型数据库系统中，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
    - 数据库函数是指当需要分析数据清单中的数值是否符合特定条件时，使用数据库工作表函数
- 触发器：触发器是一种特殊类型的存储过程，主要是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

授权管理策略

- 集中式
- 基于属主
    - **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

- 委托管理下的授权回收：A->B->C，A回收B的授权时，有时也需要回收C的授权（**递归回收**）

否定式授权

- 问题：S1不向S2发放授权A，但S3可能向S2发放，这样S2依然会拥有授权A
- 常规授权： GRANT (S, O, A)；否定式授权： GRANT (S, O, NO-A)

解决授权冲突的原则

- 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（且不管发放顺序）
- 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

视图优势

- 支持基于内容的访问控制
- 可以将复杂的查询编写为视图，减少用户查询的复杂度
- 限制某个视图只能访问基表中的部分数据，提高了安全性
- 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
- 安全的视图可代替安全标签（类似强制访问控制）
- 数据可以很容易地重新分类

用户发放授权 vs 角色发放授权

由角色发放授权

- **案例：**设U0和U1是用户，R2和R3是角色，R3是R2的成员，U0是R3的成员，U1暂无任何授权，U0是当前会话用户。

  场景1：

  GRANT R2 TO U1;　　（操作由当前会话用户U0执行）

  场景2：

  SET ROLE R3;　　（把R3设为当前会话角色）

  GRANT R2 TO U1

  GRANTED BY CURRENT_ROLE;　　（操作由角色R3执行）

- **分析：**在场景1中，执行发放授权操作的是用户U0，如果撤销用户U0的授权R3，用户U0的授权R2被撤销，用户U1的授权R2被连带撤销。在场景2中，执行发放授权操作的是角色R3，即使撤销用户U0的授权R3，用户U1的授权R2也不受影响。
- **结论：**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。

- 结论
  - 用户执行授权操作时，回收授权 **引起递归回收**
  - 角色执行授权操作时，回收授权 **不引起递归回收**

三元组安全标签 OLS-BLP标签

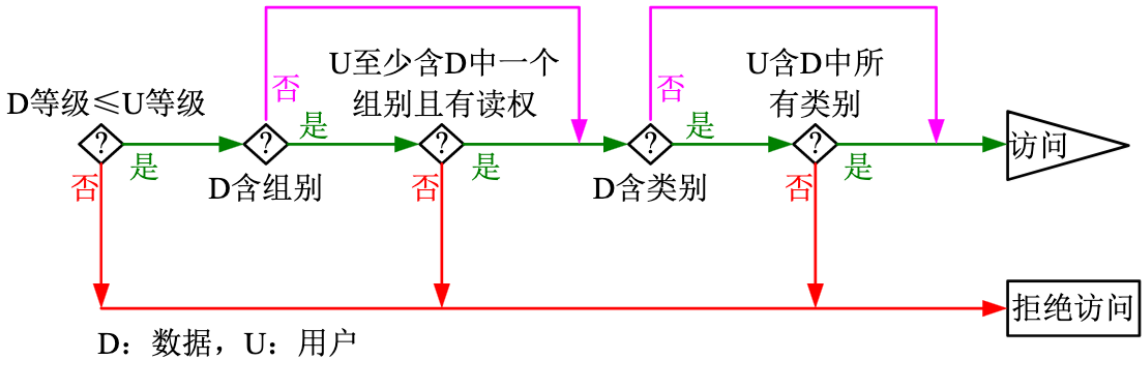
- 三元组：<等级，类别，组别>
- 一般表示：(等级：类别：组别)

基于标签的访问判定

- 读

基于标签的“读”访问判定

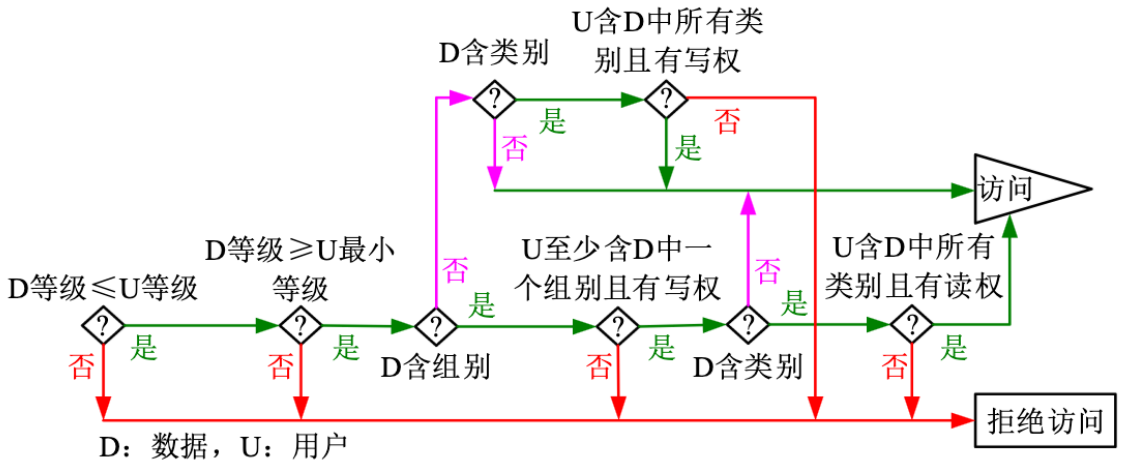
- 用户标签中的等级必须大于或等于记录标签中的等级
- 用户标签中的组别至少必须包含记录标签中的一个组别，并且用户拥有对该组别的读权限
- 用户标签的类别必须包含记录标签中的所有类别



- 写

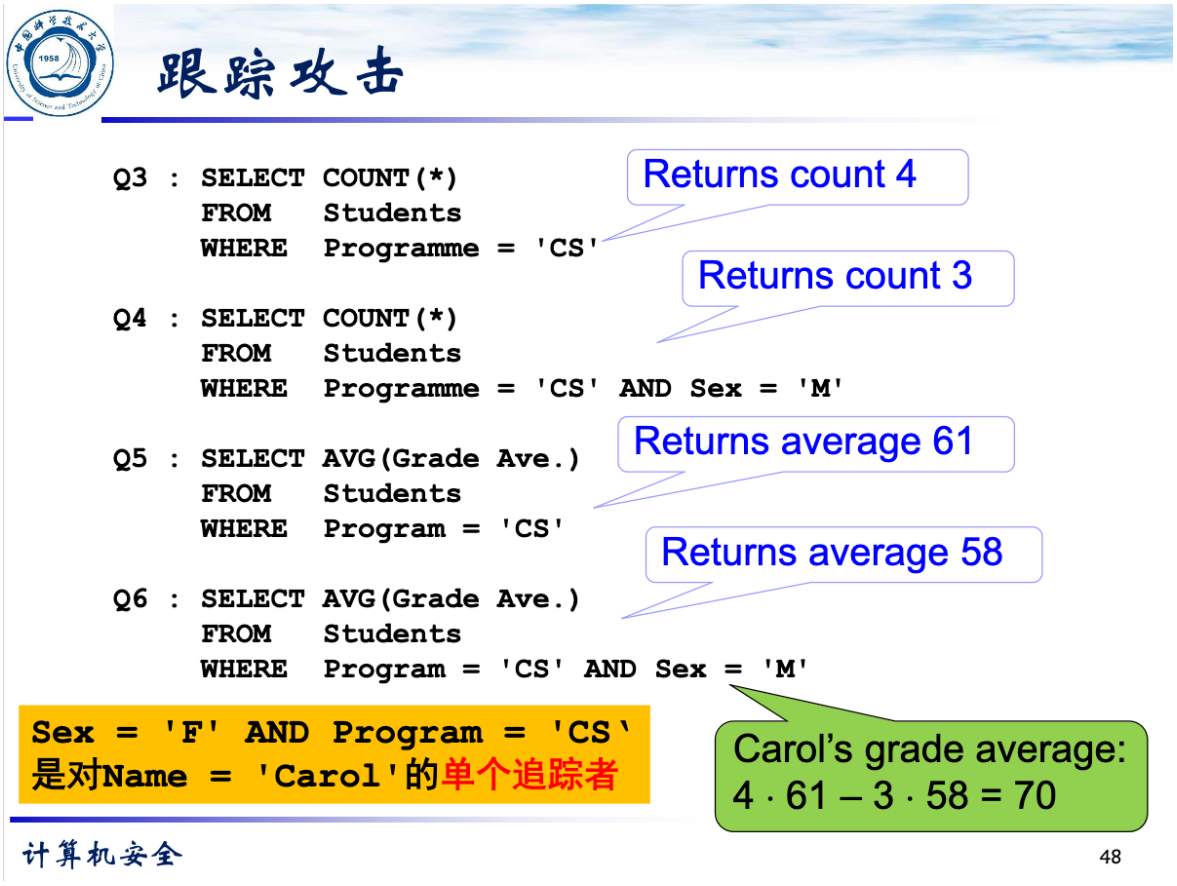
基于标签的“写”访问判定

- 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



统计数据库的攻击

- 统计数据库：对一张表中的一个属性(列)进行统计(聚集)查询来检索信息
- 推理攻击
  - 推理：由非敏感数据推断出敏感数据
  - 推理问题：从一些不敏感的数据中，派生出一些敏感的数据
    - 直接攻击：聚集是从小组样例中计算出来的，这样私人的数据项就可能被泄露出去
    - 间接攻击：这种攻击结合了与几个聚集关联的信息
    - 跟踪攻击：这是间接攻击的一种非常有效的类型
    - 线性系统的漏洞攻击：这种攻击又比跟踪攻击进了一步，它采用查询集中的算术关系，来构建等式，以产生期望的信息
- 跟踪攻击
  - 允许对单个元组追踪其信息的查询谓词T，被称为该元组的单个追踪者(individual tracker)



- 差分隐私
  - 差分隐私需要做到的就是使得攻击者的知识不会因为这些新样本的出现而发生变化
  - 加入随机噪声
  - 优点
    - 差分隐私严格定义了攻击者的背景知识
    - 差分隐私拥有严谨的统计学模型
    - 差分隐私不需要特殊的攻击假设

三种备份方法

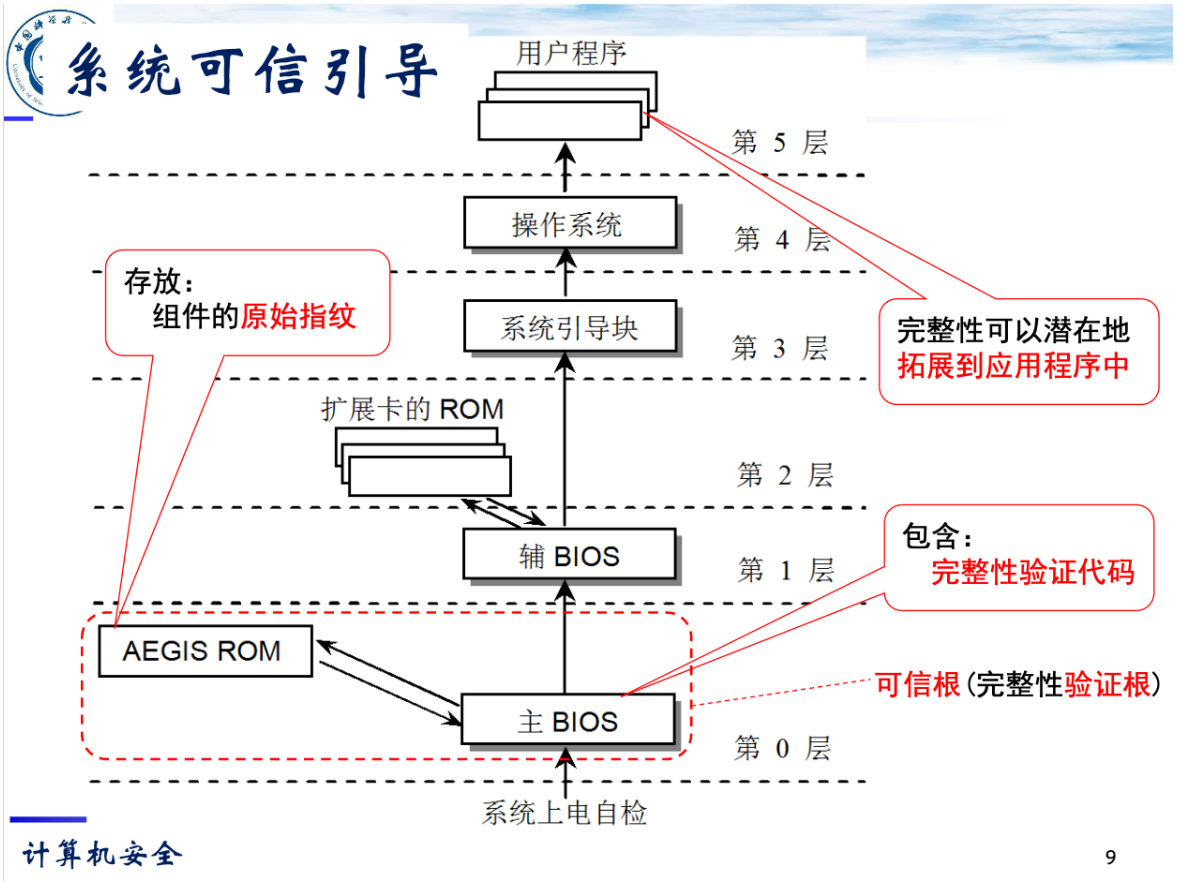
- 完全备份

- 用一块磁盘对整个系统进行包括系统和数据的完全备份
- 增量备份
  - 每次备份只是相对于上一次备份后的增加和修改
- 差分备份
  - 每次备份的数据是相对于上一次全备份之后新增加和修改的数据

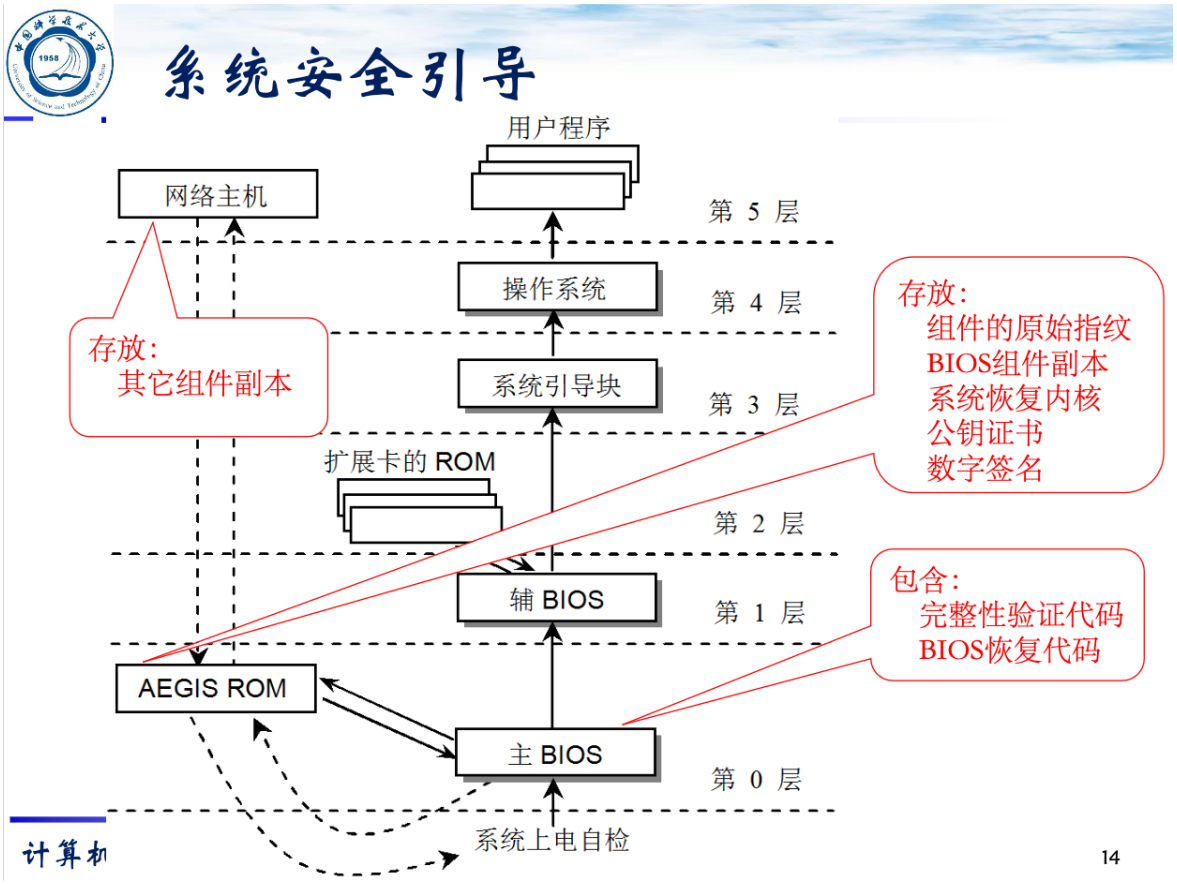
Chapter 9 系统可信检查机制

系统可信引导 vs 系统安全引导

- 系统可信引导
  - 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，原始指纹和即时指纹对比



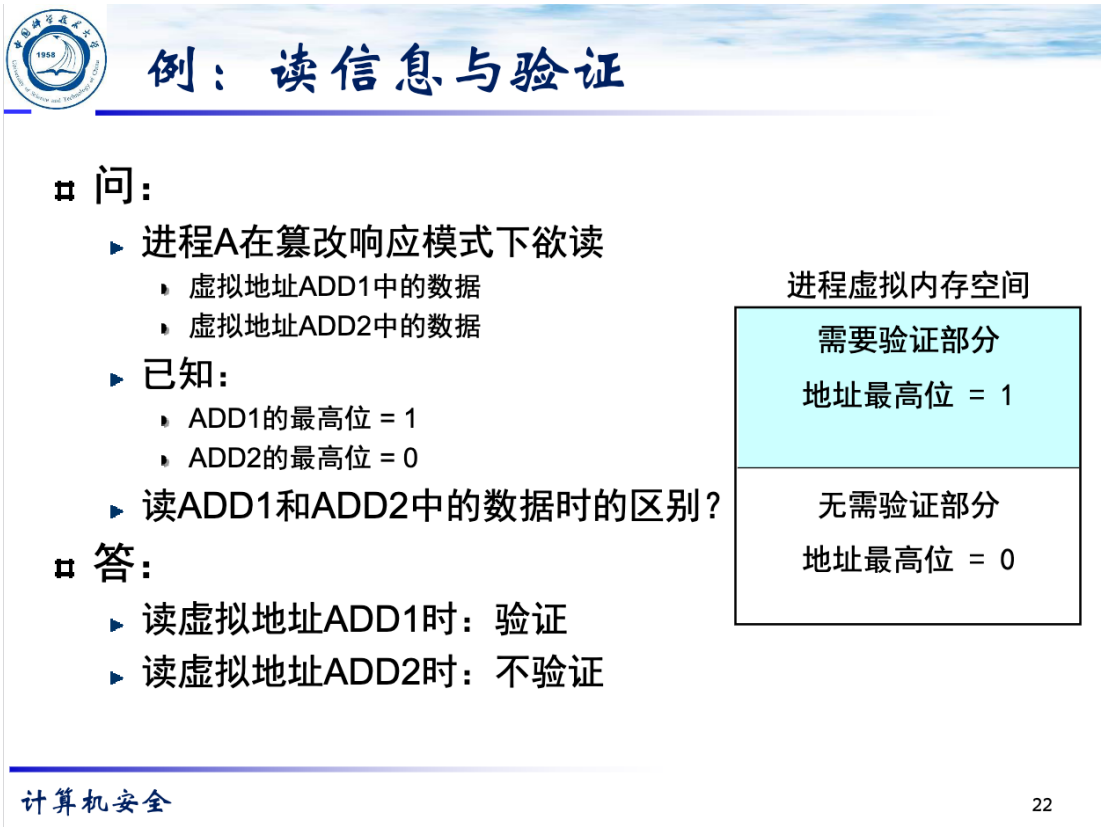
  - UP-AEGIS机制
    - 在系统中增设了一个专用的ROM卡，称为AEGIS ROM，用于存储组件的原始指纹
    - 将系统BIOS分为两部分
      - 主 BIOS：包含执行完整性检验任务的代码
      - 辅 BIOS：包含 BIOS 其他成分以及 CMOS
    - 假设 AEGIS ROM 和 主 BIOS 是可信的
      - 他们包含可信软件，是值得信赖的完整性验证的根（可信根）
    - 不能抵御拒绝服务攻击

- 系统安全引导
  - 目标：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
  - 在系统可信引导基础上增加系统恢复的功能
  - 

进程完整性的体现

- 进程的完整性
  - 在初始状态中的完整性
    - 普通模式 → 篡改响应模式：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
  - 呈现于存储介质时的完整性
    - 代码和数据驻留的存储介质
      - 片上 cache
      - 片外内存
      - 片外磁盘
    - 读：
      - 片外 → 片内：验证完整性 -- 完整性检验机制
    - 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）
- 完整性验证方法
  - 指纹法（哈希值法）
- 进程模式
  - 普通模式：不需要进行完整性检验
  - 篡改响应模式：需要进行完整性检验

Chapter 10 BLP 模型

安全模型

- 表示一个特定策略或一组策略的模型
- BLP 模型是第一个比较完整地形式化方法对系统安全进行严格证明的数学模型

状态集

- 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

- 模型状态集必须包含当前所有的许可和当前所有主体访问客体的实例
- 状态集 $V = B \times M \times F$
- $B = P(S \times O \times A)$ 是当前访问的集合
  - 元素 $b \in B$ 是一个三元组 (s, o, a) ，表示主体 s 当前在客体 o 上执行操作 a
- 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 $m = (m_{so})\ s \in S, o \in O$
  - $M = \{\{read, write\}_{Alice, fun.com}, \{read, write, execute\}_{Bob, fun.com}\}$
- $F \subset L^S \times L^S \times L^O$ 是安全级别分配的集合
  - 元素 $f \in F$ 是一个三元组 (f_S, f_C, f_o)
    - $f_S : S \to L$ 给出每个主体可拥有的最高安全级别
    - $f_C : S \to L$ 给出每个主体当前安全级别
    - $f_o : O \to L$ 给出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
  - 高安全级别有时称为主体的许可
- 层次 H ，有时状态由 (b, m, f, h) 决定
  - 客体间的层次关系，典型情况是客体在文件系统中的树状结构关系
  - 客体的安全级别必须支配其父节点的安全级别

安全策略

- 简单安全策略 ss-property
  - 对于每一个元素都有 $(s, o, a) \in b$ ，访问操作 a 是 读 或 写，主体 s 的安全级别控制客体 o 的安全级别，即 $f_S(s) \geq f_O(o)$ ，那么状态 (b, m, f) 满足 ss-property
  - 无向上读安全策略
  - 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
- 星特性 *-property

控制写访问的策略

- 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)

星特性（*-property）

- 第一种方法：假设主体被降级时，它会忘记它在较高安全级别上所知道的一切
  - 当把主体看成人时，这个观点看起来不合情理，但BLP是模拟计算机的
  - 在此主体(进程)没有自己的记忆，它们所“知道”的唯一事情是允许它们查看的对象的内容
  - 在这种情况下，临时降级确实解决了问题
- 换句话说，f_s规定了用户的许可，用户允许以低于许可的级别注册，f_C表明了用户实际注册的级别

- 无向下写安全策略
- 对于每一个元素，均有 $(s,o,a)\in b$，访问操作是 添加 或 写，主体 s 的当前级别受客体 o 的当前级别控制，即 $f_S(s)\le f_O(o)$，那么状态满足 *-property
- 此外，如果存在一个元素，有 $(s,o,a)\in b$，访问操作是 添加 或 写，那么对于所有客体 o'，$(s,o',a')\in b$，a' 是 读 或 写，必须有 $f_O(o')\le f_O(o)$

- ss-property 和 *-property 称为强制 BLP 策略
- *-property只对不可信的主体有效

可信主体

- 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
- *-property只对不可信的主体有效
- 如果确信一个主体不会造成伤害，则称它是值得信赖的(trustworthy)
- 要区分可信主体(trusted subjects)和值得信赖的主体(trustworthy subjects)

- ds-property

自主安全性（ds-property）

- 橙皮书使用术语自主访问控制（discretionary access control, DAC）来表示那些基于指定的用户和指定的客体的访问控制策略。拥有访问许可的主体可以将许可传递给其他主体。在BLP中，这种策略通过一个访问控制矩阵表示，并通过自主安全特性（ds-property）获得
- ds-property 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ds-property

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

隐蔽信道

- 指不被设计者或者用户所知道的泄露系统内部信息的信道，是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
- 关键属性
  - 存在性
  - 带宽
- 分类
  - 隐存储信道
    - 空间共享，共享资源的属性的修改
    - 必要条件
      - 资源共享
      - 发送信息
      - 接收消息
      - 同步或协同
    - 资源消耗信道、事件计数信道
  - 隐定时信道
    - 时间共享，共享资源访问中的时态或有序关系
    - 必要条件
      - 资源共享
      - 基准时间
      - 同步或协同

Chapter 11 安全模型

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（这里没有 "无XX读"，BLP 为 无向上读、无向下写）
    - 如果主体 s 可以修改客体 o，则 $f_S(s) \geq f_O(o)$
  - 完整性 *-property
    - 如果主体 s 可以读客体 o ，那么仅当 $f_O(o) \geq f_O(o')$ 时，s 可以写另一个客体 o'
- 动态完整性级别

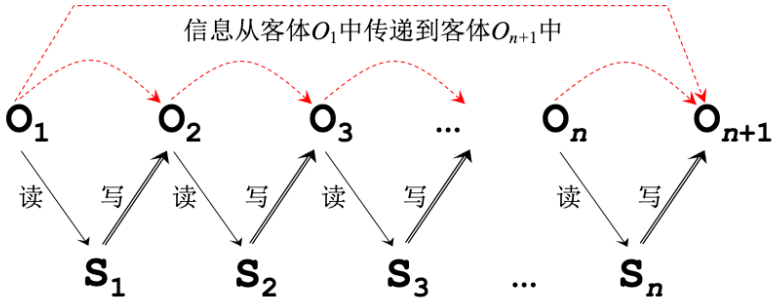
动态完整性级别（Dynamic Integrity Levels）

- 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）
  - 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 $\inf(f_S(s), f_O(o))$
  - 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$\inf(f_S(s), f_O(o))$
  - 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

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$S_i\ r\ O_i$和$S_i\ w\ O_{i+1}$



- 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$f_O(O_k) \ge f_O(O_{k+1})$

- 调用策略
  - 主体可以调用另一个主体来访问客体
  - 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，即主体 s_1 可以调用主体 s_2 ，仅当 $f_S(s_1) \geq f_S(s_2)$
    - 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 $f_S(s_1) \geq f_O(o)$ 时修改客体 o ，仅当 $f_S(s_1) \geq f_S(s_2)$ 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

Chinese Wall 模型

- 公司的集合用 C 表示
- 主体：分析员，主体的集合用 S 表示
- 客体：信息条目，每一客体均指向一个独立的公司，客体的集合用 O 表示
- 公司数据集(company dataset)
    - 有关同一公司的所有客体的集合。函数 $y : O \to C$ 给出了每个客体的公司数据集
- 利益冲突类(conflict of interest)
    - 相互竞争的公司
    - 函数 $x : O \to P(C)$ 给出了客体o的利益冲突类，即不应该知道该客体内容的公司集合
- 安全标签
    - (利益冲突类,公司数据集) 即 $(x(o), y(o))$
- 净化的信息
    - 去除了敏感细节，不受访问限制
    - 一个被净化的客体的安全标签为 $(\Phi, y(o))$
- ss-property

ss-property

- 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
- 形式化：
  - $N = (N_{so})_{s \in S, o \in O}$，布尔型矩阵，$N_{so}$ = true表示主体s访问过客体o
  - ss-property：主体s允许访问客体o，仅当对于所有$N_{so'}$ = true的对象o'，有$y(o) = y(o')$ 或 $y(o) \notin x(o')$

- *-property

* - Property

- * - Property：主体s允许对客体o进行写访问，仅当s对$y(o) \neq y(o')$ 和 $x(o') \neq \Phi$的对象o'没有读访问权
  - 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能被阅读时，对一个客体的写访问才是允许的
- 每次访问操作，主体的访问权限都要重新赋值

信息流模型

信息流模型（information flow model）

- 在BLP模型中，信息可以通过隐蔽信道从高安全级别流到低安全级别
- 信息流模型考虑了所有形式的信息流，而不光是由BLP模拟的通过访问操作的直接信息流
- 如果能够通过观察y获得更多关于x的信息，状态迁移即会引发从对象x到对象y的一个信息流
- 可以区分以下两种情况：
  - 显示信息流： *y:= x* (执行后，查看y，揭示x值)
  - 隐式信息流：IF *x=0* THEN *y:=1* (执行后，查看y，可以获得有关x的一些信息)

强信息流和弱信息流

- 强信息流（显式信息流）
  - x的值直接影响y的值，例如赋值
  - y:=f(x)，f(x)是一个带变量x的算术表达式
- 弱信息流（隐式信息流）
  - 信息从x流向y，但不存在形如y:=f(x)的显示赋值，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
  - IF $(f_1(x):=0)$ THEN $y:=f_2(a)$ *ELSE* $y:=f_3(b)$，其中f_1、f_2和f_3是算术表达式，a和b是客体

- 信息量 / 条件熵 计算

熵

- x到y的信息流用给定y的值后的x的信息量平均值(条件熵)的变化来度量
- 令$\{x_1, x_2, \ldots, x_n\}$和$\{y_1, y_2, \ldots, y_m\}$分别是变量$x$和$y$为概率$p(x_i)$和$q(y_j)$的可能取值，设$p(x_i, y_j)$为$x$和$y$取$x_i$和$y_j$值的联合概率，设$p(x_i|y_j)$为$y$取$y_j$值时$x$取$x_i$值的条件概率，则给定$y$的值后的$x$的熵$H_y(x)$定义为 $H_y(x) = -\sum_{i=1}^{n}\sum_{j=1}^{m} p(x_i, y_j)\log_2 p(x_i|y_j)$

因为 $p(x_i, y_j) = p(x_i|y_j)q(y_j)$

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

例子

- 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

$$p(0|0) = p(1|1) = 0, p(1|0) = p(0|1) = 1, \text{因此} H_y(x) = 0$$

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

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

- 安全的必要条件
  - 使信息流安全的条件为 lub{b, c, d} <= a
  - 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 $S_1 \ldots S_n$ 都是安全的
  - 使信息流安全要求对 S_1 有 $lub\{a, b\} \leq x$ 以及对 S_2 有 $lub\{x, z\} \leq y$，所以安全信息流的必要条件是 $lub\{a, b\} \leq x$ 以及 $lub\{x, z\} \leq y$

        - x = a + b; y = x + z;
    - 略

Chapter 12 安全评估

安全评估框架

- 评估对象
    - 产品：即已出售的部件，它将被用于多种应用且需要满足一般的安全要求
    - 系统：即满足给定应用特定需求的各种产品的集合体
- 评估目标
    - 评估：评定一个产品是否具有它所声明的安全属性
    - 证明：评定一个(已评估的)产品是否适合给定的应用场合
    - 鉴定：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
    - 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
    - 必要要求：可重复性 和 可再现性

TCSEC

- 安全级别定义递增
- 四个安全等级
    - D - 最小保护级
        - 不能在多用户环境下处理敏感信息
    - C - 自主保护级
        - 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
        - C1 - 自主安全保护级
            - 访问控制
            - 适用于处理同一敏感级别数据的多用户环境
            - **可信计算基 TCB 通过隔离用户数据**使用户具备自主保护能力
        - C2 - 控制访问保护级
            - 比C1级具有更细粒度的自主访问控制
            - 审计
            - 大多数操作系统都是 C2
    - B - 强制保护级
        - 必须携带 敏感标记
        - B1
            - 包含 C2 的全部特性
            - 命名主体和客体的强制访问控制

        - 消除测试中发现的所有缺陷
    - B2
        - 对 B1 的访问控制扩展到所有的主体与客体
        - 对于登录和初始认证，必须有可信路径
        - **对隐蔽信道分析**
        - 提供严格的配置管理控制
        - 统应具备相当的抗渗透能力
    - B3
        - **对 B2 增加了访问（引用）监控器**
- A - 验证保护级
    - A1 - 验证设计级
        - 要求更严格的配置管理
        - 要求建立系统安全分发的程序
        - 支持系统安全管理员的职能
- 特点
    - 只考虑保密性，不考虑完整性、可用性

ITSEC

CC

共同标准（Common Criteria）

- 对产品和系统(评估对象TOE)进行安全评估的标准
- 保护框架(Protection Profile)：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(Security Target)：表示一个特定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(Evaluation Assurance Level)：定义了一个TOE的开发者和安全评估者的责任；有七个递增定义的级别：EAL1~EAL7

CC评估保证级别（EAL）

- 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

Chapter 13 网路安全等级保护

等级保护的内涵

- 等级保护概念
  - 根据信息系统应用业务重要程度及其实际安全需求
  - 实行分等级、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 blablabla

等级保护制度主要内容

- 国家秘密信息 法人和公民的专有信息 公开信息 分类分等级进行管理和保护
- 对信息系统分等级保护
- 对信息安全产品分等级许可管理
- 对安全服务资质分等级许可管理
- 对信息安全事件分等级响应、处置

等级保护1.0十大安全类

- 技术要求
  - 物理安全
  - 网络安全
  - 主机安全
  - 应用安全
  - 数据安全
- 管理要求
  - 安全管理制度
  - 安全管理机构
  - 人员安全管理
  - 系统建设管理
  - 系统运维管理

等级保护2.0十大安全类

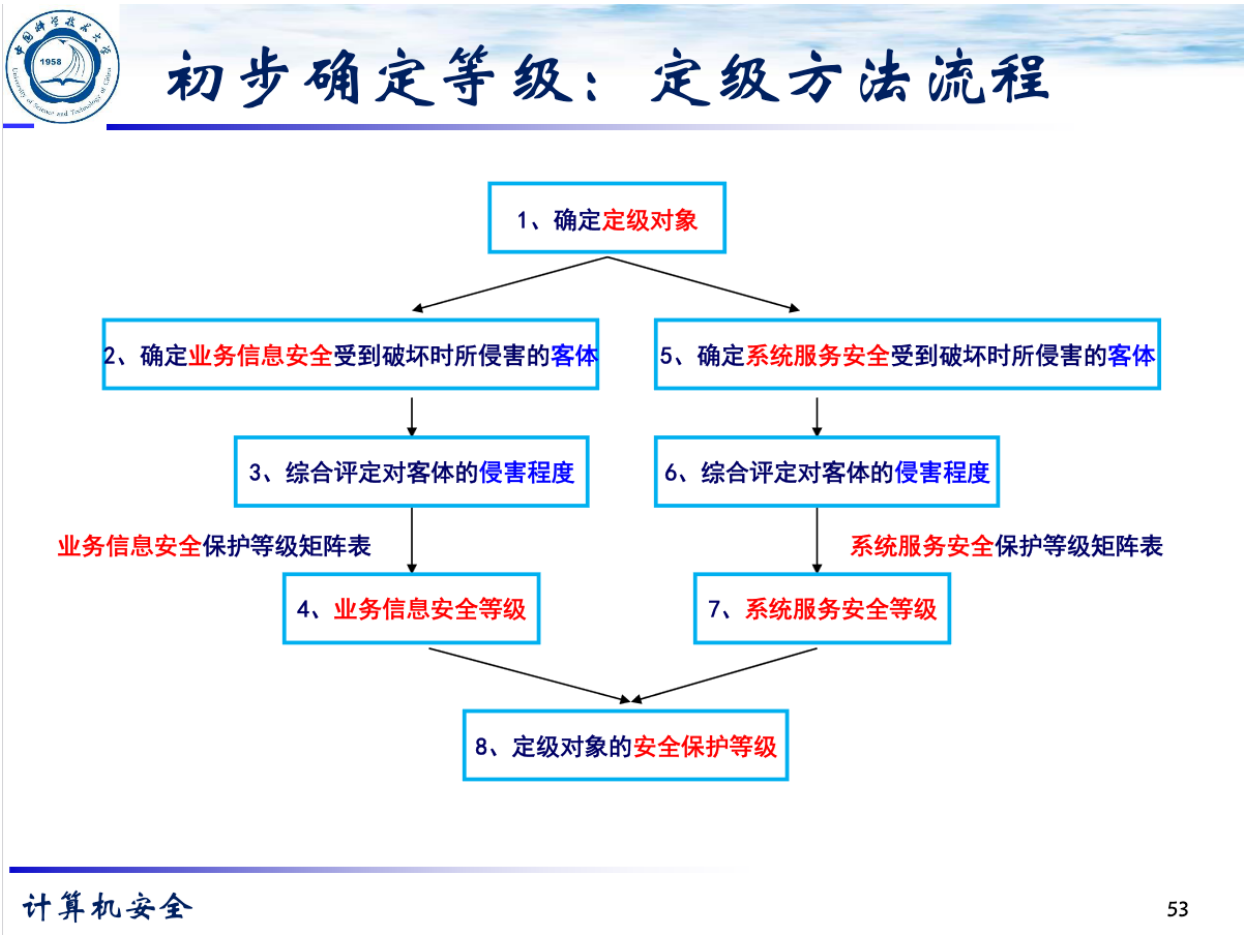
- 技术要求
  - 安全物理环境
  - 安全通信网络
  - 安全区域边界
  - 安全计算环境
  - 安全管理中心
- 管理要求
  - 安全管理制度
  - 安全管理机构
  - 安全管理人员
  - 安全建设管理

- 安全运维管理

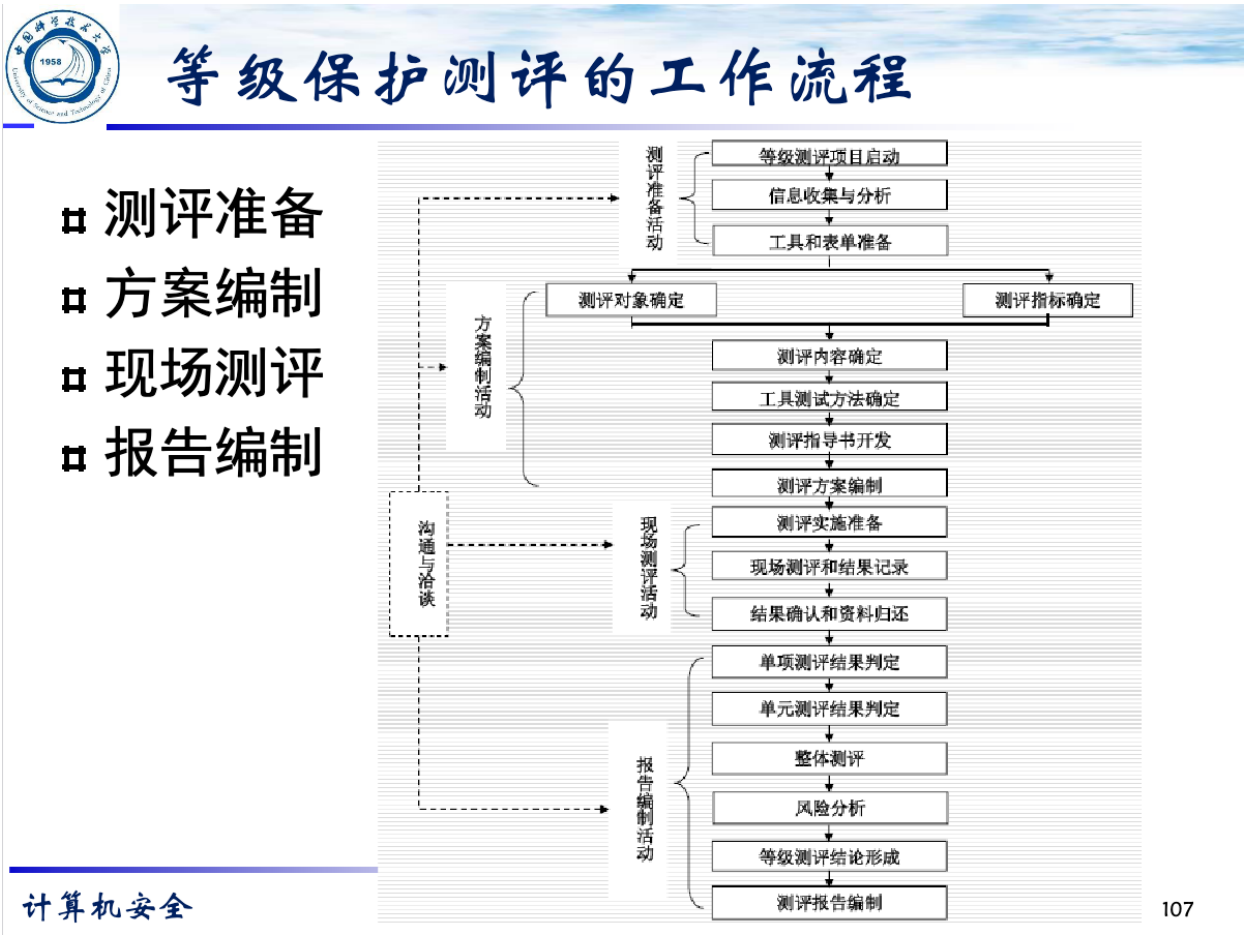
等级保护的主要工作

- 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



等级保护测评的工作流程



Chapter 13+ 密码应用安全性评估

密码应用安全性评估的含义

- 对 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

密码应用基本要求的八大安全类

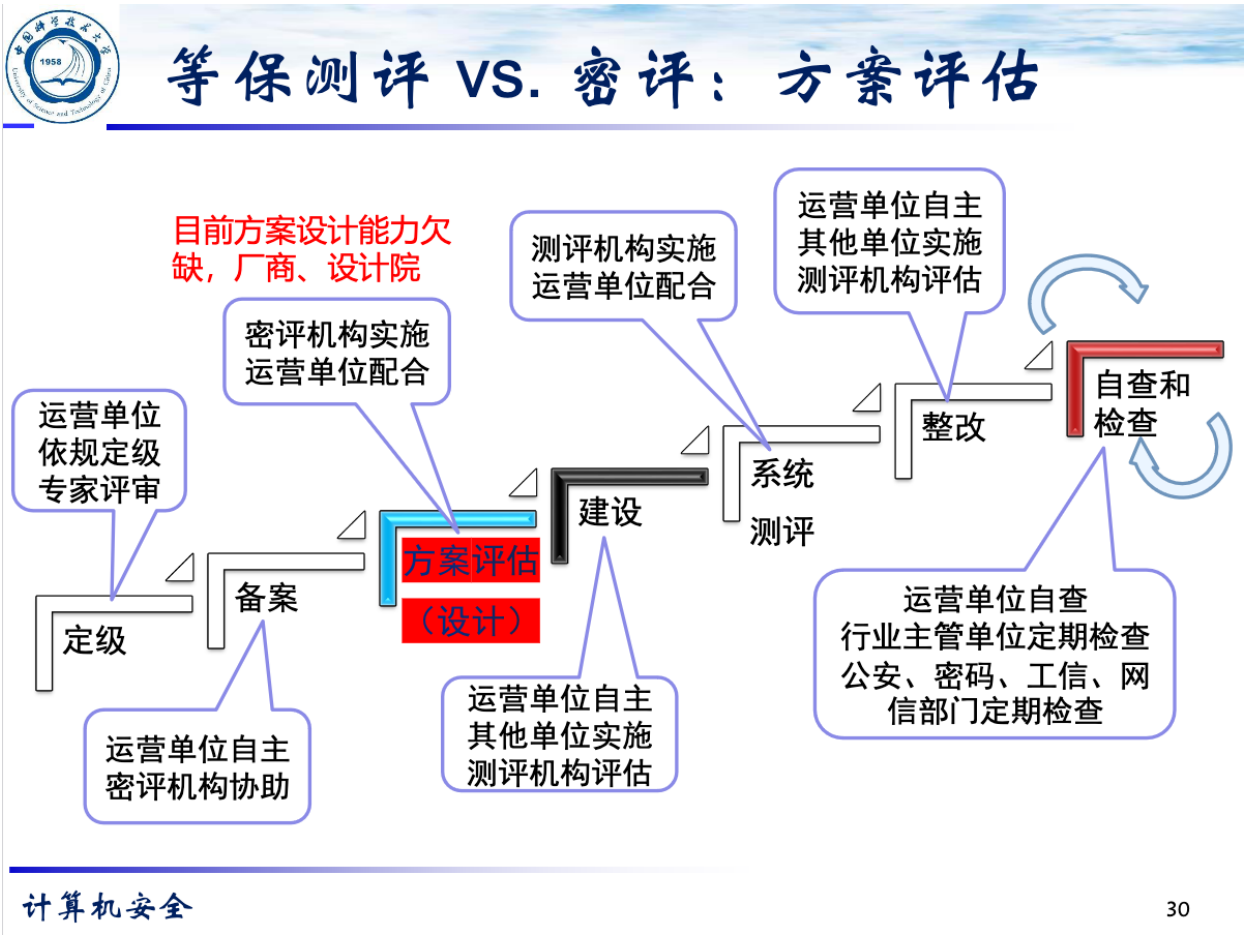
- 技术要求
  - 物理和环境安全
  - 网络和通信安全
  - 设备和计算安全
  - 应用和数据安全
- 管理要求
  - 管理制度
  - 人员管理
  - 建设运行
  - 应急处置

技术标准中“应”“宜”“可”的区别

- 应：要求型描述
  - 必须遵循相关要求

- 宜：推荐型描述
  - 通过密评的密码应用方案决定，如不遵循说明原因以及替代性安全措施
- 可：陈述型描述
  - 自主选择

等保和密评的区别



Chapter 14 云计算安全

云计算的主要特性

- 按需自主服务
- 泛在接入
- 资源池化
- 快速伸缩性
- 服务可计量

云计算的服务模式

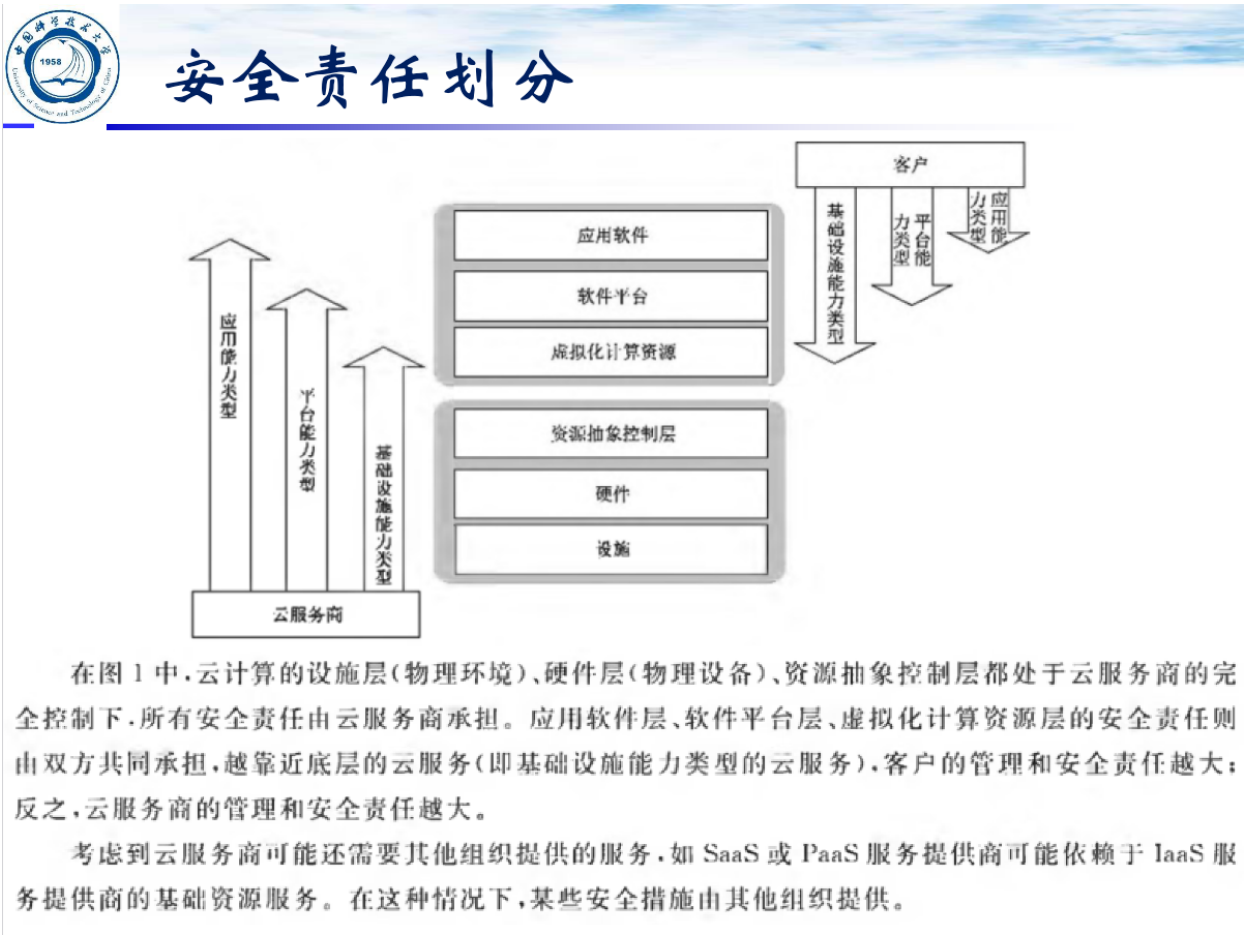
- SaaS
  - 云服务商给用户提供运行在云基础设施上的**应用软件**
  - 客户不需要购买、开发软件
  - 客户不能管理或控制支撑应用软件运行的底层资源

    - 客户可以有限的配置管理
    - 基础软件
        - 管理工具型 SaaS
        - 业务管控型 SaaS
        - 交易平台型 SaaS
        - 个人应用型 SaaS
- PaaS
    - 云服务商给客户提供运行在云基础设施上的**软件开发和运行平台**
    - 客户可以利用平台开发和部署软件
    - 可对应用的运行环境进行配置
    - 基础软件
        - GAE Google App Engine
        - Windows Azure
        - BAE Baidu App Engine
        - SAE(Sina)
        - ACE Aliyun Cloud Engine
- IaaS
    - 云服务商给客户提供**虚拟计算机、存储、网络等计算资源以及云基础设施的服务接口**
    - 客户可以部署或运行操作系统、中间件、数据库和应用软件
    - 基础软件
        - OpenStack
        - Amazon EC2
        - GCE Google Compute Engine

云计算的部署模式

- 私有云
    - 给某个特定用户
    - 场外/外包、场内/自有
- 公有云
- 社区云
    - 特定客户群体
    - 群体中客户有共同属性
    - 场外、场内
- 混合云

云服务商和客户之间的安全责任划分



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。

Chapter 15 基于代码的访问控制

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
- 有效许可：调用栈熵所有函数许可的**交集**
- **惰性计算**
  - 只有当访问一个资源需要一个许可时采取评估授予的许可

许可断言

- 附着在当前栈帧熵
- 堆栈遍历在遇到许可断言的帧并授予这个许可时终止
- 断言允许不可信的调用者调用可信的方法

热情计算

- 前摄性 估算调用者的许可
- 加载时**静态许可** S 被关联到代码的每一部分
- 每一个执行单元相关联的**当前许可** D 在规则 $D = D \cap S$ 下自动更新
- 使用 `grant` 和 `accept` 进行可控修改

Chapter 16 入侵检测

IDS 入侵检测系统

- 信息收集
  - 系统 网络 数据 用户活动 的 状态和行为
  - 不同网段不同主机收集信息
- 信息分析
  - 模式匹配
  - 统计分析
  - 完整性分析（往往用于事后分析）
    - 主要关注某个文件或对象是否被更改
- 结果处理
  - 警报
  - 通告
  - 应急响应

IPS 入侵防御系统

- IDS 和防火墙联动
- 问题
  - 接口规范
  - 瞬间攻击
- IPS 是对防病毒软件和防火墙的补充
- 能够即时的中断、调整或隔离不正常的行为

DPI 深度包检测

- 基于应用层的流量检测和控制技术
- 深度读取 IP 包载荷来对应用层信息进行重组，按照系统定义的管理策略对流量进行操作
- DPI 识别技术
  - 特征字识别技术 -- 协议指纹
  - 应用层网关识别技术 -- 控制流和业务流分离
  - 行为模式识别技术

DFI 深度/动态流检测

- 基于流量行为的应用识别技术
- 不同应用，会话连接或数据流上的状态不同
- 建立流量特征模型
  - 分析包长、连接速率、传输字节量、包与包之间间隔

态势感知

- 基于环境的、动态、整体的洞悉安全风险的能力

    - 以安全大数据为基础
    - 全局视角 提升 发现识别、理解分析、响应处置能力
    - 决策与行动，安全能力的落地

入侵检测性能关键参数

- 误报
- 漏报

入侵检测方法分类

- 异常检测模型
    - 总结正常操作应具有的特征 -- **用户轮廓**
    - 前提：入侵是异常活动的子集
    - 漏报率低，误报率高
    - 效率取决于
        - 用户轮廓的完备性
        - 监控的频率
    - 能有效检测未知的入侵
- 误用检测模型
    - 收集非正常操作的行为特征建立**攻击特征库**
    - 前提：所有入侵行为都有可被检测到的特征
    - 过程：监控 $\rightarrow$ 特征提取 $\rightarrow$ 匹配 $\rightarrow$ 判定
    - 误报低、漏报高
    - 攻击特征有变化就寄

入侵检测来源分类

- 基于主机
- 基于网络
- 混合型

Chapter 17 应急响应与灾备恢复

IATF 信息保障技术框架

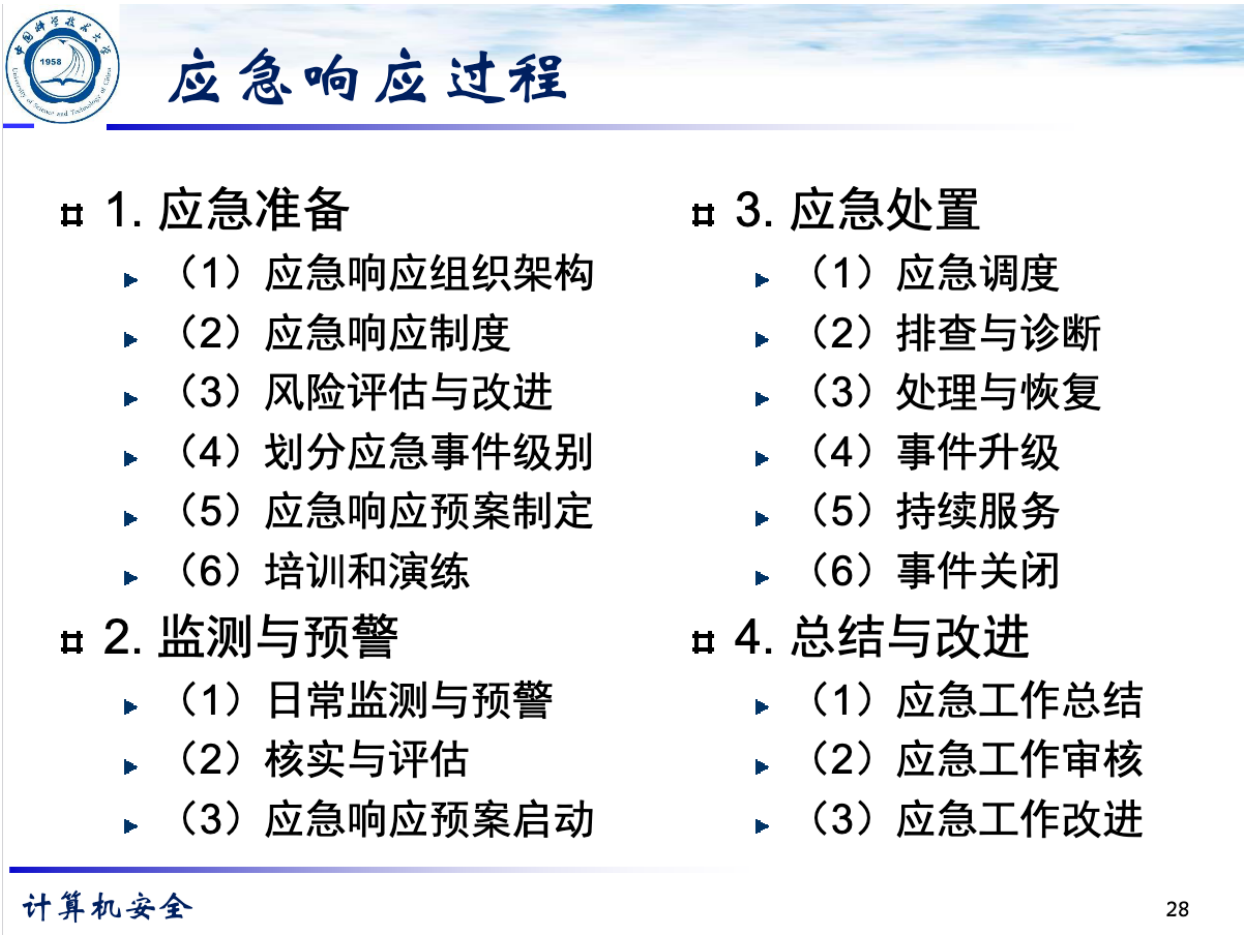
- 一个核心思想
    - 纵深防御（深度防护）
- 三个核心要素
    - 人（管理）-- 核心
    - 技术 -- 重要手段
    - 操作（运维）-- 主动防御体系
- 四个焦点领域（信息安全保障区域）

- 网络和基础设施
- 区域边界
- 计算环境
- 支撑性基础设施

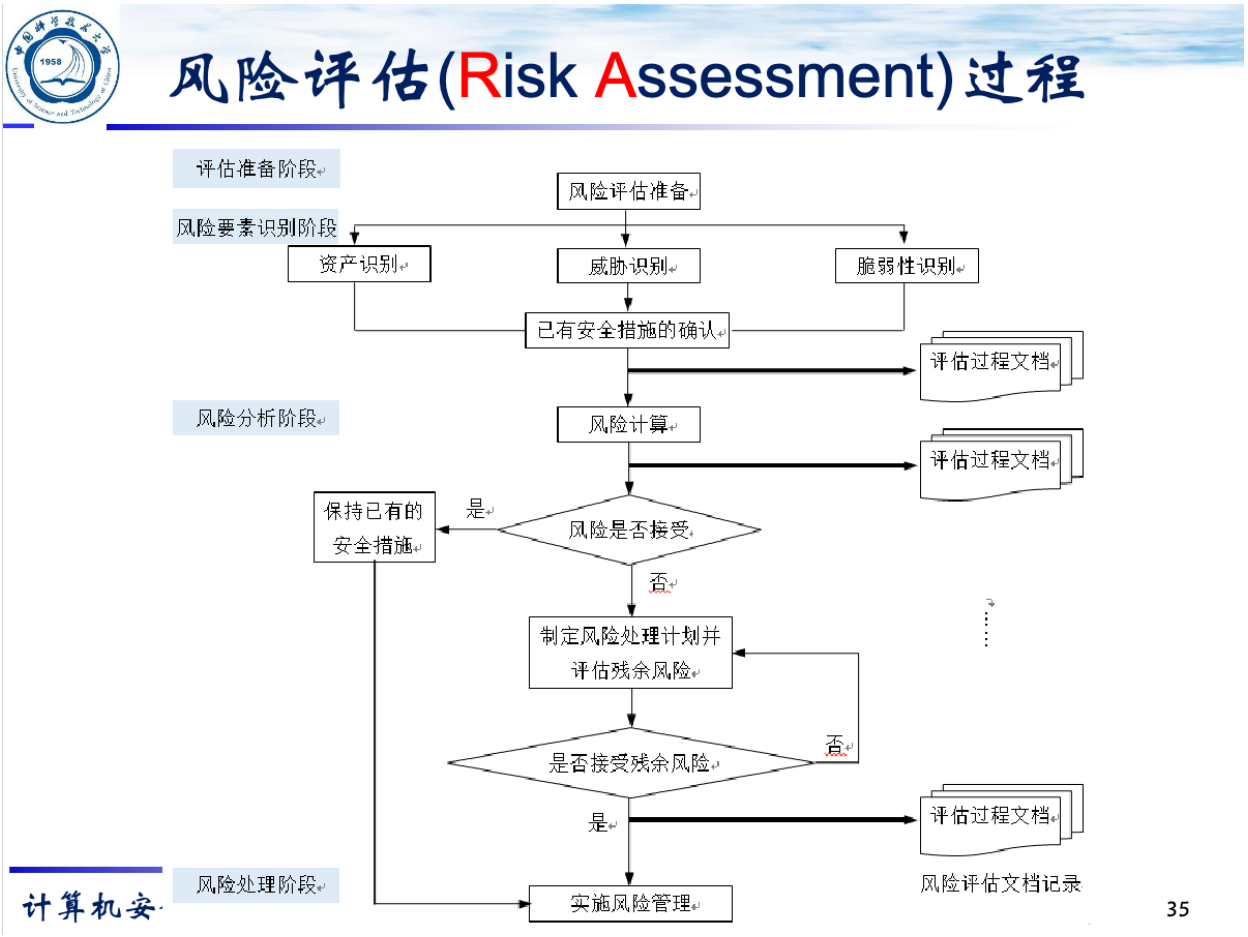
应急响应概念

- 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事件发生后采取的错书
- 目的：金肯呢个减少和控制安全事件的损失
- 重点：预防，而不是应急

应急响应过程



风险评估过程



灾难恢复的概念

- 为了将信息系统从在闹造成的故障状态恢复到正常，而设计的**活动和流程**

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.....在灾难发生后可以恢复的过程

RAID 独立冗余磁盘阵列

- RAID 0
  - 没有冗余或错误修复能力
- RAID 1
  - 磁盘镜像
  - 磁盘利用率 50%，成本最高
  - 多用于保存关键性的重要数据
- RAID 10
  - 镜像阵列条带，RAID 0+1
  - 不考虑价格，提供最好的性能
- RAID 5
  - RAID 0 和 RAID 1 的折衷方案
  - 奇偶校验信息
  - 存储性能、数据安全和存储成本兼顾的解决方案

- 至少 3 块硬盘
- 只允许有一块硬盘出现故障

RAID总结

类型	读写性能	安全性	磁盘利用率	成本	应用方面
RAID0	最好（因并行性而提高）	最差（完全无安全保障）	最高（100%）	最低	个人用户
RAID1	读和单个磁盘无分别，写则要写两边	最高（提供数据的百分之百备份）	差（50%）	最高	适用于存放重要数据，如服务器和数据库存储等领域。
RAID5	读：RAID 5 = RAID 0（相近似的数据读取速度） 写：RAID 5<对单个磁盘进行写入操作（多了一个奇偶校验信息写入）	RAID 5	RAID 5>RAID 1	RAID 5	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。
RAID10	读：RAID10 = RAID0 写：RAID10 = RAID1	RAID10 = RAID1	RAID10 = RAID1（50%）	RAID10 = RAID1	集合了RAID0，RAID1的优点，但是空间上由于使用镜像，而不是类似RAID5的"奇偶校验信息"，磁盘利用率一样是50%