第8章　操作系统安全

南京大学计算机系　黄皓教授

2011年12月5日

参考文献

- Charles P. Pfleeger, Shari Lawrence Pfleeger. 李毅超等译。信息安全原理与应用。电子工业出版社，2004年7月第1版。
- 刘克龙等，安全操作系统原理与技术，科学出版社，2004年7月第1版。
- 卿斯汉等，操作系统安全，清华大学出版社，2004年8月。

内容

1. 可信操作系统研究的发展
2. 操作系统面临的威胁
   - 控制流篡改
   - 关键数据篡改
   - 隐蔽信道
3. 可信操作系统的基本概念
   - 安全功能与安全保障
   - 可信与不可信
   - 引用监视器
   - 安全核
   - 可信计算基
4. 可信操作系统的安全机制与安全功能
   - 硬件安全机制
   - 标识与鉴别
   - 访问控制
   - 可信通道
   - 安全审计
   - 最小特权管理
5. 可信操作系统的保障机制
   - 形式化的描述与验证
   - 入侵检测

1. 可信操作系统研究的发展

萌芽与访问控制抽象

- 1969年Weissman C. 发表了有关ADEPT-50安全控制的研究成果。 ADEPT-50运行于IBM/360硬件平台。
- 高水印模型(high-water-mark model):
  - 为客体标上敏感级别(sensitivitylevel)属性。
  - 对于读操作，不允许信息的敏感级别高于用户的安全级别clearance)。
  - 对于写操作，在授权情况下，允许使信息从高敏感级别移向低敏感级别。

Weissman C., Security Controls in the ADEPT-50 Time Sharing System。 Proceedings of the1069 AFIPS Fall Joint Computer Conference, AFIPS Press, 119-133.

访问矩阵(access matrix)

- Lampson B.W. 通过形式化表示方法，运用主体(subject)、客体(object)和访问矩阵(access matrix)的思想，第一次对访问控制问题进行了抽象。
  - **Lampson B.W., Dynamic Protection Structures. In Proceedings of the AFIPS, Fall Joint Computer Conference, volum 35, Las Vegas, Nevada, 27-38. Nov. 1969.**
- 访问矩阵是以主体为行索引、以客体为列索引的矩阵，矩阵中的每一个元素表示一组访问方式，是若干访问方式的集合。
- 矩阵中第i行第j列的元素M_{ij},记录着第i个主体S_i，可以执行的对第j个客体O_j的访问方式，比如M_{ij}，等于(read,write)表示S_i可以对O_j 进行读和写访问。

多级安全（Multilevel security system）

Ware W.H., Security Controls for Computer Systems(U)， Report of Defense Science Board Task Force on Computer Security. Technical Report, the Rand Coporation, Santa Monica, CA, published for the Office of the Director of Defense Research and Engineering, Washington, D.C.。

- Ware W.H.推出的研究报告对多渠道访问的资源共享的计算机系统引起的安全问题进行了研究。
- 结合实际的国防信息安全等级划分体制，提出了解决计算机安全问题的建议途径。
- 研究的主要目标是多级安全系统(multi-level security system)在计算机中的实现。
- 对计算机安全系统的设计提出了两个限制条件:
  - 计算机安全系统必须与现实的安全等级划分结构一致。
  - 计算机安全系统必须与现实的手工安全控制规程相符。

多级安全（Multilevel security system）

- 安全控制措施的特点
  - 系统灵活性：在应用中可调整个体和信息的安全等级
  - 可靠性：贯彻“失败-保险”思想，当不能确定是否授权时，采取不授权的措施)
  - 可审计性：记录安全相关行为人可管理性(安全控制、审计控制等管理)
  - 可依赖性：避免拒绝对用户的服务
  - 配置完整性：确保系统自身的完整
- 报告认为计算机系统的安全控制是个系统设计问题，必须从硬件、软件、通信、物理、人员和行政管理规程等各个方面综合考虑。

安全评价标准

- 1983年，美国国防部颁布了历史上第一个计算机安全评价标准，这就是著名的可信计算机系统评价标准，简称**TCSEC**，又称橙皮书。1985年，美国国防部对TCSEC进行了修订。
- TCSEC标准是在基于安全核技术的安全操作系统研究的基础上制定出来的，标准中使用的可信计算基(Trusted Computing Base，**TCB**)就是安全核研究结果。

Nibadi G.H., Specification of a Trusted Computing Base, M79-228, The MITRE Corporation, Bedford, MA, USA。

- 给出了TCB的定义，该方法要求把计算机系统中所有与安全保护有关的功能找出来，并把它们与系统中的其他功能分离开，然后把它们独立出来，以防止遭到破坏，这样独立出来得到的结果就称为TCB。

国外安全操作系统

- 1984年， AXIOM公司的Kramer S.发表了 LINUS IV 。4.l BSD Unix为原型，结合TCSEC标准的要求，对安全性进行了改造和扩充。
  - 身份鉴别
  - 自主访问控制
  - 强制访问控制
  - 安全审计
  - 超级用户特权分离

安全XENIX系统开发

- 1986年， IBM公司的Gligor V. D.等发表了安全XENIX系统设计与开发成果。
- 实现了TCSEC B2-A1级的安全要求。
- 它采用改造法，对原有的XENIX内核进行改造和扩充，使它支持新的安全策略。

System V / MLS

- 1988年AT&T Bell实验室的Flink II C.W. 和Weiss发表了System V / MLS 系统的设计与开发成果。
- 以Unix System V为原型，以TCSEC B级为设计目标。

安全TUNIS

- 1989年加拿大多伦多大学的Grenier G.L., Holt R. 和Funkenhauser M. 发表了安全TUNIS系统。
- TUNIS(Toronto UNIversity System)是加拿大多伦多大学开发的一个与Unix 兼容的操作系统，用强类型的Turing Plus高级语言编写。
- 安全TUNIS是以TUNIS为原型的安全操作系统。
- Grenier等指出，如果不进行系统的重新设计，以传统的Unix系统为原型，很难开发出高于TCSEC标准的B2级的安全操作系统，主要原因是C语言不是安全语言，Unix的内部模块化程度不够。
- Turing Plus是一个可验证语言，安全TUNIS的目标是B3-A1级。

ASOS

- 1990年TRW公司的Waldhart N.A. 和Di Vito B. L.等发表了ASOS系统的设计与开发成果。
- ASOS(army secure operating system)是针对美军的战术需要而设计的军用安全操作系统。
- 从1984-1986，ASOS项目的过渡时期，产生了一个运行在MCF(Military Computer Family)计算机原型上的DS操作系统原型。
- 1986-1989， ASOS项目组开发出了MLS（A1)和DS（C2)两个操作系统。
- ASOS项目在形式化验证中建立了两个层次的描述和证明，一个层次用于抽象的安全模型，一个层次用于形式化的顶层描述。

多策略与信息域

- 一个信息域由一些信息对象、一些用户和一个信息安全策略构成。
- 支持多种安全策略的系统将拥有与安全策略数量一样多的信息域，这样的系统有能力处理数量非常多的信息域。
- 每一个信息对象落在一个信息域中，受所落信息域的信息安全策略控制。
- 每个信息域可以拥有自己的信息安全策略，几个信息域也可以共享一个安全策略。
- 安全操作系统应该能够灵活地支持在一个计算平台上同时工作的多种安全策略，应该能够与不同平台上能够支持相同安全策略的其他操作系统进行互操作。

DTOS

- 1997年完成的DTOS(**D**istributed **t**rusted **o**perating **s**ystem)以Mach为基础。
  - 策略灵活性：采用安全判定与安全实施分离支持安全策略的灵活性。
  - Mach兼容
  - 性能接近Mach内核的性能

动态策略

- 从单一策略支持到多种策略支持，安全操作系统迈出了向实际应用环境接近的可喜一步。
- 支持策略灵活性的系统
  - 有能力对执行安全策略控制下的高级功能的低级对象进行细粒度的访问控制
  - 能够确保访问权限的传播与安全策略保持一致
  - 有能力撤回先前已授予的访问权限

Flask体系结构

- Flask是以Fluke操作系统为基础开发的安全操作系统原型。Fluke是一个基于微内核的操作系统。
- DTOS项目之后，Mach微内核的工作没有得到持续的支持，因而NSA和Utah大学合作启动了Fluke保障计划项目，把DTOS安全体系结构集成到Utah大学开发的Fluke操作系统中。同时，对该体系结构进行了改造，后来形成的就是Flask安全体系结构。
- 安全性：主要的安全性目标是在DTOS安全体系结构的基础上建立一个策略灵活的访问控制模型的原型，重点是对动态安全策略的支持。
- 保障能力：保障能力目标是通过运用形式化描述和推理手段实现对关键安全功能的验证。

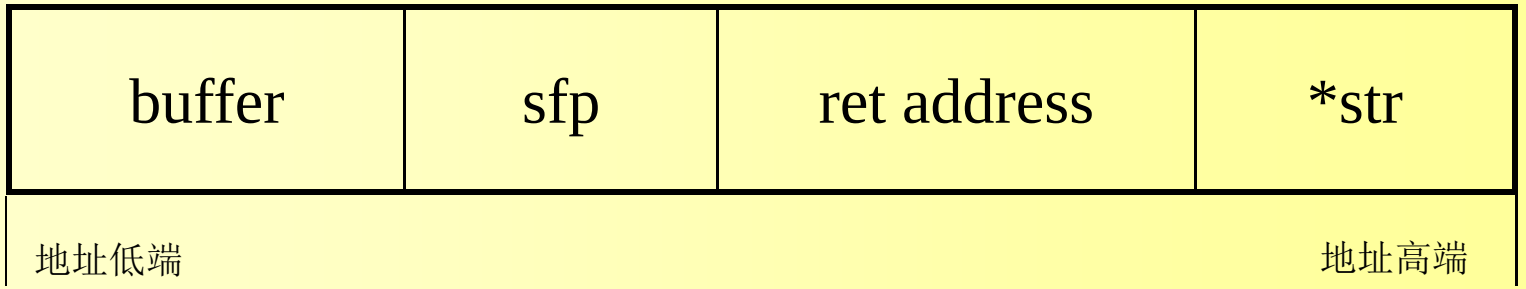
SE-Linux

- 以Linux操作系统为基础，采用Flask安全体系结构。
- Flask是基于微内核的系统原型，Linux是非微内核的操作系统。
- Flask项目完成后，作为Flask系统的主要开发者的美国国家安全安全局(NSA)启动了把Flask的安全体系结构集成到Linux操作系统中的项目
- 网络伙伴公司(NAI)的实验室、安全计算公司(SCC)和MITRE公司等协助NSA完成集成工作。
- NSA已经在Linux内核的主要子系统中实现了Flask安全体系结构，这些子系统包括进程、文件和socket等操作的强制访问控制。
- NAI实验室与NSA合作进一步开发和配置这个安全增强的Linux系统，SCC和MITRE协助NSA开发应用层的安全策略和增强的实用程序。

2. 操作系统面临的威胁

（1） 控制流篡改 — 缓冲区溢出攻击

- int DoSomething(char *str){
  char buffer[BUFFER_LEN];
  strcpy(buffer, str);
  }



缓冲区溢出攻击的危害

- 进程或内核的服务如果存在缓冲区溢出，可被利用来
  - 改变程序的控制流
  - 加载并执行shellcode
  - 修改内核资源
  - 改变执行流程

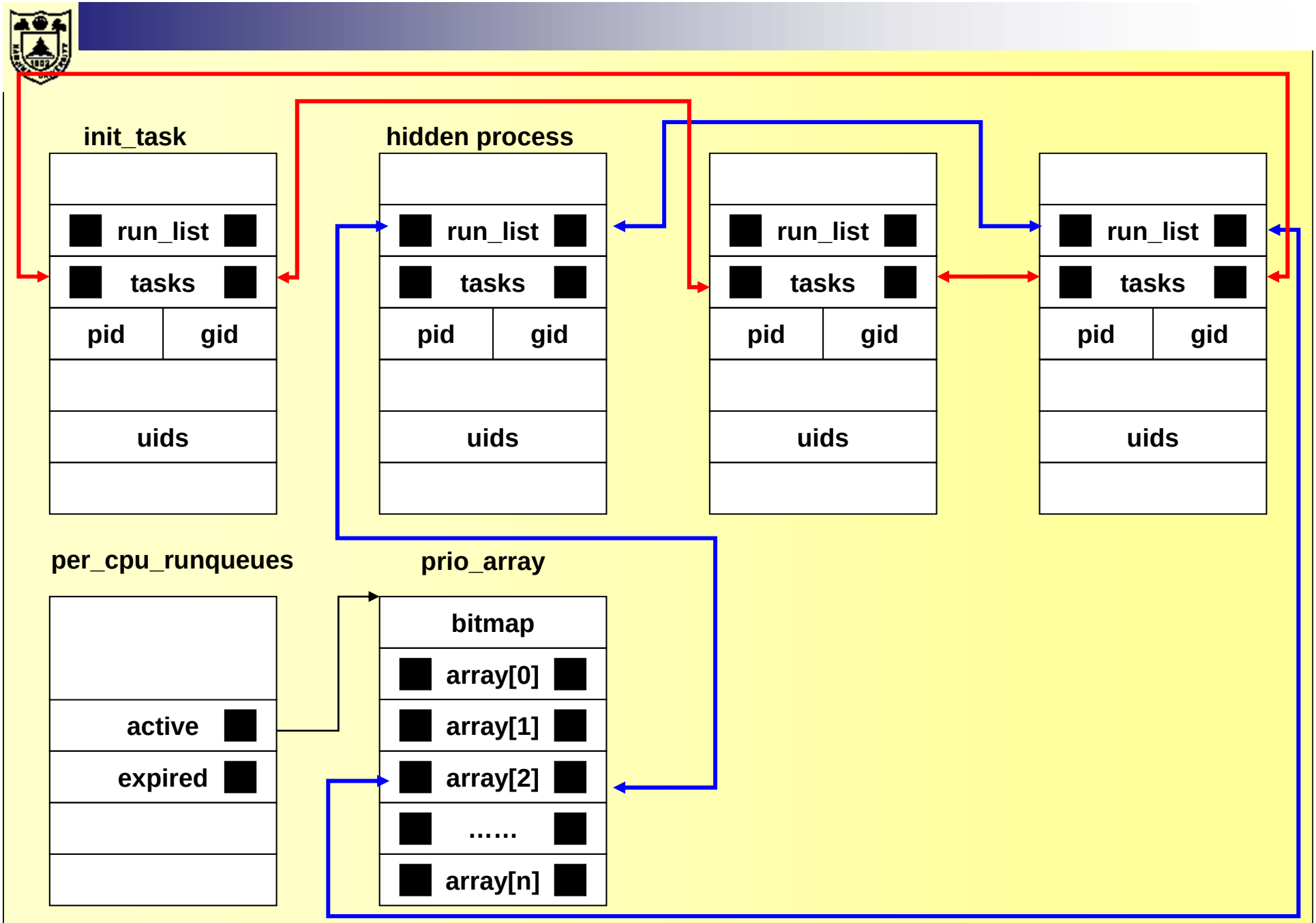
  最终达到：
  - 攻击者能获得非法的权限
  - 攻击者能非法地提升权限；
  - 攻击者可以安装后门；
  - 泄露内核机密信息。

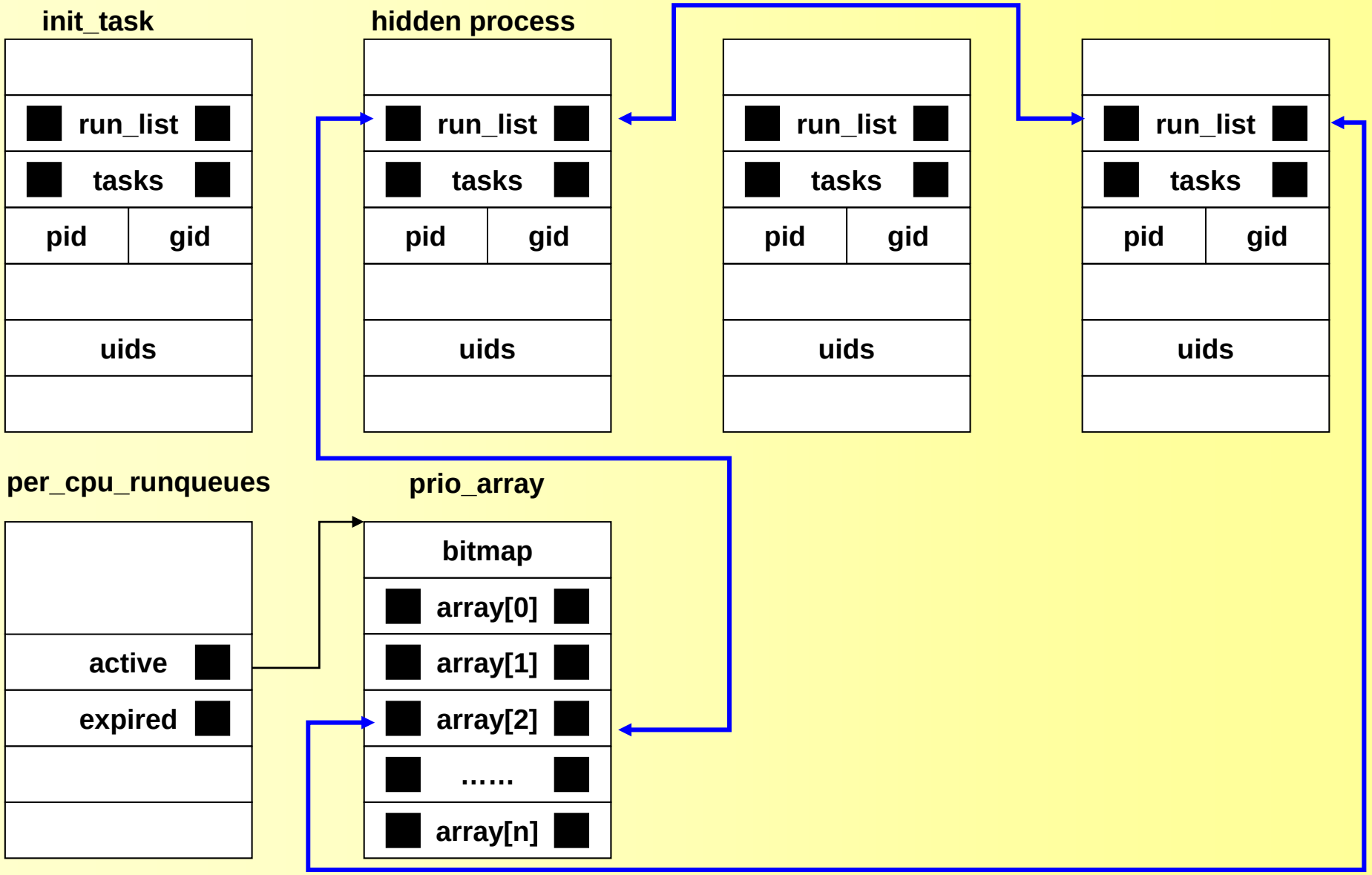
一些缓冲区溢出攻击的例子

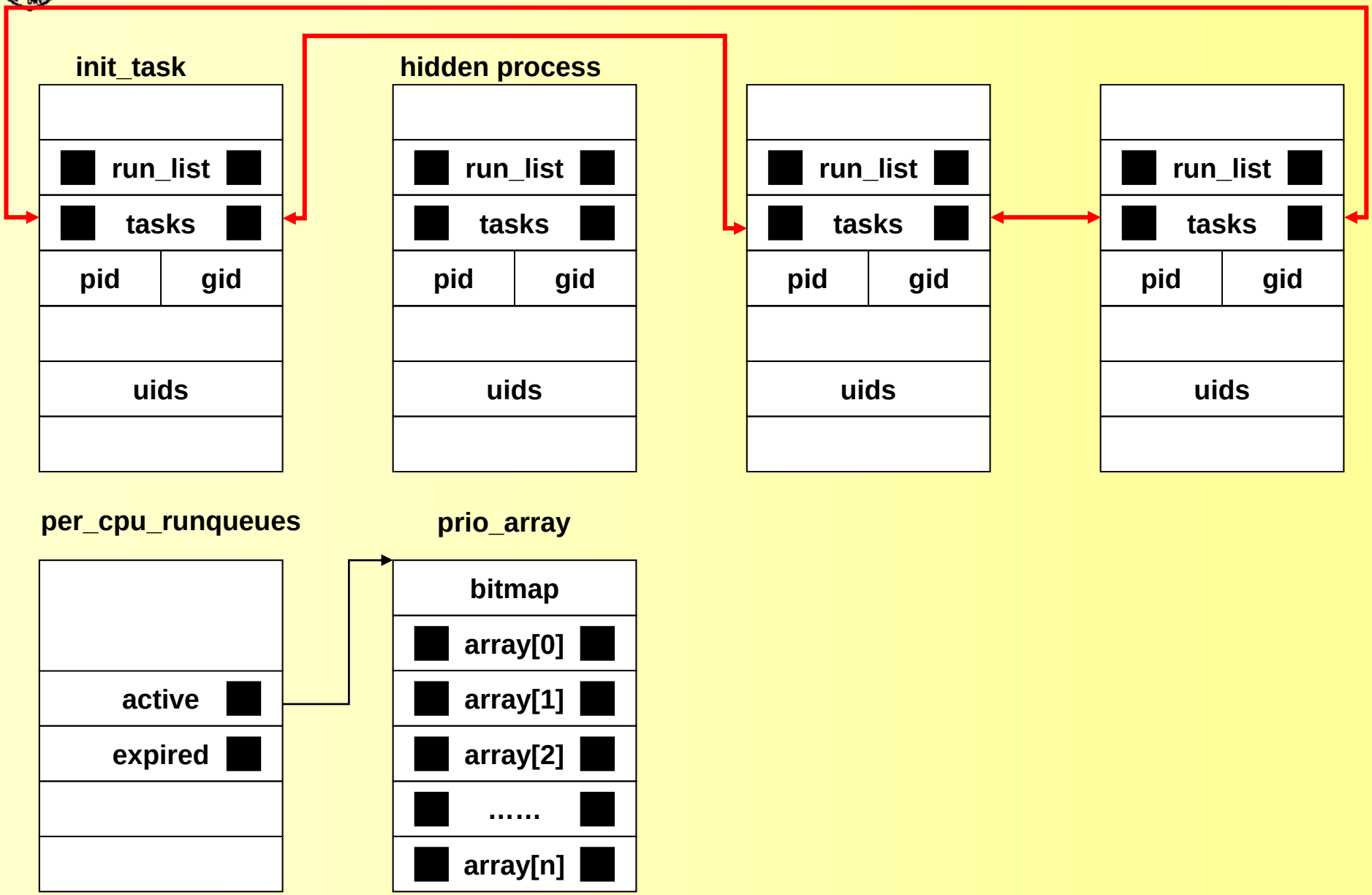
- ping of death
- Netmeeting
- Linuxconf
- IMAPD
- AOL
- Microsoft Windows 2000 ActiveX控件
- IIS4.0/5.0 Phone book 服务器缓冲区溢出。
- SQL Server 2000扩展存储程序缓冲区溢出。

(2) 关键数据篡改 — 进程隐藏的攻击与检测

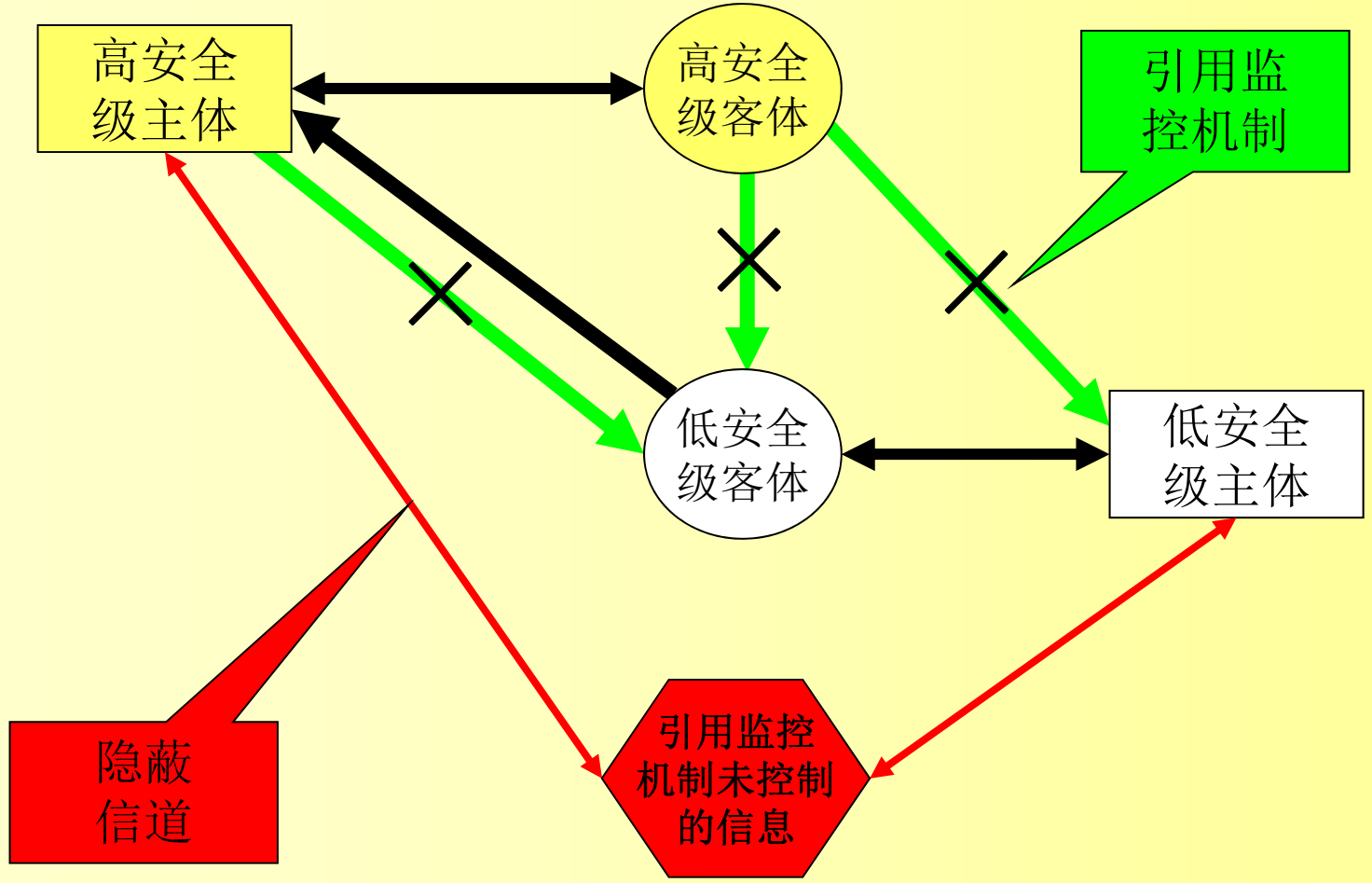
- 修改数据：攻击者可以在不修改内核代码和控制流的情况下隐藏它们的进程。
  - 修改ps所依赖的底层数据，使其无法看到被隐藏的进程。
  - 能够让进程调度程序调度被隐藏的进程。


init_task
hidden process
run_list
tasks
pid
gid
uids
per_cpu_runqueues
active
expired
prio_array
bitmap
array[0]
array[1]
array[2]
……
array[n]


init_task
hidden process
run_list
tasks
pid
gid
uids
per_cpu_runqueues
prio_array
active
expired
bitmap
array[0]
array[1]
array[2]
……
array[n]


init_task
hidden process
run_list
tasks
pid
gid
uids
per_cpu_runqueues
prio_array
active
expired
bitmap
array[0]
array[1]
array[2]
……
array[n]

（3）隐蔽信道



（3）隐蔽信道

- 例：假如两个用户要共享一个系统。这两个用户是分开的，每个用户都有一个自己的虚拟机，并且他们不能直接通信。然而，他们根据负载而共享CPU。
- 如果用户Matt(具有SECRET级别)运行一个需要占用CPU时间较长的程序，用户Holly(具有CONFIDENTIAL级别)运行一个需要占用CPU时间较短的程序，则Matt的程序将占用大部分的CPU时间。
- 这就提供了一条可供Matt和Holly通信的隐蔽信道。
- 他们先协商好共同的开始时间和时间片长度(例如，在中午12：00开始，时间片长度为1分钟)。如果要传输比特1，则Matt就在这个时间片运行他的程序；如果要传比特0，Matt就不在这个时间片运行他的程序。
- 这样，每隔一分钟，Holly尝试运行一个程序，如果这个程序可以运行，则表示Matt的程序没有占用CPU，所以要传输的比特是0；如果这个程序在这个时间片内不能运行，则要传输的比特是1。在这个例子中，虽然没有传统意义上的“写”，但信息却从Matt传到Holly，从而违反了Bell-LaPadula模型的约束条件。

3. 可信操作系统的基本概念

（1） 安全功能

- 安全审计
- 可信通道
- 安全通信
- 标识与鉴别
- 密码服务
- 用户数据保护
- 安全管理
- 私密性
- 系统完整性
- 系统可用性

(2) 安全保障

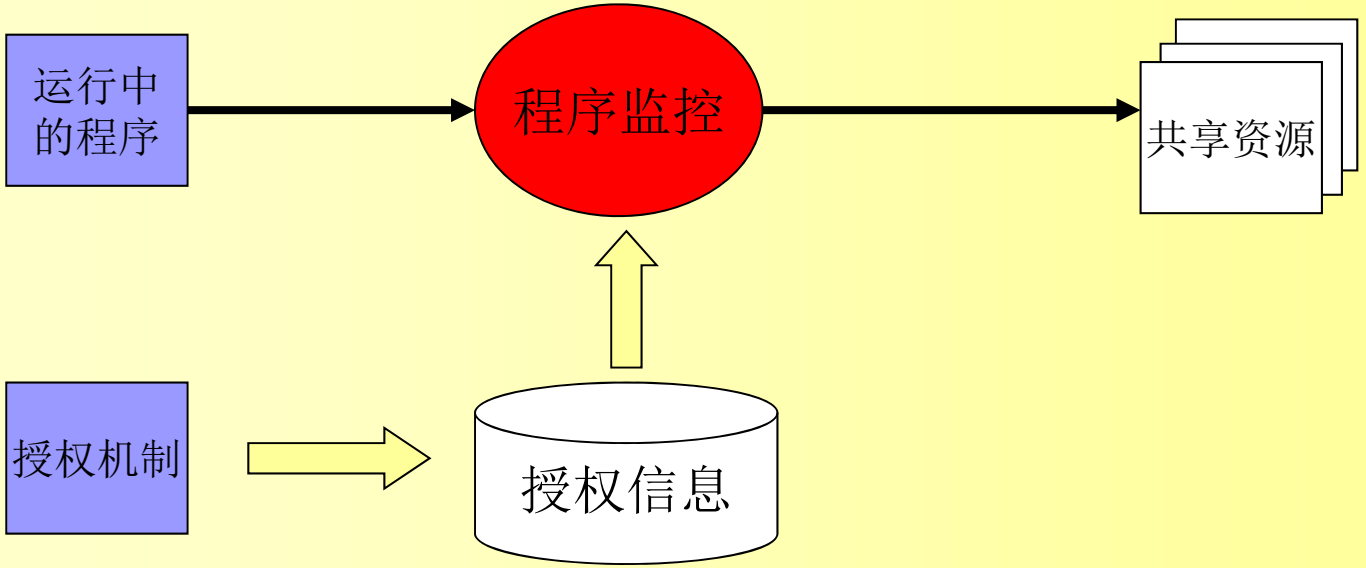
- 配置管理:
  - 配置自动化；版本，配置项，修改授权的控制：配置的覆盖面。
- 交付与运行
  - 交付； 安装，生成，启动。
- 开发
  - 功能规约；高层设计；实现表述；内部结构；底层设计；安全模型；相关性表述。
- 指南
  - 管理员指南，用户指南。
- 生命周期支持
  - 安全开发，错误补救，生命周期定义；工具与技术。
- 测试
  - 测试覆盖；测试深度；功能测试；安全功能满足性测试。
- 脆弱性分析
  - 隐蔽信道测试；误用情况下的安全保障；安全功能的强度；脆弱性分析。

（3） 隔离

- 物理隔离
- 时间分离
  - 例如上午运行非敏感信息任务，下午运行敏感信息任务
- 逻辑分离
  - 操作系统实施了访问控制，程序不能访问许可域之外的对象，使得程序好像在没有其他程序的情况下运行。
- 密码分离

（4）引用监视器（Reference Monitor）

- James P. Anderson, Computer Security Technology Planning Study, Oct.,1972.
- 提出了引用监视器、引用验证机制和安全建模的概念。



（4）引用监视器

- 引用监控器是一个抽象的概念，它表现的是一种思想。Anderson J.P.把引用监控器的具体实现称为引用验证机制，它是实现引用监控器思想的硬件和软件的组合。
- 一个引用监视器是对设备、文件、内存、进程间通信以及其他种类的对象的访问控制的集合。
- 引用验证机制需要同时满足以下三个原则：
  - 必须具有自我保护能力；
  - 必须总是处于活跃状态；
  - 必须设计得足够小，以利于分析和测试，从而能够证明它的实现是正确的。

（4）引用监视器

- 一个引用监视器只有当它不被修改或者一个一个恶意程序绕过的时候才能有效地控制。
- 一个引用监视器是所有访问请求的必由之路。
- 保证实施正确策略的最好办法就是建立一个小而简单的、容易理解的引用监视器。
- 引用监视器不是可信操作系统中惟一的安全机制。
- 引用监视器的概念也被用于一些较小的可信软件。

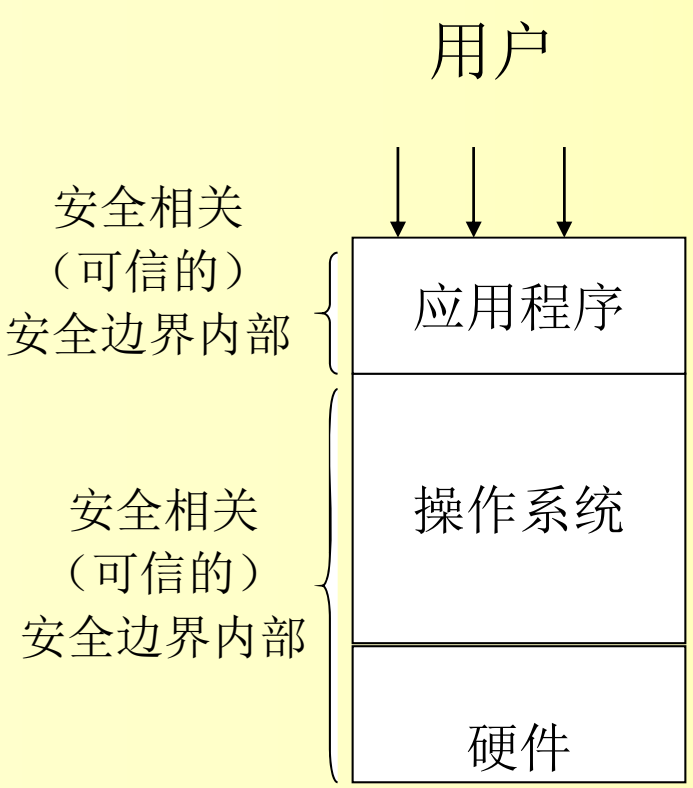
（5）安全内核

- 在受控共享和引用监控器思想的基础上，Anderson J.P.定义了安全核的概念。
- 安全核是系统中与安全性的实现有关的部分，包括
  - 引用验证机制
  - 访问控制机制
  - 授权(authorization)机制
  - 授权的管理机制

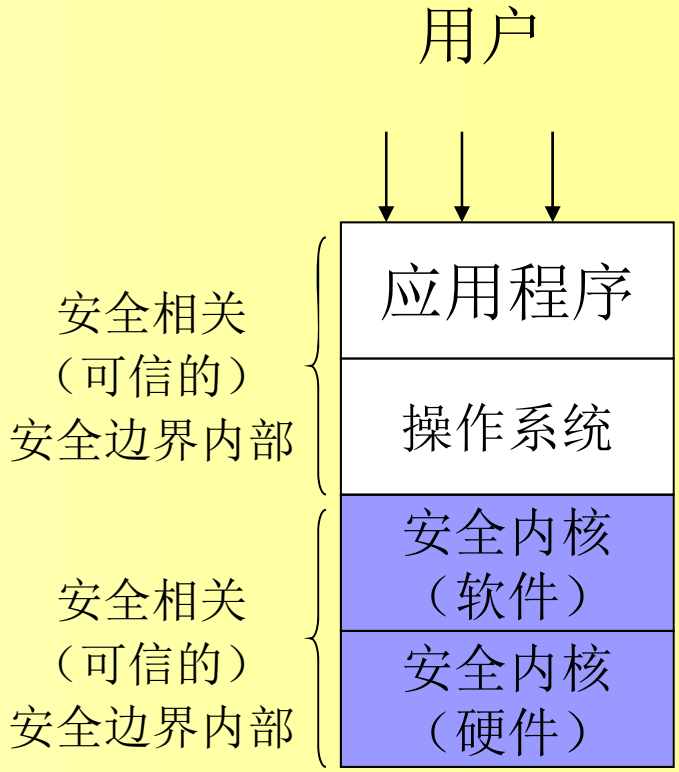
（5）安全内核 — 安全内核的理论依据

- 安全内核是实现引用监视器概念的一种技术；
- 在重新生成操作系统过程中，可用其中安全相关的软件来构成操作系统的一个可信内核，称为安全内核；
- **紧凑性**。在一个大型操作系统中，只有其申的一小部分软件用于安全目的。
- **隔离**。通过将安全机制从操作系统的其余部分和用户空间中隔离出来，容易保护安全机制免遭操作系统或用户的侵入。
- **统一**。所有的安全功能都由单一的代码集合来执行，这样，追踪由于这些功能引起的任何问题的原因就会容易一些。
- **覆盖**。对受保护对象的每次访问必须通过安全内核。绝不能有任何绕过安全内核访问控制检查的访问行为。
- **可验证性**。由于相对较小，安全内核能够被严格地分析。例如，我们能够利用形式化的方法来确保设计覆盖了所有的安全状况。
- **可修改性**。易于对安全机制做出修改和对这种改变进行测试。

（5）安全内核 — 安全内核与操作系统的结构



一般操作系统结构



带有安全内核的操作系统结构

（5）安全内核 — 安全内核的最小化

- 安全内核是一个简单的系统，它为操作系统提供基本服务。安全内核也对操作系统施加限制。
- 操作系统和应用程序的任何错误均不能破坏安全内核的安全策略。
- 安全内核必须做得尽可能小，以便于采用各种方式来有效地增强人们的安全信任度。
  - 在设计时必须坚决贯彻安全内核小型化这一原则
  - 凡不是维持安全策略所必需的功能都不应置于安全内核之中。
  - 虽然在进行安全内核设计时还要考虑诸如性能、使用方便等因素，但这些与小型化要求相比，均居从属地位。

（5）安全内核 — 基本原则

- 完整性原则
  - 完整性原则要求主体引用客体时必须通过安全内核。
  - 完整性原则对支持内核系统的硬件也有一定要求。
    - 内核不检查不可信程序执行的每条机器指令
    - 硬件就必须保证程序不能绕过内核的存取控制。
    - 硬件能保证内核对一般程序在执行：
      - 特权指令
      - 所有对内存、寄存器、输人输出设备的访问

      时，触发内核安全进行合法性检查。
    - 内核必须使各个进程独立，并且保证未通过内核的各进程间不能相互联系。

（5）安全内核 — 基本原则

- 隔离性原则
  - 隔离性原则要求安全内核具有防篡改的能力，即可以保护自己，防止偶然破坏。
  - 在实际实施隔离性原则时常需要软硬件相结合。硬件的基本特性是使安全内核能防止用户程序访问内核代码和数据，这与内核防止一进程访问别的进程是同一种内存管理机制。
  - 防止用户程序执行内核用于控制内存管理机制的特权指令。这需要某种形式的域控制机制，比如保护环机制。
  - 保障用户程序完全没有机会通过写内核的存储器、执行特权指令或修改安全内核的数据等方法使内核受到直接攻击。

（5）安全内核 — 基本原则

- 可验证性原则
  - 利用最新的软件工程技术，包括结构设计、模块化、信息隐藏、分层、抽象说明以及合适的高级语言；
  - 安全内核接口简单化；
  - 安全内核小型化；
  - 代码检查；
  - 安全测试
  - 形式化描述
  - 形式化验证

（5）安全内核 — 安全内核可能的缺点

- 可能降低系统的性能
- 并不能保证包括一切安全功能
- 安全核可能也很大

（6） 可信计算基

- 可信计算基(Trusted Computing Base, **TCB**）

  操作系统的安全所依赖的具体实施的安全策略的软件、硬件和系统安全管理的人员等。

- TCB的组成
  - 操作系统的安全内核。
  - 具有特权的程序和命令，处理敏感信息的程序，如系统管理命令等。
  - 与TCB实施安全策略有关的文件。
  - 其他有关的固件、硬件和设备。
  - 负责系统管理的人员。
  - 保障固件和硬件正确的程序和诊断软件。

(6) 可信计算基 TCB

- 假设你把可信操作系统分为很多部分，有的属于TCB，而有的不是；
- 然后你让技术最高的恶意程序员编写所有不是TCB的部分；
- 由于TCB处理所有的安全，所以恶意的非TCB部分想要削弱TCB的安全策略的正确实施是无能为力的。

(6) 可信计算基 TCB — TCB监视着4个基本的交互

- **进程激活** 在多道程序环境下，进程的激活和钝化是经常发生的。从一个进程切换到另外一个进程需要一个完整的改变，内容包括寄存器、重定位映射表、文件访问表，进程状态信息以及其他指针，它们中的大部分都是安全敏感信息。

- **执行域切换** 在一个域中运行的进程经常会激活其他域的进程以获得更多的敏感数据和服务。

- **内存保护** 因为每个域包含的代码和数据都存储在内存中，因此TUB必须监视内存引用以确保每个域的保密性和完整性。

- **输入／输出(I／O)操作** 在一些操作系统中，软件涉及I／O操作中每一个字符。这种软件将最外层域的用户程序和最内层(硬件)域的I／O设备连接起来。因此，I／O操作能够贯穿所有的域。

(6) 可信计算基 TCB — TCB的设计

- 将操作系统划分为TCB和非TCB
  - 逻辑上 所有和安全相关的代码都被放置在某个部分。
  - 物理上 TCB代码必须运行在一些可以将其区分开的受保护状态下。
- 创建TCB和非TCB的结构
  - 这种结构一旦创建，TCB之外的代码就能够任意被修改，且不会影响TCB实施安全的能力。
  - 操作系统的主要部分(设备驱动器、用户接口管理器等)都能够在任何时间被修改，只有TCB代码需要更加小心的控制。
  - 对于可信操作系统的安全评估来讲TCB和非TCB的划分大大简化了评估，因为不需要考虑非TCB。

4 可信操作系统的安全机制

4 可信操作系统的安全机制

(1) 标识与鉴别

(2) 访问控制

(3) 数据安全保护

(4) 入侵检测

(5) 安全审计

(6) 可信通道

4 可信操作系统的安全机制

(1) 标识与鉴别

(2) 访问控制

(3) 数据安全保护

(4) 入侵检测

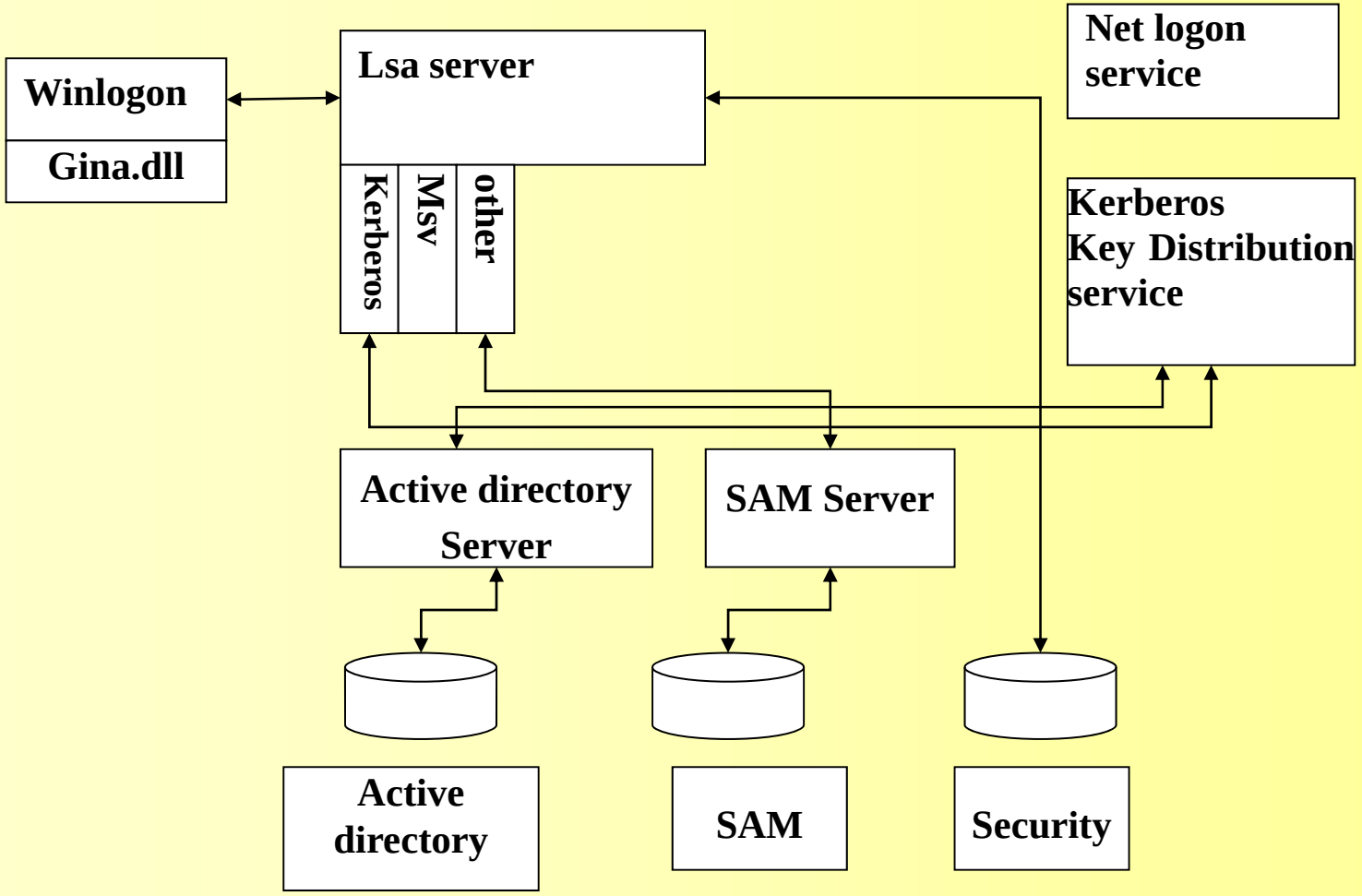
(5) 安全审计

(6) 可信通道

（1）标识与鉴别

- Windows用户认证
- Linux 用户认证：PAM

（1）标识与鉴别— Windows的用户认证



用户名与用户SID

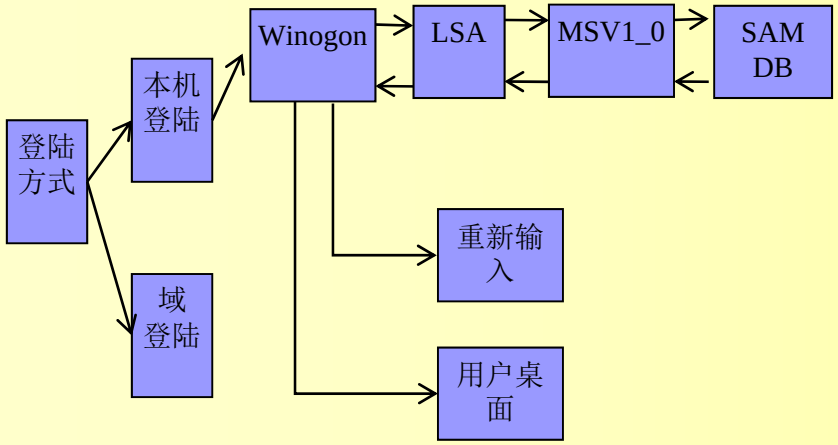
- SID也就是安全标识符（Security Identifiers），是标识用户、组和计算机帐户的唯一的号码。
- 在第一次创建该帐户时，将给网络上的每一个帐户发布一个唯一的 SID。
- Windows 内部进程将引用帐户的 SID 而不是帐户的用户或组名。
- 如果删除帐户，然后使用相同的用户名创建另一个帐户，则新帐户将不具有授权给前一个帐户的权力或权限，原因是该帐户具有不同的 SID 号。
- 当你使用类似Ghost的软件克隆机器的时候，就会产生不同的机器使用一个SID的问题。产生了很严重的安全问题。

（1）标识与鉴别— Windows的用户认证

- Winlogon进程调用gina来获取用户输入的用户名和口令。Gina默认为msgina.dll,可以由第三方提供。
- winlogon通过LPC（本地过程调用）调用lsass来试图认证要登陆的用户 。
- LSASS通过认证包（authentication package）来认证用户，认证包有msv1_0、kerberos和第三方提供认证包 。
- 认证接受用户名和散列后的口令，并与账号数据库进行比较。
  - 账号数据库一般存在于两个地方，如果是本地登陆，则账号数据在SAM储巢（hive）中（\WINDOWS\system32\config\SAM），这是一个注册表的储巢文件，被加载在HKEY_LOCAL_MACHINE\SAM下；
  - 如果是域登陆，则账号数据库在活动目录中（域控制器上的SAM只存放该系统的管理员恢复账号的定义及其口令）。
- 如果认证通过，则创建一个令牌，之后用这个令牌创建userinit.exe，explorer.exe。

Windows用户登陆的流程

- 本地登陆的认证过程



- 访问令牌

令牌源
模仿类型
令牌ID
认证ID
修改ID
过期时间
默认的主组
默认的DACL
用户账户SID
组1 SID
……
组n SID
受限制的SID
……
受限制的SID n
特权 1
……
特权 n

用户口令的破解

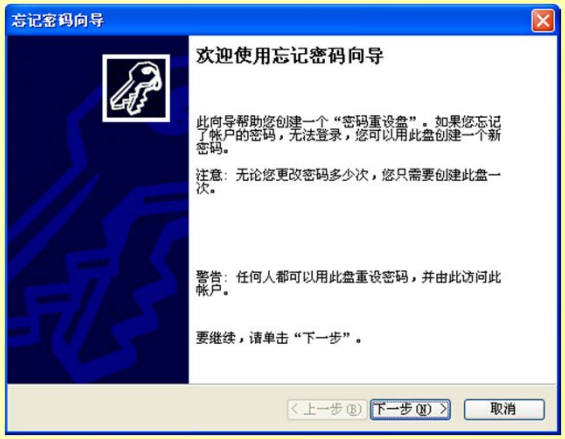
- 攻击者获取SAM 文件
  - SAM 中包含了用户口令的散列值：h=h(password)。
- 攻击者猜测用户可能的口令：p_i，i=1,2, ……
  - 比较 $h(p_i) = h$?
  - 如果相等，则用户的口令是p_i。

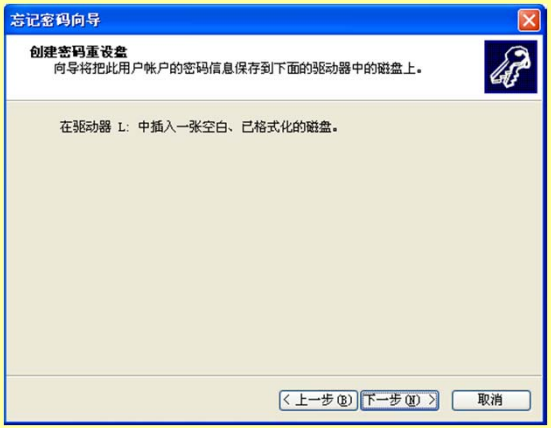
密码启动盘

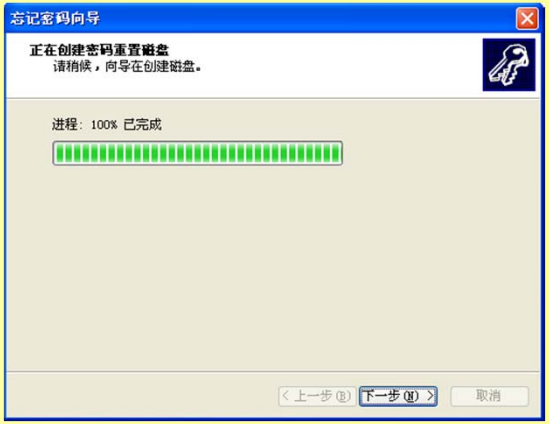
用户账户窗口

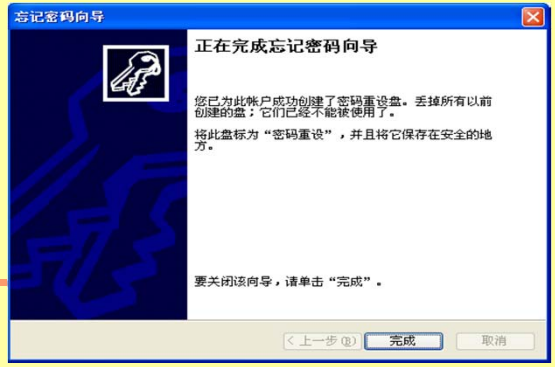
阻止一个已忘记的密码

忘记密码向导









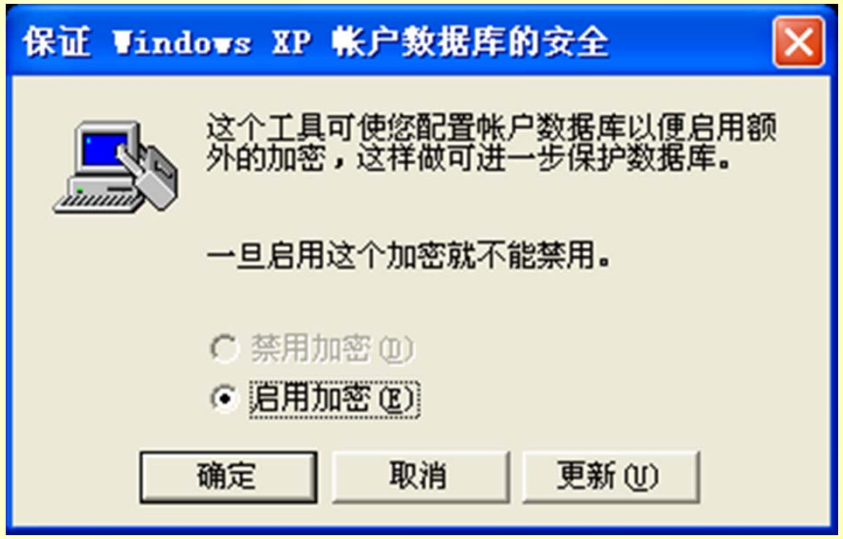
保护安全账户管理器

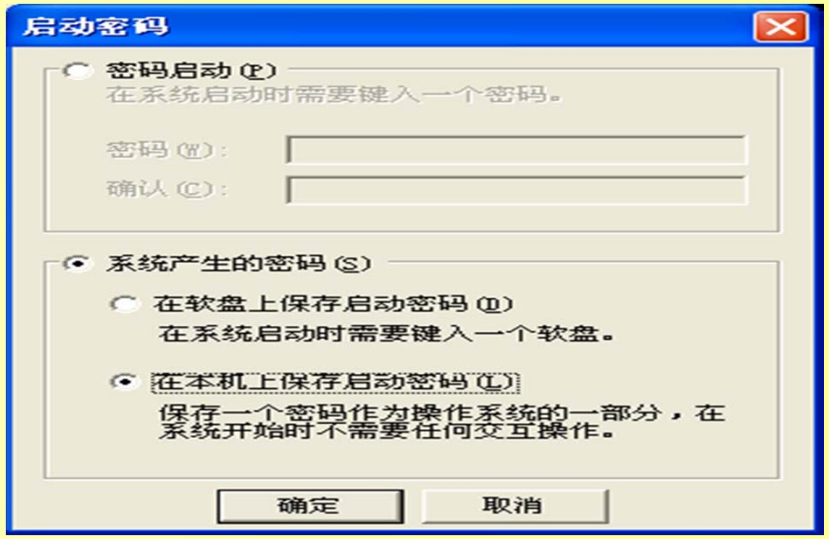
- 安全账户管理器（Security Accounts Manager）
  - 将账户密码信息储存在注册表中。
  - 信息是使用128位密钥进行加密的。
- 攻击者闯入SAM获取加密的密码数值。
- Windows NT4.0 ServicePack3中引入的称为Syskey的工具来加强安全性。SySkey在WindowsXP和Windosws2000中是默认启用的。
  - 用户的账户密码信息由用户账户特有的一个密钥加密。
  - 用户账户特有的一个密钥由主保护密钥加密。
  - 主保护密钥加密由启动密钥加密。

保护安全账户管理器 syskey

- 在命令提示符下输入syskey
- 单击【更新】

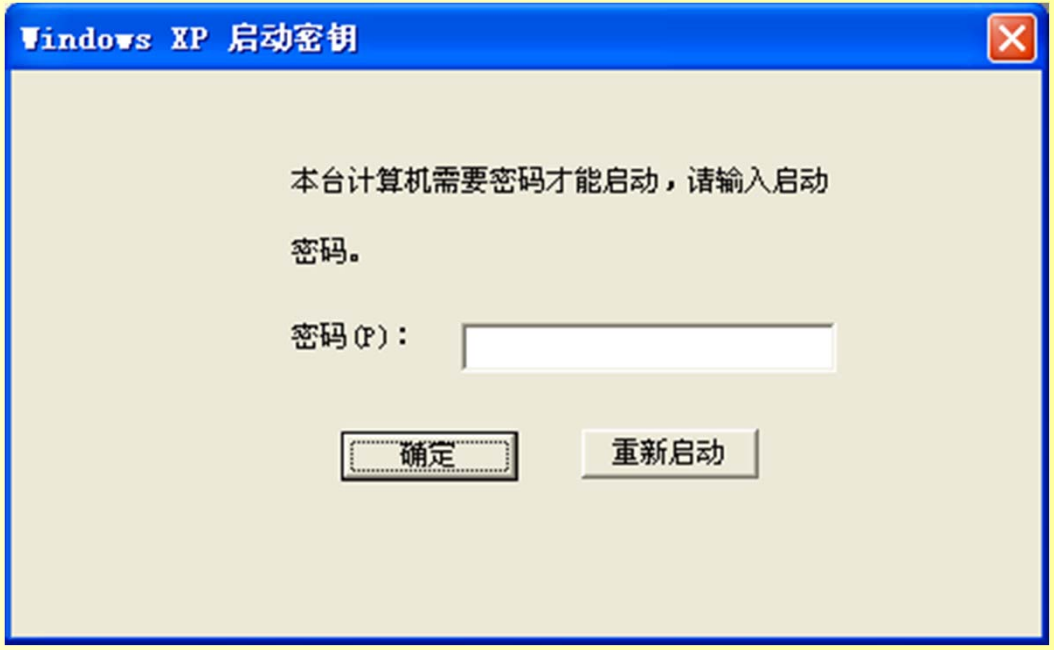
- 密码启动
- 系统产生密码
  - 在软盘上保存
  - 在本机上保存





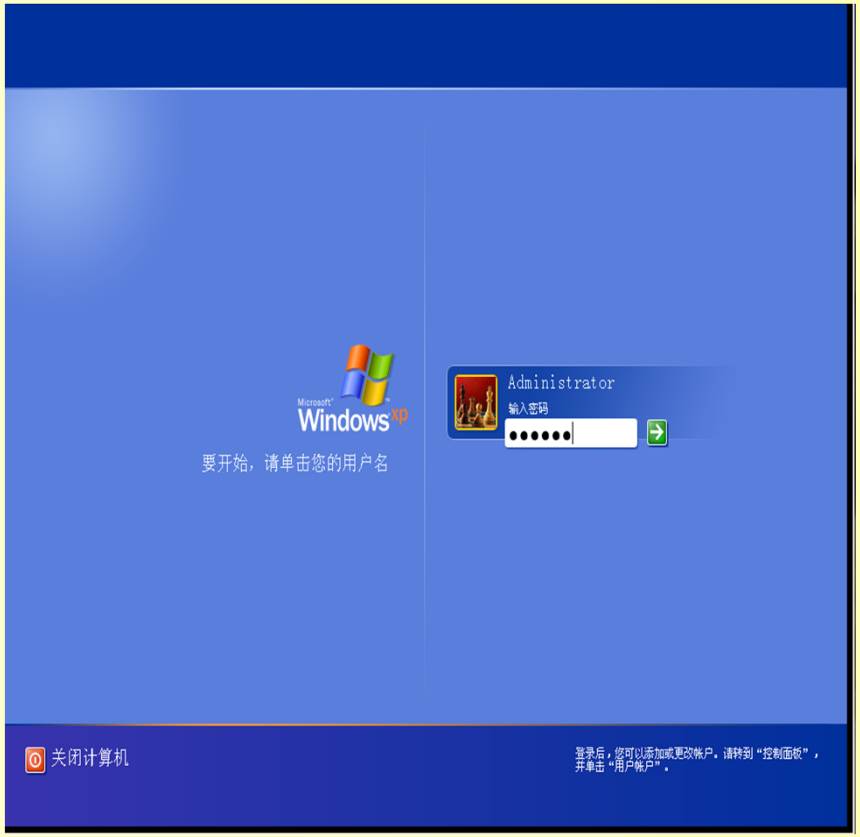
保护安全账户管理器 syskey

- 用户登录时会出现窗口要求用户输入密码。

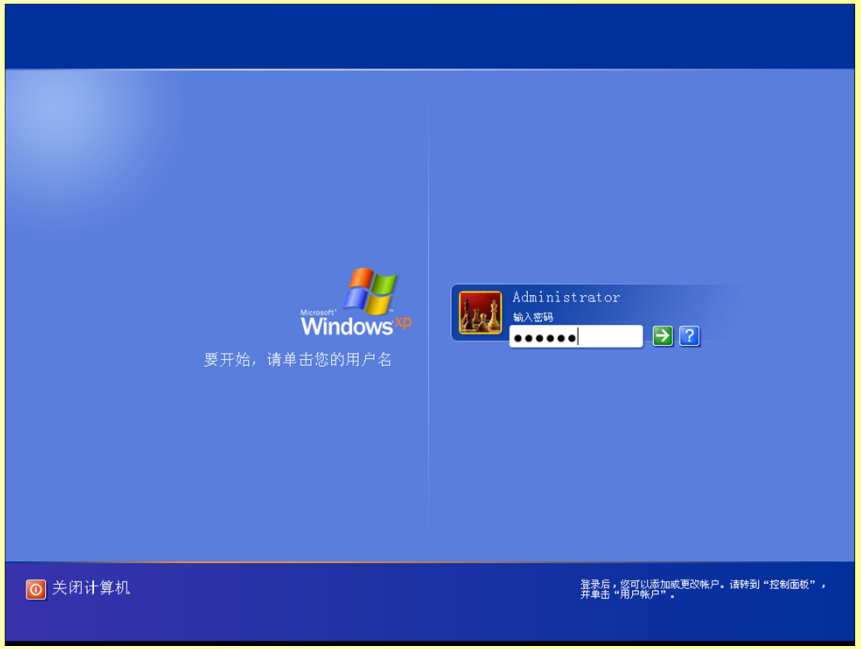


欢迎屏幕

欢迎屏幕泄露了用户名、密码提示



设置了密码提示



关闭“欢迎屏幕”，使用传统方式登陆（1）

- 控制面板-用户账户

关闭“欢迎屏幕”，使用传统方式登陆（2）

- 清除“使用欢迎屏幕”复选框。

提高传统登陆方式的安全性（1）

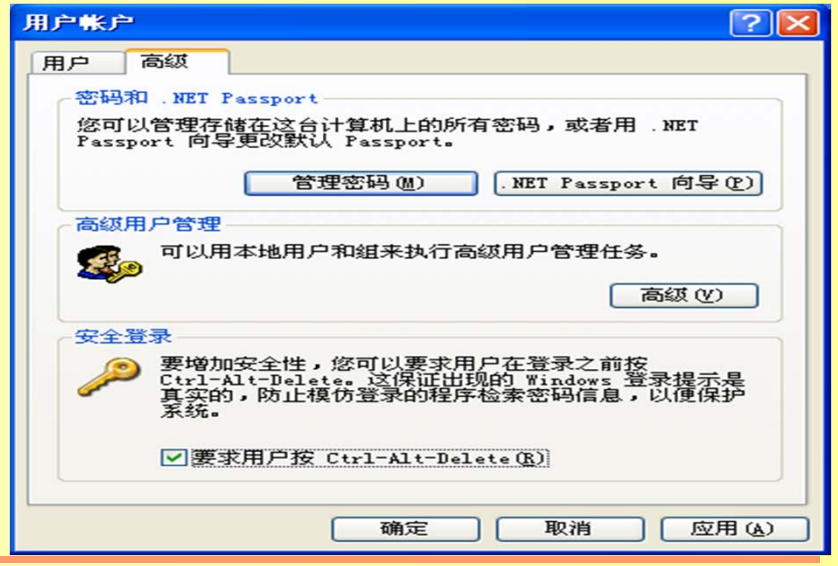
- 自动登陆
  - control userpasswords2
  - 清除“要使用本机,用户必须输入用户名和密码”。
- 关闭自动登陆
  - 勾选“要使用本机,用户必须输入用户名和密码”。

提高传统登陆方式的安全性（2）

- 控制自动登陆
  - 在产品宣传或广告场合：需要自动登陆，不需要密码，但是是指定的受限用户，即使重启也只能以这个受限的用户登录：取消选择用户的机会。
  1. 在HKLM\SoftWare\Microsoft\Windows NT\CurrentVersion\WinLogon添加ForceAutoLogon字符串值，设置为1。
     - 用户登陆后，注销后会自动返回到登陆窗口。
  2. 添加IgnoreShiftOverride字符串值，设置为1。
     - 可以在启动或注销时按下“Shift”键来阻止自动登陆，弹出选择用户的窗口。
  - 注意要给自己留下以高权限用户登陆的机会。

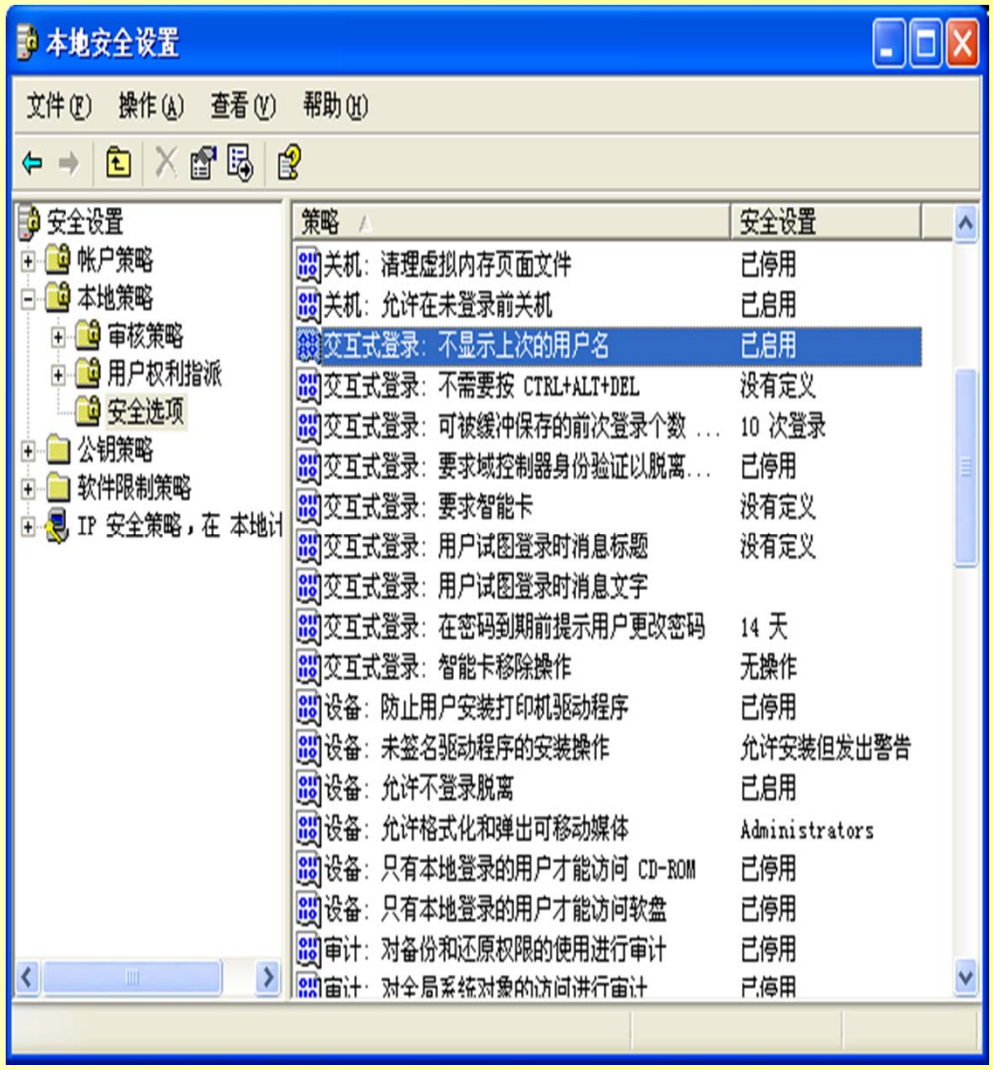
提高传统登陆方式的安全性

- 需要提高传统登陆方式的安全性的理由
  - 其他用户可以替换欢迎登陆界面，窃取您的用户名和密码。
  - Windows保证“Ctrl+Alt+Del”只能被Winlogon截取。
- 设置方式
  - Control userpasswords2
  - 选择“高级”选项卡
  - 勾选“要求用户按Ctrl+Alt+Delete(R)”



提高传统登陆方式的安全性

- 登陆界面不显示上次的用户名
  - 运行secpol.msc
  - 本地策略-安全选项-“交互式登录:不显示上次的用户名”



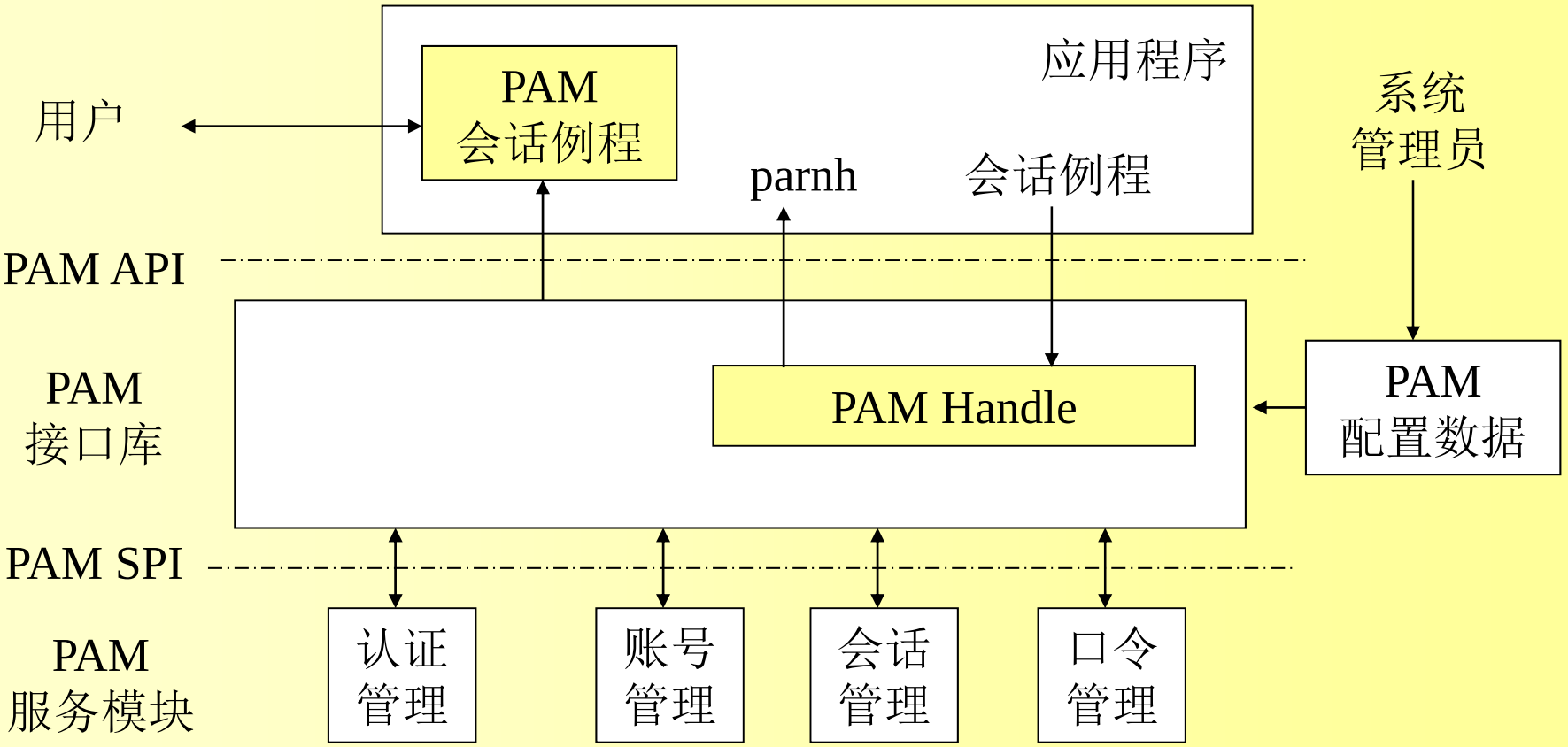
（1）标识与鉴别

- 本地用户认证
  - Unix 用DES 加密用户的口令，放在/etc/passwd或shadow文件中。

Pluggable Authentication Modules（PAM）

- PAM是一套共享库，它包括PAM接口和PAM服务模块。
- 本地的系统管理员通过PAM能够选择应用程序认证用户的方法。
- 一个应用程序使用PAM模块后，应用程序的用户认证机制交给PAM 机制完成，应用程序本身不涉及用户认证的细节。这样，认证工作从程序员交给管理员。
- PAM允许管理员在多种认证方法之间作出选择，它能够改变本地认证方法而不需要重新编译与认证相关的应用程序。

PAM的框架结构


应用程序
PAM
会话例程
parnh
会话例程
用户
系统
管理员
PAM API
PAM
接口库
PAM Handle
PAM
配置数据
PAM SPI
PAM
服务模块
认证
管理
账号
管理
会话
管理
口令
管理

PAM的框架结构

- **应用程序层**：具体的使用PAM机制的应用程序，如login。它调用PAM接口库的上层接口PAMAPI，实现认证功能。
- **PAM API 接口层**：连接应用程序和服务模块的中间层，它根据配置文件中的设置，加载相应的服务模块，将请求传递到具体的服务模块。
- **服务模块**：动态链接库，它给应用程序提供具体的认证用户服务。不同的服务模块完成不同的功能。应用程序可以使用多个服务模块。

PAM的框架结构

- 认证用户的方法交给PAM完成，系统管理员可以任意配置认证具体方法。
- 例如：Login应用程序使用PAM 机制后，可以选择通过本机的etc/passwd内容信息认证用户或者通过服务器认证用户等。尽管认证方法可以任意配置，但应用程序不需要重新编译。
- 系统管理员通过设置PAM配置文件制定认证策略。认证策 指使用的服务和服务采用的认证方法。
- 应用程序使用PAM API的接口， PAM接口层通过PAM SPI 接口找到PAM务模块，服务模块实现用户的认证工作。

PAM的参考文献

- 倪继利，Linux安全体系分析与编程，电子工业出版社，2007年11月。
- 曹江华，Linux服务器安全策略详解，电子工业出版社，2009年6月。

(2) 访问控制

- 自主访问控制
- 强制访问控制
- Flask 体系结构

(2) 访问控制 — 强制访问控制

- Mandatory Access Control (MAC)
- 访问控制策略的判决不受一个对象的单个拥有者的控制。
- 中央授权系统决定哪些信息可被哪些用户访问，而用户自己不能够改变访问权限。
- 强制访问控制出现在军事安全中。

(2) 访问控制 — 自主访问控制

- Discretionary Access Control (DAC)
- 拥有者能够决定谁应该拥有对其对象的访问权及其内容。
- 在商业环境中，常用DAC来允许指定群体中的所有人(有时是其他的命名个体)改变访问权。
  - 例如：一个公司需要建立访问控制以便会计组能够访问个人文件。但是，公司也可以允许Ana和Jose以巡警总办事处领导的身份来访问这些文件。典型地，DAC访问权能够被动态改变。如根据商业的需要，财务文件的拥有者可能向许可访问者的名单里添加Renee，并将Walter从名单中删除。
- MAC和DAC可同时应用于同一个对象。一个行为必须同时满足MAC和DAC控制策略才能实施。

(2) 访问控制 — GFAC

- Generalised Framework for Access Control (GFAC)
- An improved framework for expressing and integrating multiple policy components.
- The main objectives are
  - Make it easy to state, formalise, and analyse diverse access control policies
  - Make it feasible to configure a system with security policies chosen from a vendor provided set of options with confidence that the system's security policy makes sense and will be properly enforced.
  - Construct the model in a manner that allows one to show that it satisfies an accepted definition of each security policy it represents.

(2) 访问控制 — GFAC

GFAC's premise

- **GFAC** is based on the premise
  - all access control policies can be viewed as rules expressed in terms of attributes by authorities;
- **Authorities:** An authorised agent that
  - defines security policies;
  - identifies relevant security information;
  - assigns values to attributes.
- **Attributes**
  - Characteristics or properties of subjects and objects defined within the computer system for access control decision making;
- **Rule:** A set of formalized expressions
  - define the relationships among attributes and other security information for access control decisions in the computer system;
  - reflecting the security policies defined by authority.

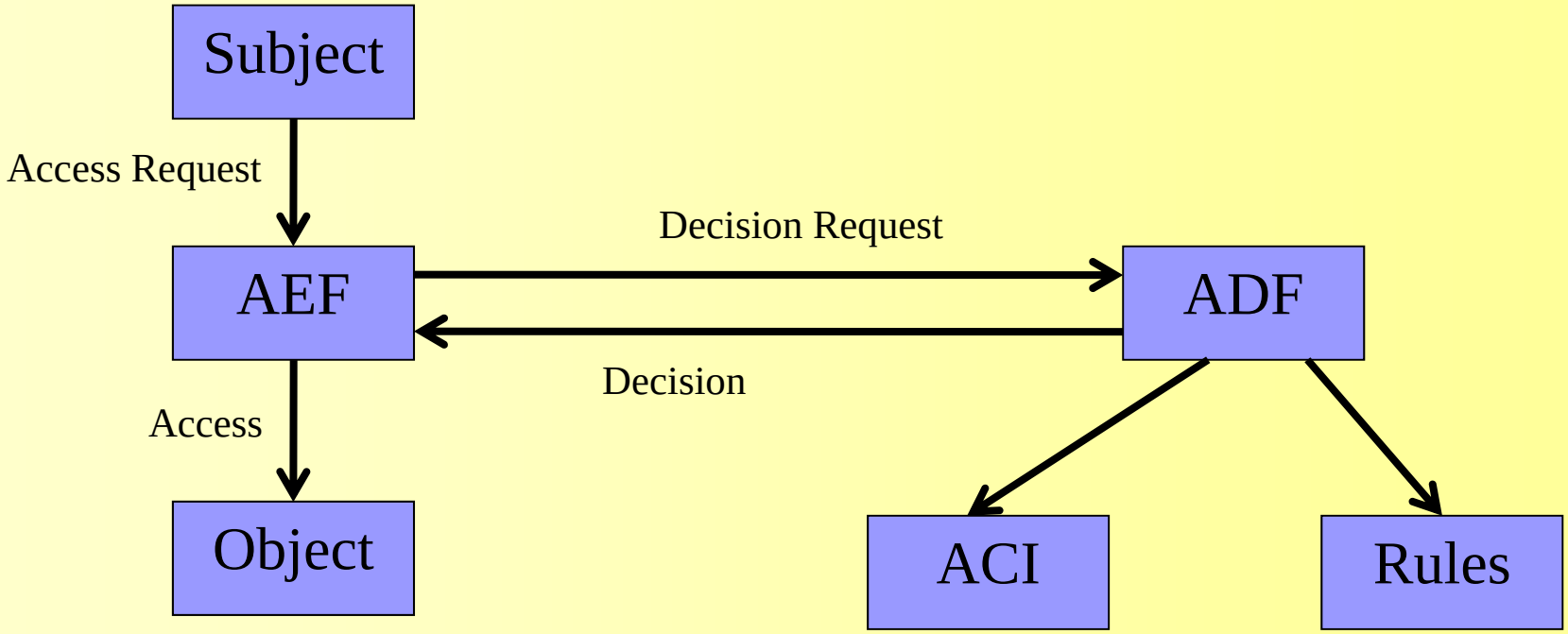
ACI & ACR

- Access control information (ACI)
  - security attributes and other access control data;
- Access control rules (ACR).
  - the rules that implement the trust policies of a system

Adjudication and Enforcement

- The agent that adjudicates access control requests is called access control decision facility (ADF);
  - ADF corresponds to the access control rules within the TCB that embody the system´s security policy.
- The agent that enforces the ADF´s decision is called access control enforcement facility (AEF);
  - AEF corresponds to the system functions of the trusted computing base (TCB)

Overview of GFAC



Rule-Set Modelling approach

- In traditional security modeling approaches, the security model rules describe both access policy and system behavior;
- The rule set modelling approach separates the decision criteria (access rules) from the state transition (system operations)
- AEF corresponds to a model of the system operations, called state-machine model
  - abstractly defines the interface of processes to the TCB;
- ADF corresponds to a policy model, called the rule-set model
  - defines the security policies of the trusted computer system.

(2) 访问控制 — Flask 体系结构

策略的灵活性

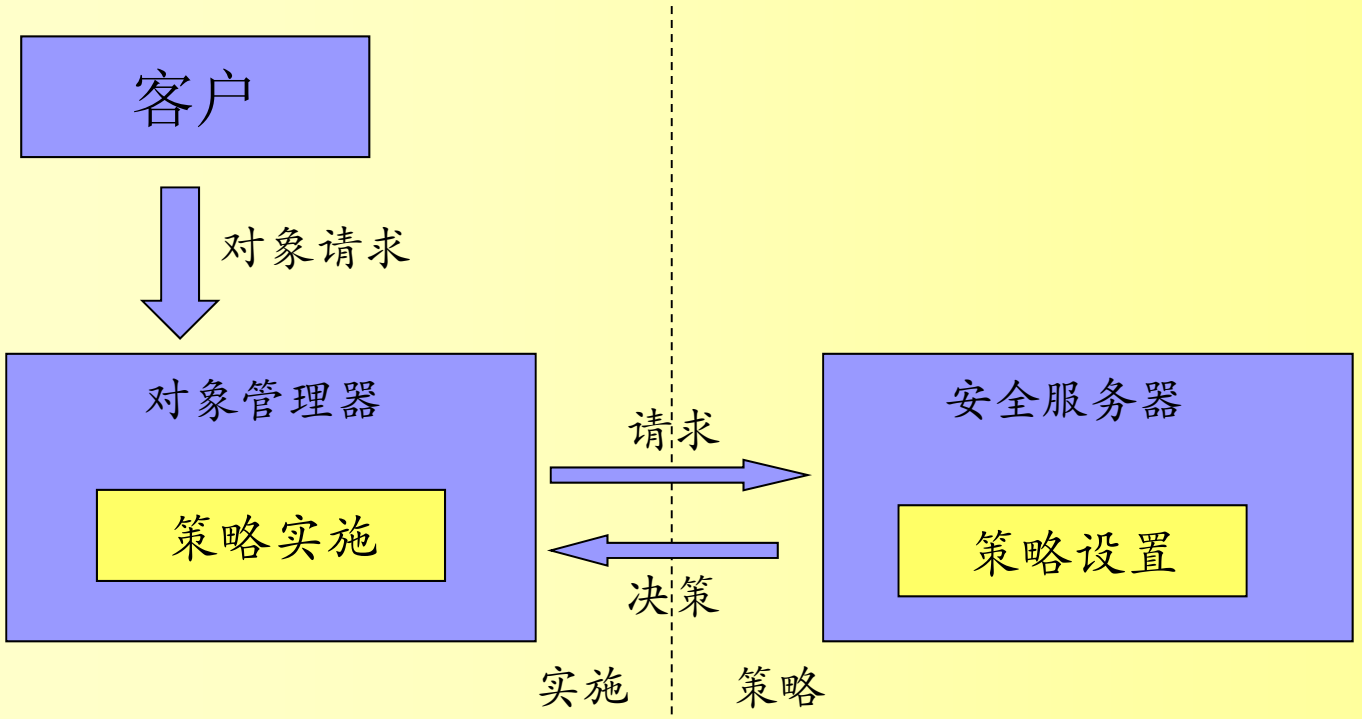
- 不同的安全需求
  - 不同的计算环境，以及运行在上面的应用往往有着不同的安全需求。
- 不同的安全策略和不同类型的策略
  - 因为任何安全概念都是被一个安全策略限制着，而且有许多不同的安全策略甚至许多不同类型的策略。
- 策略的灵活性
  - 为了获得大范围的使用，安全方案必须具有灵活性，足以支持大范围的安全策略。甚至在今天的分布式环境中，这种安全策略的灵活性也必须由操作系统的安全机制来支持。

(2) 访问控制 — Flask体系结构　策略的灵活性

- 系统必须支持对底层客体的细粒度访问控制，使得根据安全策略执行高层功能。
- 系统必须确保访问权限的传播和安全策略一致。
- 策略在通常情况下不是固定不变的。为了解决策的变化和动态的策略，系统必须有一种机制来撤消以前授予的访问权限。

(2) 访问控制 — Flask结构

- 结构的基本目标是提供安全策略的灵活性，确保不管决策怎样产生，怎样随时变化都有一致的策略。
- 其它目标包括应用透明性，深度防御、易于提供保障、最小的性能影响。



- 结构提供一个接口获取
  - 访问决策：一个主体和对象之间许可权是否被允许。
  - 标记：规定分配给对象的安全属性
  - 多实例决策：多实例资源集的哪个成员被特定的请求访问。
- 结构提供一个访问向量缓存(AVC)模块，允许对象管理器缓存访问判定来减小性能损耗。
- 结构提供对象管理器注册，以便接受安全策略的改变通知。

对象标记

- 结构的一个基本要点是怎样维护对象和安全上下文的关联。
- 简单的解决办法是定义一个简单的策略无关的数据类型作为与每个对象相关联的数据的一部分。
- 没有一种数据类型能很好的适应系统中使用的所有不同形式的标记。

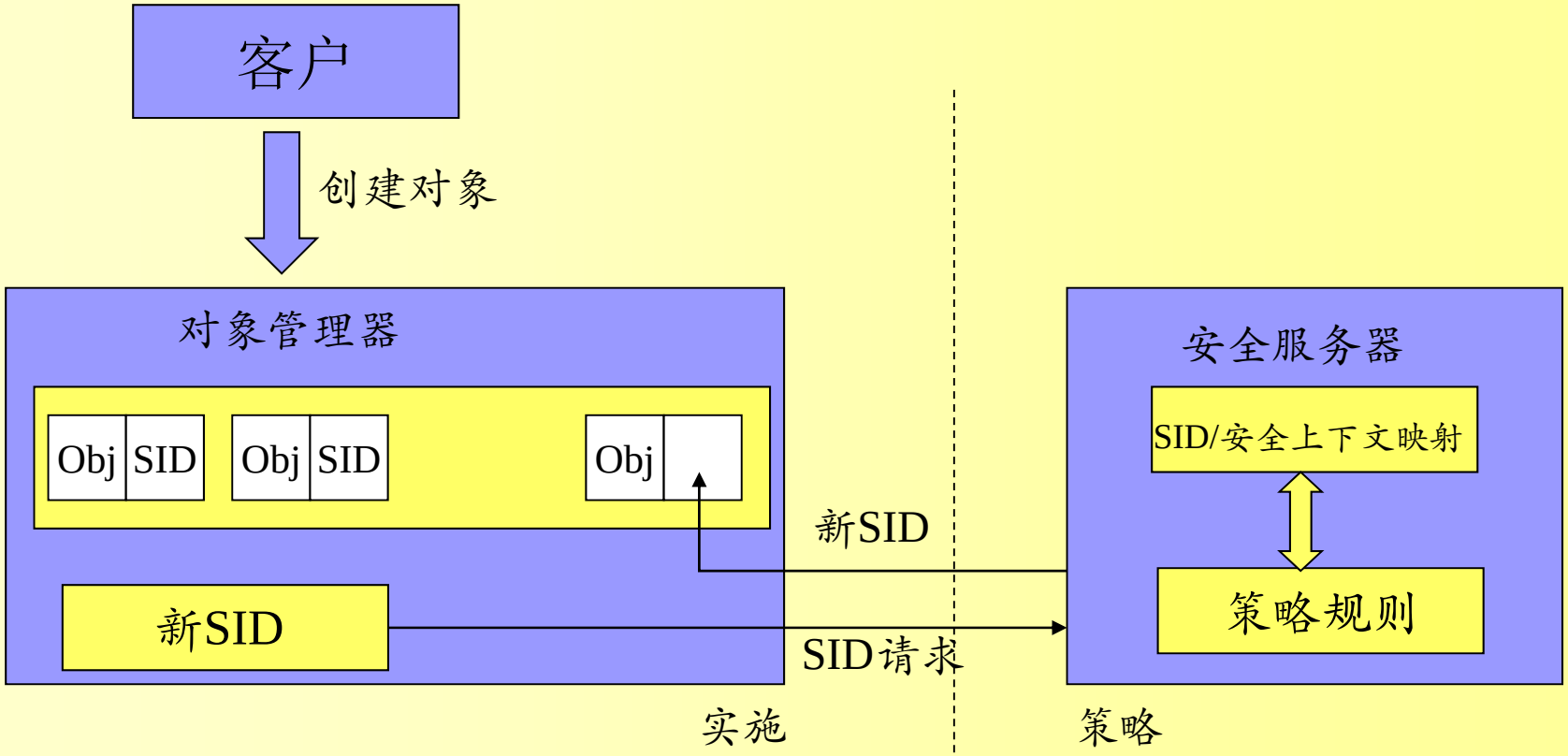
对象标记 —安全上下文

- 一个长度可变的字符串，它能被任何了解安全策略的应用或用户解释。
- 一个安全上下文可以由几个属性组成，如用户标志、安全等级、角色、DTE的域等。但这与特定的安全策略有关。

对象标记 — 安全标志符(SID)

- 它被Flask定义成32位整数，只能被安全服务器解释并由安全服务器映射到一个特定的安全上下文。
- SID没有特定的内部结构；任何内部结构只有安全服务器知道。
- 当一个对象被创建时，它被分配一个SID来表示它创建的安全上下文。
- 上下文依赖于客户对象创建的请求和创建的环境。SID必须由安全上下文和安全服务器选择的唯一标志来计算。

对象标记


客户
创建对象
对象管理器
Obj
SID
Obj
SID
Obj
新SID
新SID
SID请求
安全服务器
SID/安全上下文映射
策略规则
实施
策略

安全上下文的绑定过程

- 用户请求创建一个新的对象;
- 对象管理器得到用户的SID;
- 对象管理器向安全服务器提出一个新对象SID的请求;
- 安全服务器根据策略规则为新对象确定一个安全上下文和相应的安全上下文SID;
- 将新对象的安全上下文SID返回给返回给对象管理器;
- 对象管理器绑定新对象与相应的安全上下文SID。

安全上下文的制定

- 一个新对象的安全上下文与请求的用户和环境有关。
  - 例如：一个用户请求创建一个文件，其安全上下文就与用户、新文件所在的目录有关。

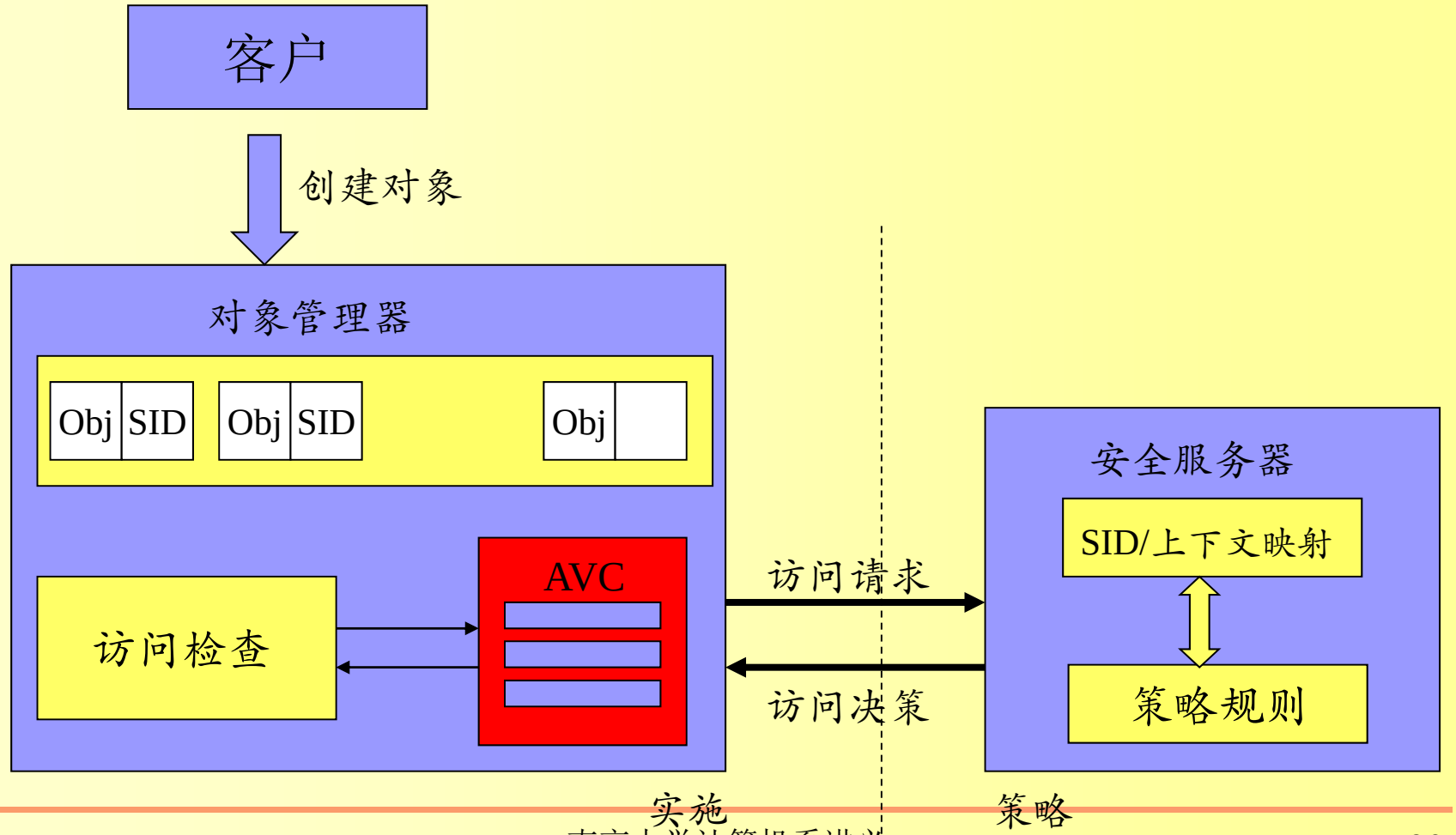
客户和服务器鉴别

- 对象管理器必须能鉴别客户发出的请求SID。
- 必须能使客户能鉴别服务器的SID，以确保服务是从一个适当的服务器发出的。
- Flask微内核作为IPC处理的一部分直接提供这个服务。

缓存安全决策

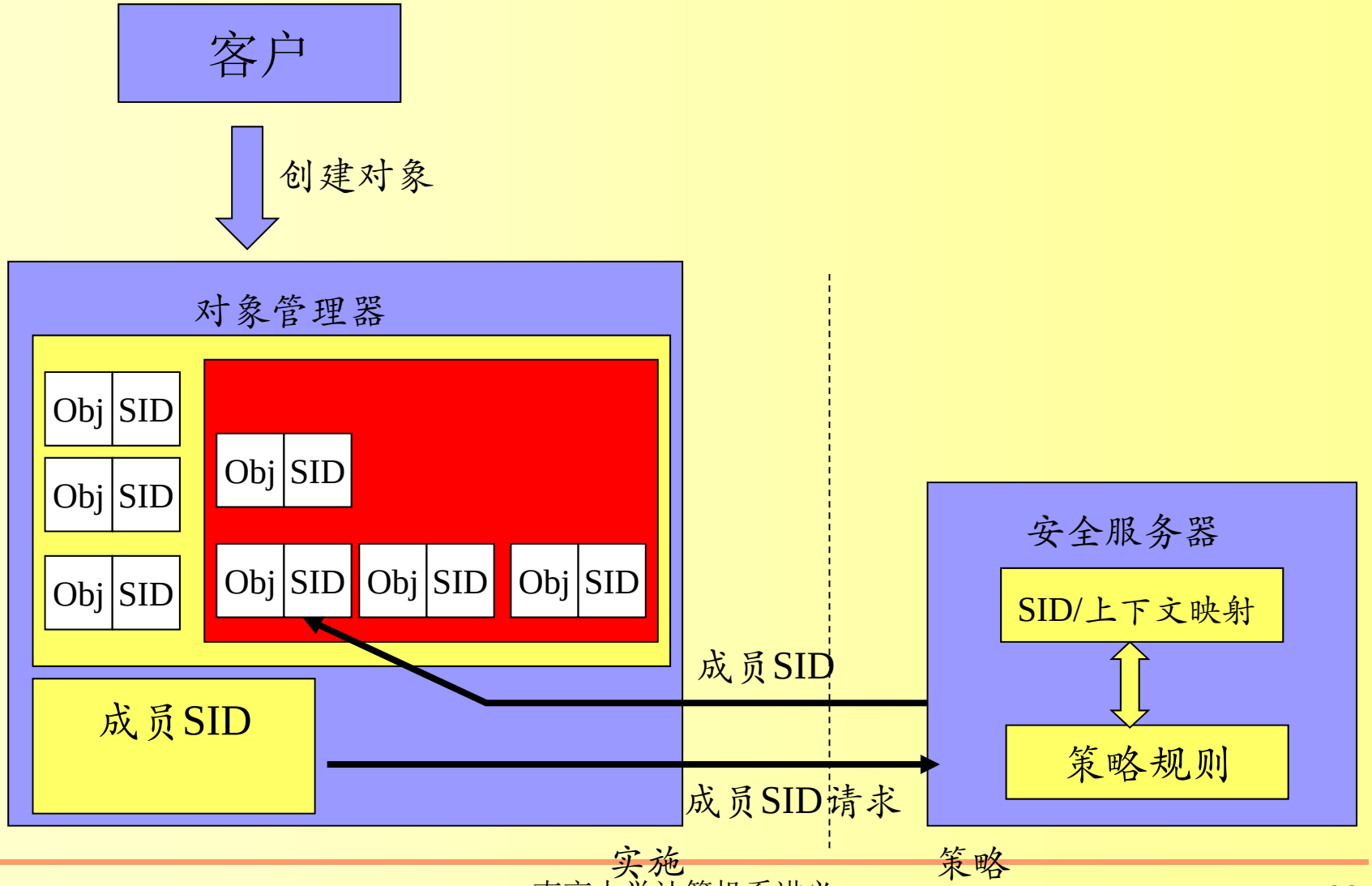
- 缓存访问向量(AVC)模块提供了对象管理器和安全服务器之间的协调策略。
- 这里缓存访问向量是由对象管理器共享的一个公共资源库。
- 这个协调策略既表达了来自对象管理器对策略决策的请求又表达了来自服务器对策略变迁的请求。

缓存安全决策


客户
创建对象
对象管理器
Obj SID
Obj SID
Obj
AVC
访问检查
访问请求
访问决策
安全服务器
SID/上下文映射
策略规则
实施
策略

多实例化

- 通过多实例化某资源及按组划分终端实体，使得每个群中的实体能共享该资源的相同实例，限制固定资源在终端实体间的共享。
- 例如，多级安全的Unix系统经常划分/tmp列表，为每个安全级保留独立的子列表。


客户
创建对象
对象管理器
Obj SID
Obj SID
Obj SID
Obj SID
Obj SID
Obj SID
Obj SID
Obj SID
成员SID
成员SID
安全服务器
SID/上下文映射
策略规则
成员SID请求
实施
策略

支持吊销机制

- 在Flask体系中，最难处理的就是对象管理器要高效保存一些安全决策的局部拷贝。
- 原子性的需求
  - 策略变动完成后，对象管理器的行为必须反映这个变化。
  - 对象管理器必须采用实时的方式完成策略变化。
- 合理定义的协议把对象管理器和安全服务器联系起来
  - 安全服务器让所有对象管理器注意到已改变策略.
  - 每个对象管理器更新内部状态以反映这些变化。
  - 每个对象管理器让安全服务器注意到改变已完成。

（3） 数据安全机制

- Windows的加密文件系统
- LINUX的加密文件系统

EFS为文件加密的方法

① 当第一次一个文件被加密时，EFS为执行此次加密的用户账户分配一对私钥／公钥。

② 当一个文件被加密时，EFS为该文件生成一个随机数，EFS称其为该文件的文件加密密钥(FEK，file encryption key)。

③ EFS使用此FEK来加密该文件的内容。

④ 通过RSA公钥加密算法，利用该用户的EFS公钥对FEK进行加密。EFS将该文件加密后的的FEK与该文件存储在一起。

EFS加密文件的特点

① 文件的属主不登陆，系统无法获得解密密钥。

② 用户的帐户被删除或系统崩溃后，无人可以恢复密文。

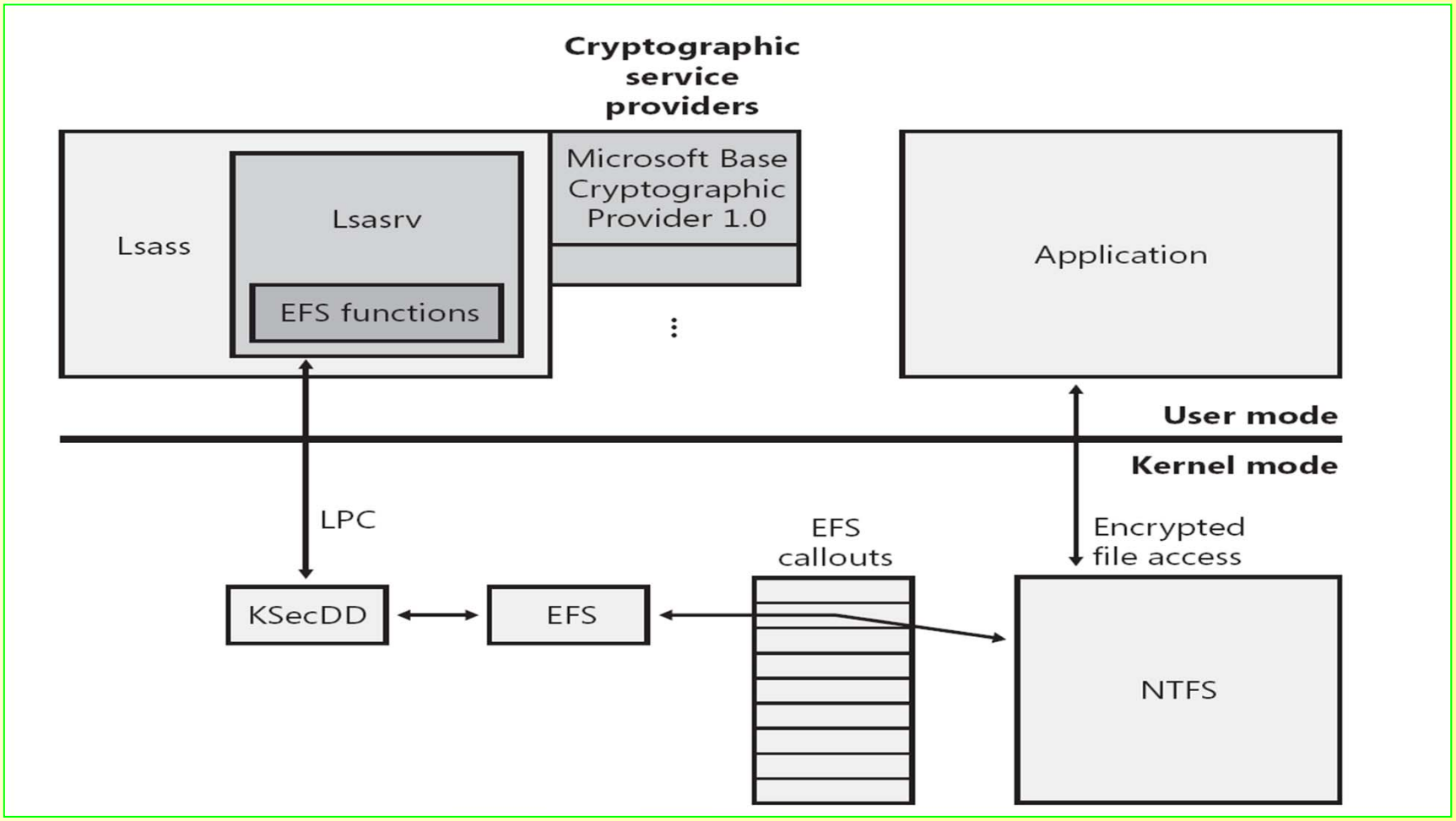
③ 用户的私钥可以导出，备份密钥可以在系统崩溃后恢复密文数据。

④ 系统可以建立恢复代理，在文件的属主用户的密钥无法恢复时，恢复密文数据。

用户的私钥

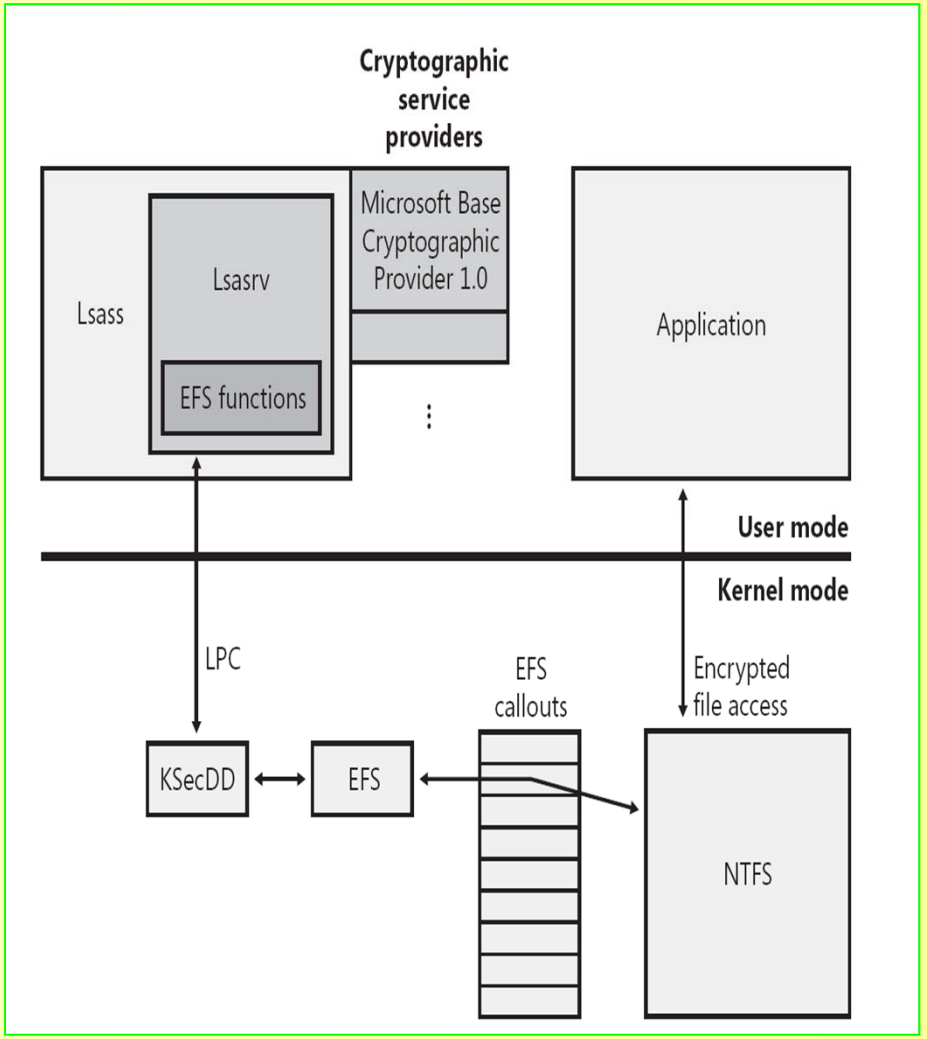
- Windows 把一个用户的私钥保存在它的用户轮廓目录中：ApplicationData\Microsoft\Crypto\RSA
- Windows用一个随机的对称密钥，即该用户的主密钥(masterkey)来加密RSA文件夹中的所有文件。
  - 主密钥有64字节长，是通过一个强随机数生成器得到的。主密钥也存储在Application Data\Microsoft\Protect目录下该用户的轮廓中，是用3DES来加密的，并且加密密钥是部分根据该用户的口令生成的。
- 用户口令改变
  - 当一个用户改变其口令时主密钥也自动被解密，并利用新的口令重新加密。

EFS 体系结构


Cryptographic service providers
Microsoft Base Cryptographic Provider 1.0
Lsass
Lsasrv
EFS functions
Application
User mode
Kernel mode
LPC
EFS callouts
Encrypted file access
KSecDD
EFS
NTFS

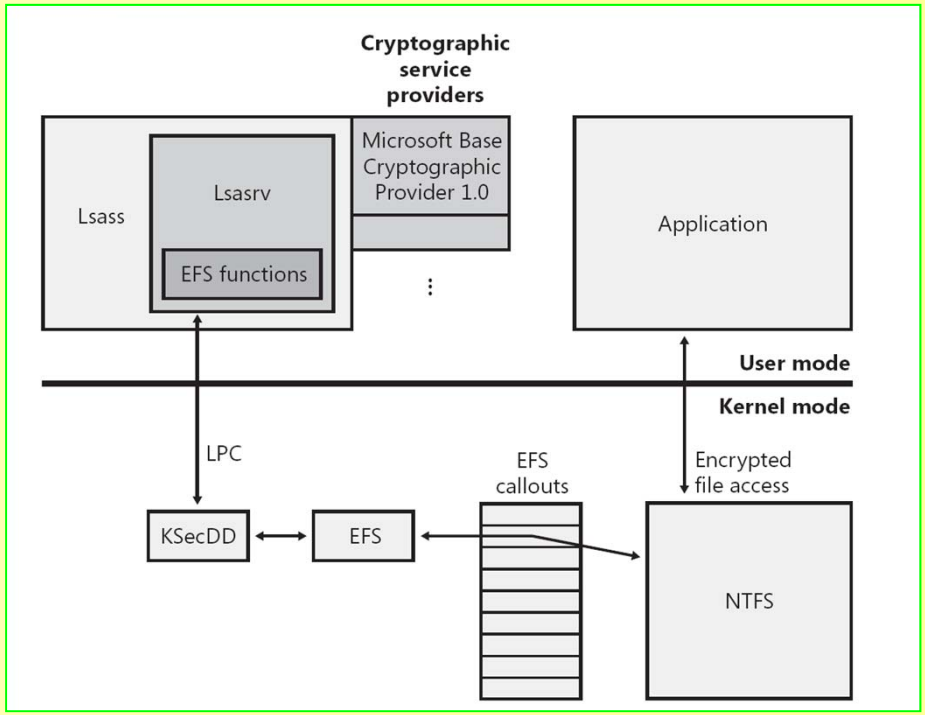
EFS体系结构

- Windows 2000中
  - EFS被实现为一个设备驱动程序，它运行在内核模式下，与NTFS文件系统驱动程序紧密地连接在一起。
- WindowsXP和Windows 2003
  - EFS的功能被合并到NTFS驱动程序中。
- 加密与解密
  - 当NTFS碰到一个加密文件时，NTFS执行EFS的函数；
  - 这些函数是当EFS内核模式代码初始化时向NTFS注册的；
  - 当应用程序访问加密文件时，EFS的这些函数负责加密和解密文件数据。



EFS 体系结构

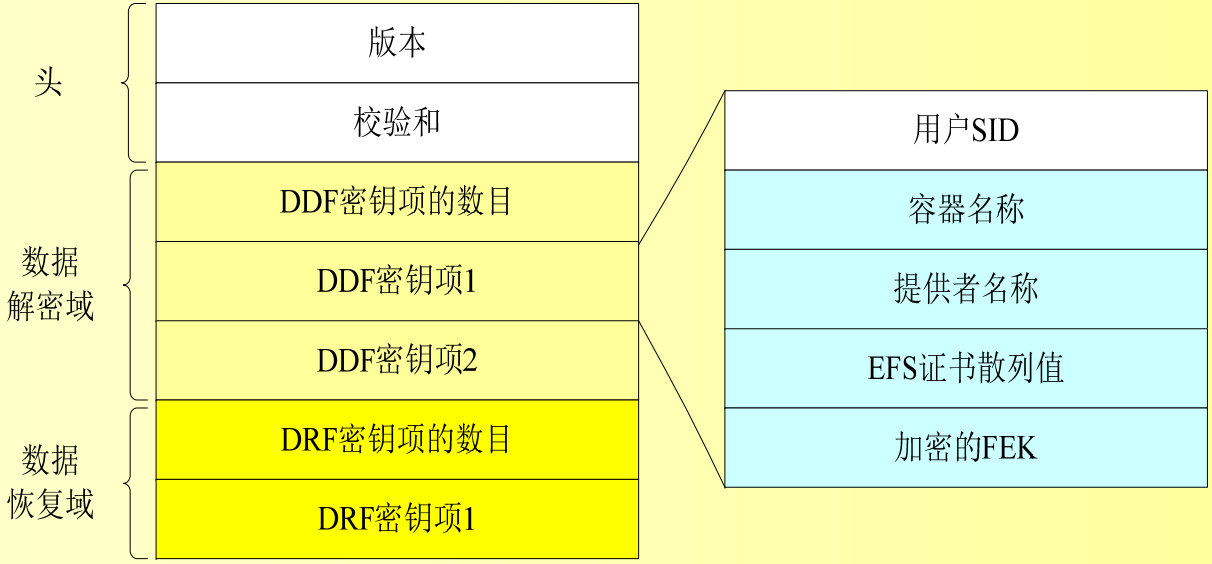
- 本地安全权威子系统(LSASS-\Windows\System32\LSASS.exe)不仅负责管理登录会话，而且也处理与EFS密钥管理相关的事务。
- 当EFS驱动程序需要解密一个FEK以便解密用户想要访问的文件数据时，EFS驱动程序向LSASS发送一个请求。
- EFS通过一个本地过程调用(LPC)发送这一请求。
- KSecDD设备驱动程序导出了一些函数，供其他的需要向LSASS发送LPC消息的驱动程序使用。



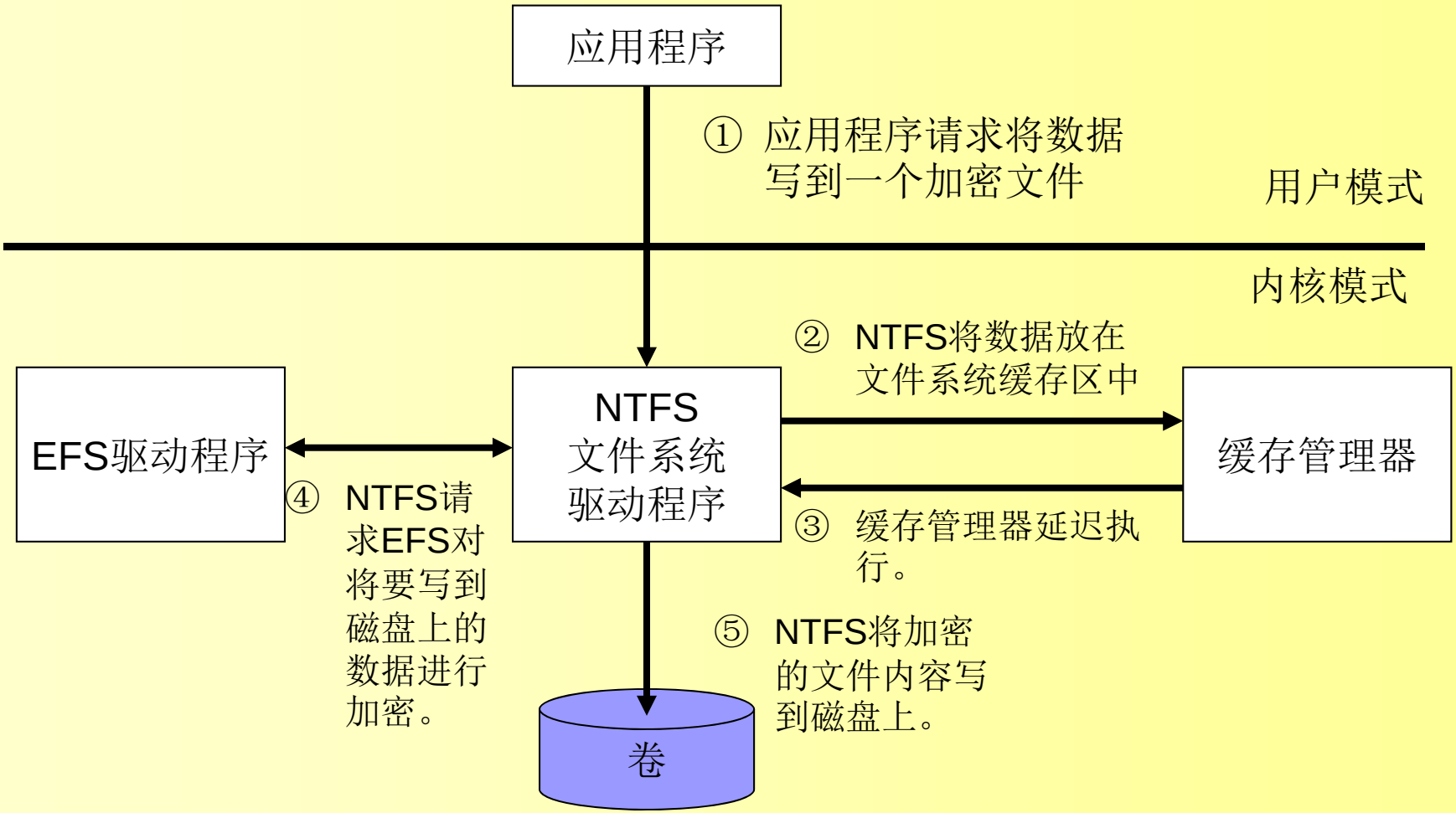
- Lsasrv使用Microsoft CryptoAPI中的函数来解密此FEK（EFS驱动程序刚才将此FEK以加密形式传递给LSASS了）。

EFS信息

- 对于共享此文件的每个用户，该信息块都包含了一个数据项，称为密钥项 。
- EFS将它们存储在该文件的EFS数据的数据解密域(DDF，DataDecryption Field)部分。
- 数据恢复域 DRF (Data Recovery Field)项的格式与DDF项的格式完全相同。



EFS流程


应用程序
① 应用程序请求将数据写到一个加密文件
用户模式
内核模式
② NTFS将数据放在文件系统缓存区中
EFS驱动程序
NTFS 文件系统 驱动程序
缓存管理器
④ NTFS请求EFS对将要写到磁盘上的数据进行加密。
③ 缓存管理器延迟执行。
⑤ NTFS将加密的文件内容写到磁盘上。
卷

加密过程

① 在System Volume lnformation目录下创建一个日志文件，名为Efsx.log。随着后续的步骤逐渐被执行，该日志文件中就会留下纪录，万一在加密过程中系统失败的话，该文件可以被恢复。

② 通过基础密码提供者1.0为该文件生成一个随机的128位FEK。

③ 生成或者获得一个用户的EFS私钥／公钥对。

④ 为该文件创建一个DDF密钥环，其中有一项是为该用户而建立的。此密钥项包含该FEK的一份拷贝，但已经用该用户的EFS公钥进行了加密。

⑤ 为该文件创建一个DRF密钥环。对于系统中的每个恢复代理都有一个对应的密钥项，且每一项所包含的FEK都用相应的恢复代理的EFS公钥进行了加密。

⑥ 在被加密文件所在的目录中，创建一个备份文件，其名称的形式为Efs0.tmp。

⑦ 将DDF和DRF密钥环添加到头部，作为该文件的EFS属性。

⑧ 将原始文件标记为加密文件，并将其内容拷贝到备份文件中。

⑨ 原始文件的内容被销毁，备份文件的内容加密后后被拷回到原始文件中。

⑩ 删除备份文件、删除日志文件。

解密过程

- NTFS在打开该文件时检查其属性，然后执行EFS驱动程序中的一个回调函数。
- 驱动程序调用了NTFS导出给EFS使用的EFS支持函数， 读入与此加密文件相关联的$EFS属性。
- EFS驱动程序确保：当前正在执行打开操作的用户有权访问该文件的加密数据(也就是说，在DDF或DRF密钥环中有一个加密的FEK对应于与该用户相关联的私钥 / 公钥对)。
- 在EFS执行了此项验证以后，EFS获得该文件的解密的FEK，以便用于后续的数据操作。
  - 依赖于Lsasrv来执行FEK解密操作。
  - EFS通过Ksecdd.sys驱动程序给Lsasrv发送一个LPC消息，请Lsasrv获得 “在该文件的$EFS属性数据(EFS数据)中对应于当前正在执行打开操作的用户的FEK的解密形式”。
  - 解密后的FEK的缓存。

文件驱动与缓存管理器

- 一个应用程序打开了一个加密的文件以后，它可以从该文件中读数据，或者向文件写数据。
- NTFS从磁盘上读取数据时，它先调用EFS驱动程序来解密文件数据，再把解密后的文件数据放到文件系统缓存中。
- 当应用程序向一个文件写数据时，文件数据在文件系统缓存中一直是非加密的形式，一直到应用程序或者缓存管理器通过NTFS将数据刷新到磁盘上。
- 当一个加密文件的数据被从缓存中写回到磁盘上的时，NTFS调用EFS驱动程序来加密这些数据。

加密文件的备份

- 除了“通过加密设施来访问加密文件”的应用程序以外，其他应用程序是无法得到非加密形式的文件数据的。
- 专门为备份工具提供一个设施，让备份工具能够在文件处于加密状态下进行备份和恢复操作。
- 备份工具具不必具备解密文件数据的能力，在其备份过程中无需解密文件数据。

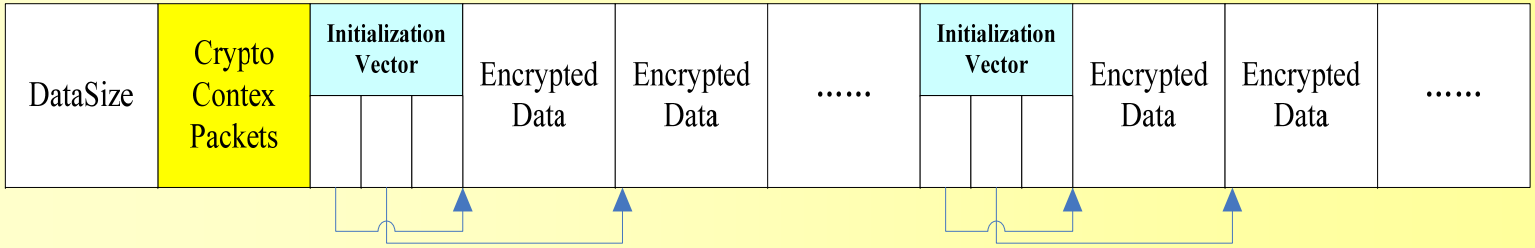
Windows加密文件系统方面的工作

- API层
- 系统服务层
- 过滤驱动层

Linux系统的加密文件系统 eCryptFS

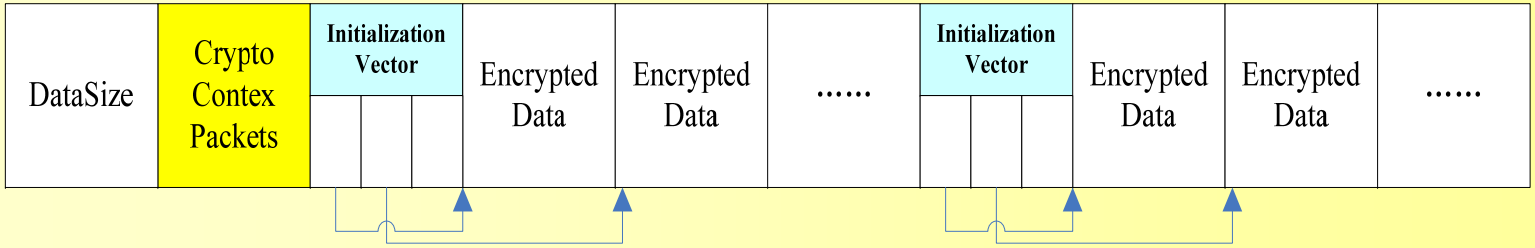
- eCryptFS的加密文件格式
- LINUX文件的HMAC
- 加密文件系统的结构

Linux的加密文件格式



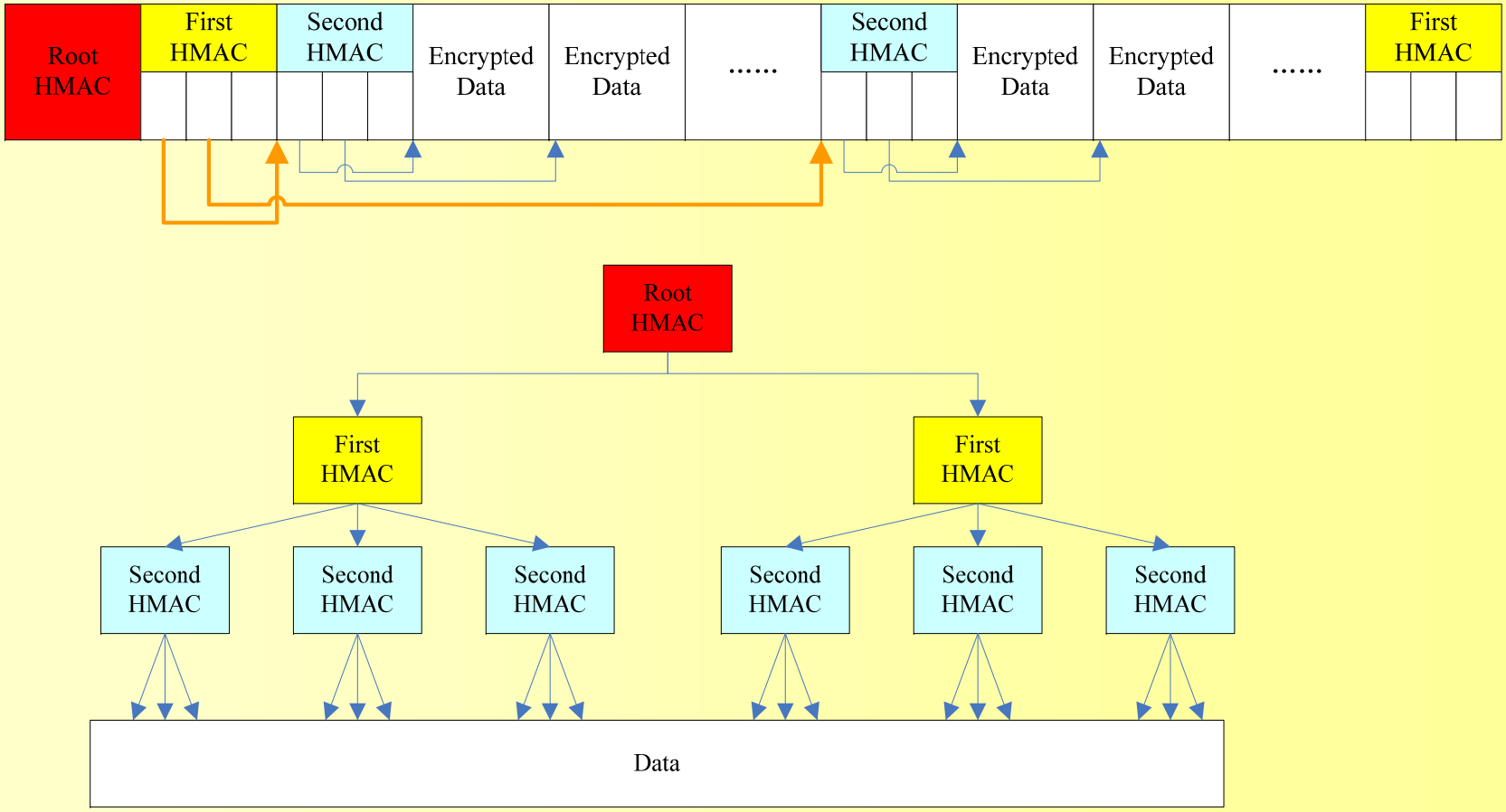
- 加密上下文包(RFC2440)
  - tag1：公钥加密的会话密钥密文
  - tag2：数字签名包
  - tag3：对称算法加密的会话密钥密文
  - tag11：数据包

Linux的加密文件格式

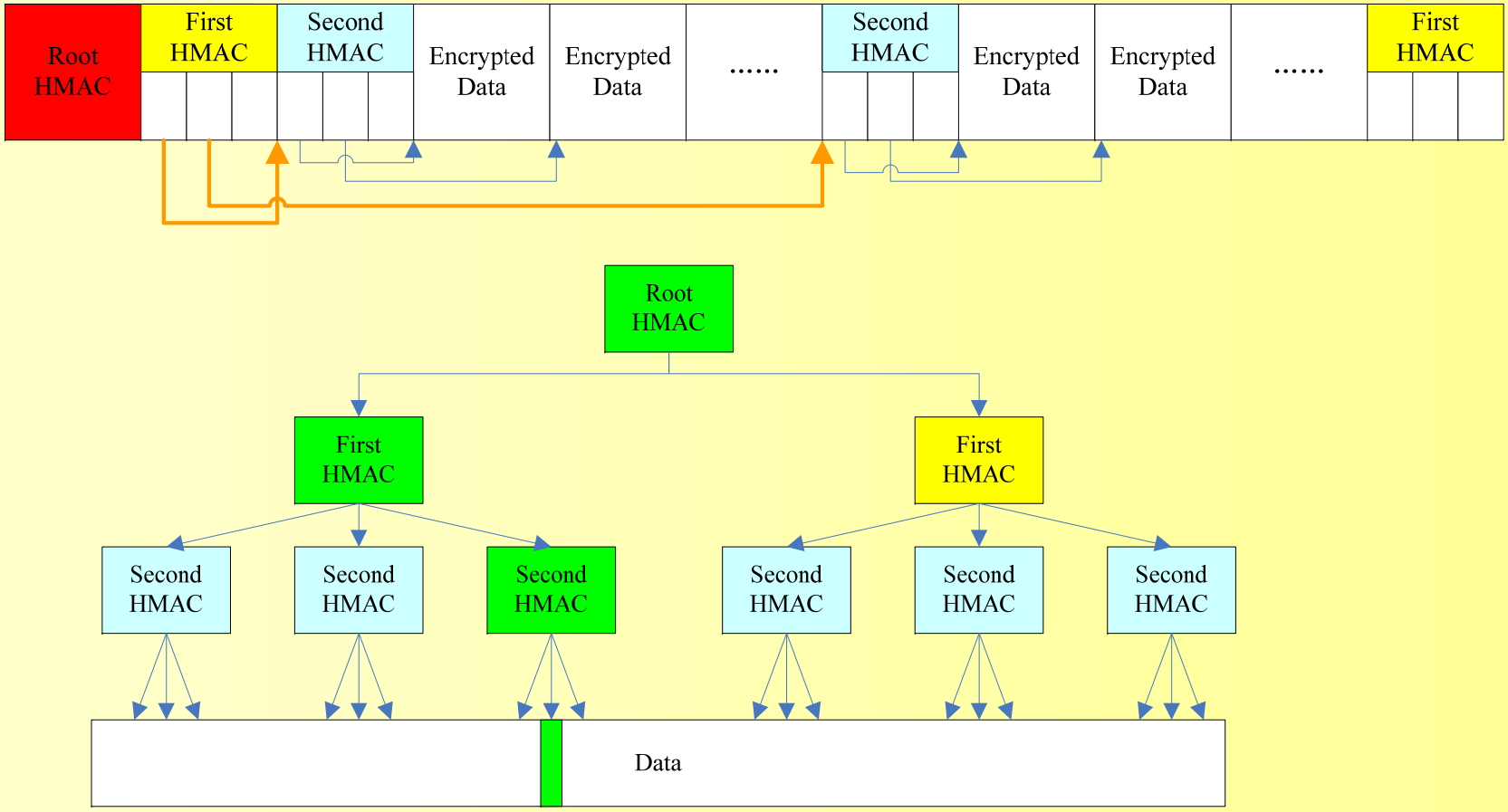


- 区块
  - 每个文件有唯一的一个根初始向量
  - 每个区块包含一个唯一的初始向量
  - 每个区块的初始向量由根初始向量派生
  - 密码操作采用密码分组链接(CBC)
  - 可以加快文件的操作，提高应用程序的处理速度。

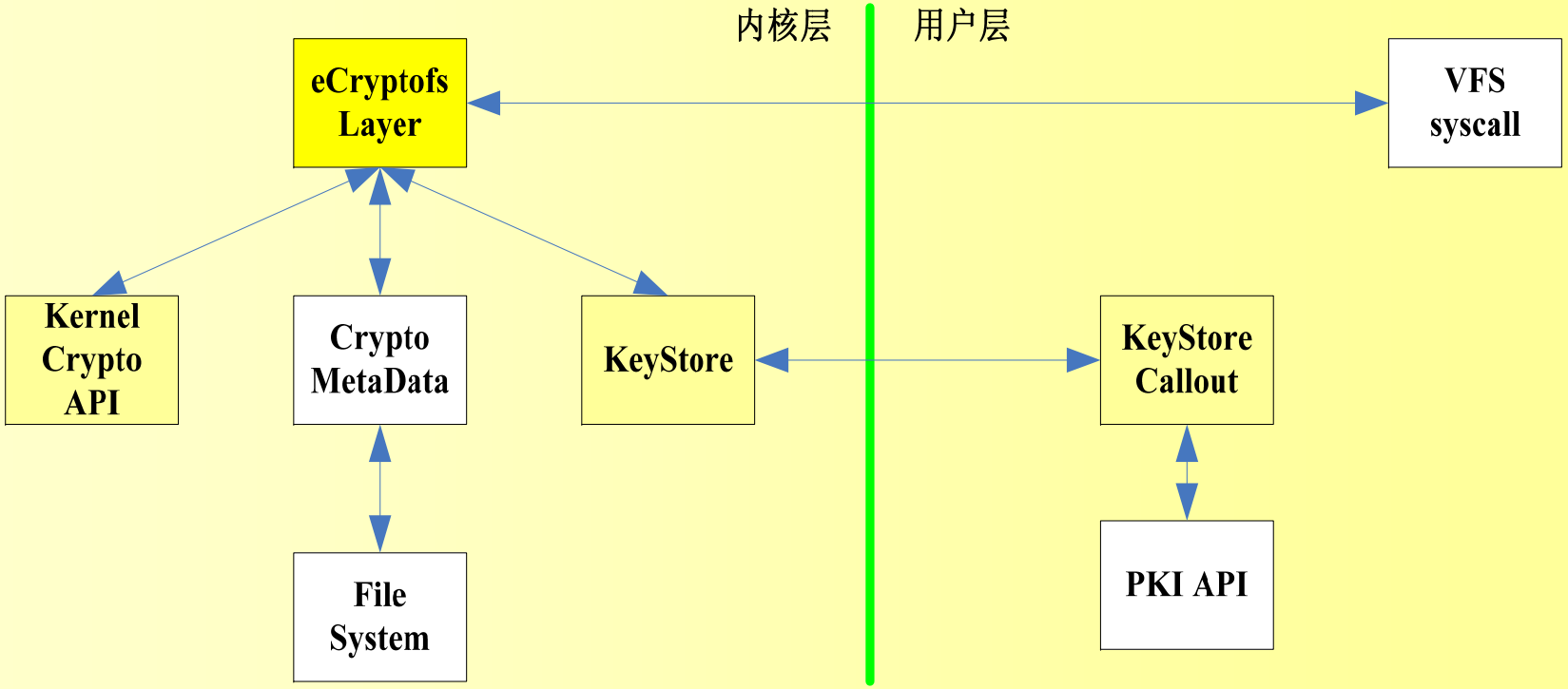
LINUX文件的HMAC



LINUX文件的HMAC



加密文件系统的结构


内核层
用户层
eCryptofs Layer
VFS syscall
Kernel Crypto API
Crypto MetaData
KeyStore
KeyStore Callout
File System
PKI API

BitLocker

- 为什么需要BitLocker
  - 操作系统可以在其工作时保护系统和系统中的数据。
  - 如果计算机的硬盘被卸下来放在其它的系统中，硬盘中的数据就无法受到原操作系统的保护了。
  - 即使原来的文件夹用EFS加密过，也可能无法保护了。
    - 攻击者可能获取口令hash
    - 进行口令破解
    - 利用口令访问EFS加密的数据文件

为什么需要BitLocker

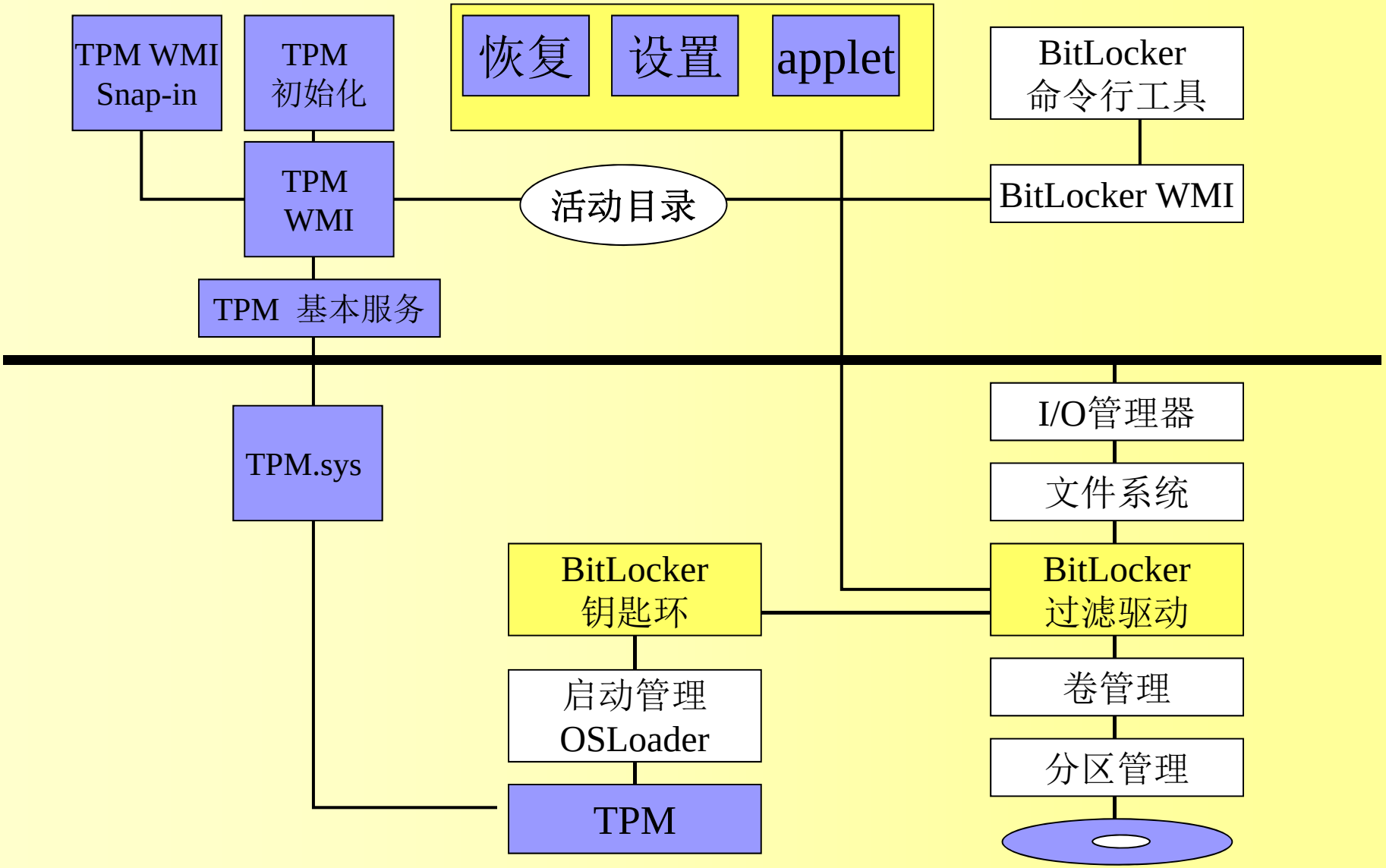
- BitLocker 可以加密整个系统分区。
- 验证启动分区的操作系统的完整性。

- 使得移动电脑在丢失的情况下其中的数据不会泄露。

BitLocker 结构

- 内核模式驱动：%System%\System32\Drivers\tpm.sys
- 用户模式的服务： %System%\System32\tbssvc.dll
  %System%\System32\tpm.msc
- BitLocker相关的启动管理
- BitLocker的过滤驱动：
  %System%\System32\Drivers\Fvevol.sys 实时加解密硬盘分区的数据。
- BitLocker的Window管理接口

BitLocker的结构



- BitLocker 利用“全盘加密密钥FVEK”加密整个磁盘(AES128-CBC 或AES256-CBC ）。
- FVEK被“卷主密钥VMK”加密存储。
- VMK的利用使得在VMK失效时只要将用新的VMK加密FVEK，销毁原来的被加密的FVEK。而不需要重新加密磁盘。

VMK的加密

- 用保存在TPM中的密钥加密VMK。
  - 能抵抗软件攻击，不能抵抗硬件攻击。
- 用保存在TPM中的密钥和一个U盘中密钥加密VMK。
  - 完全能抵御硬件攻击，但是U丢失就有问题。
- 用保存在TPM中的密钥和PIN码加密VMK。
  - 能抵御大多数的硬件攻击。
- 用保存在TPM中的密钥、PIN码和一个U盘中密钥加密VMK。
  - 最安全的一种方式。
- 用U盘中密钥和PIN码加密VMK。
  - 最低程度的保护。

BitLocker的启动过程

- TPM自检
- CRTM度量自身和PCR装载代码，将hash写入第1个PCR。
- 度量BIOS，也将hash写入第1个PCR。
- BIOS 度量系统盘的MBR……

每一个步骤度量下一个组件，将其hash写入指定的PCR，装载这个组件，转移控制权。

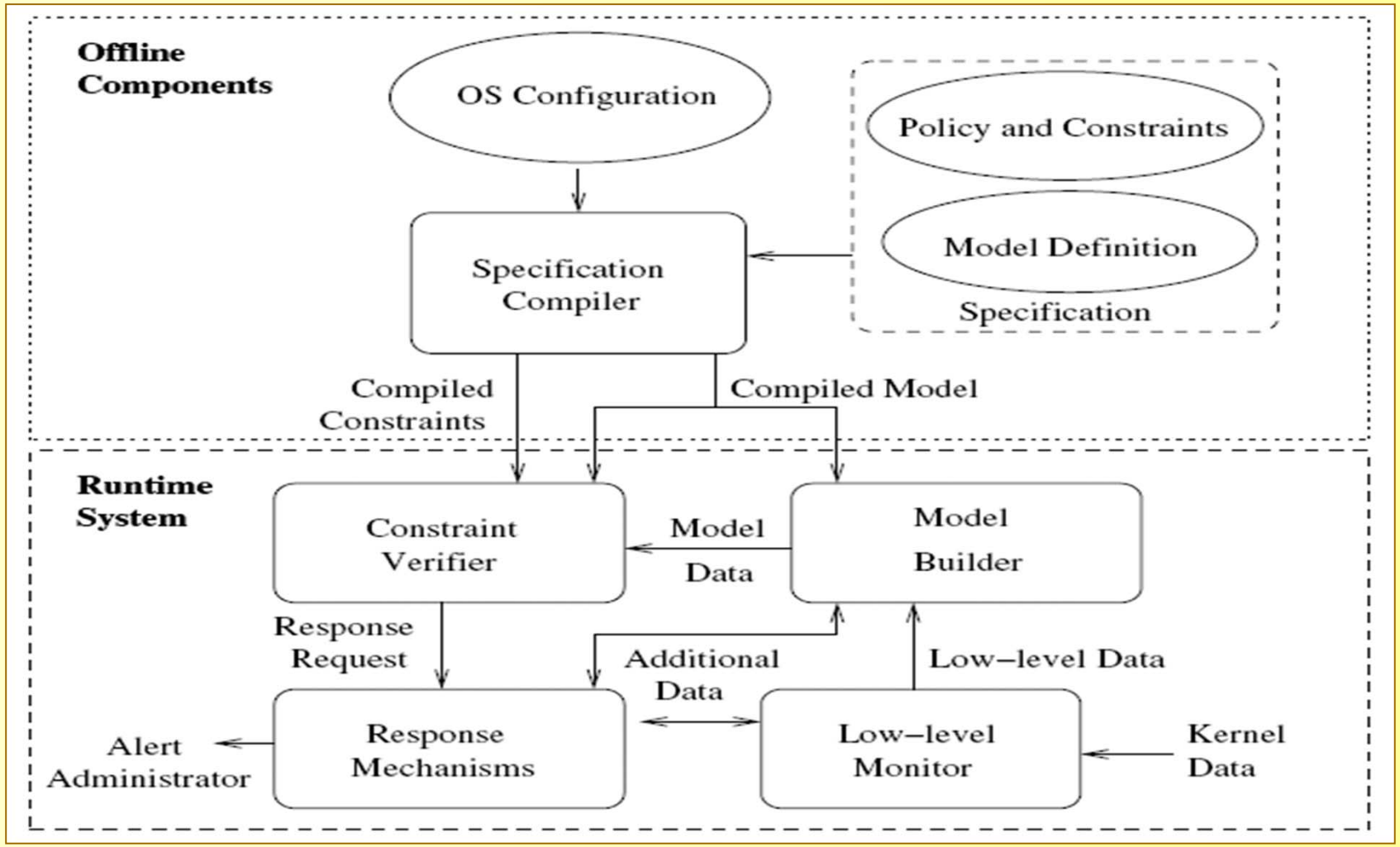
(4) 入侵检测

- 与审计精简紧密联系的是检测安全漏洞的能力，理想情况下是在它们发生的时候就被检测出来。
- 在审计日志中有太多的信息需要人们去分析。但是计算机有助于将独立数据联系起来。
- 入侵检测(intrusion detection)软件构造了正常系统使用的模式，一旦使用出现异常就发出警告。

基于语义的内核完整性监控

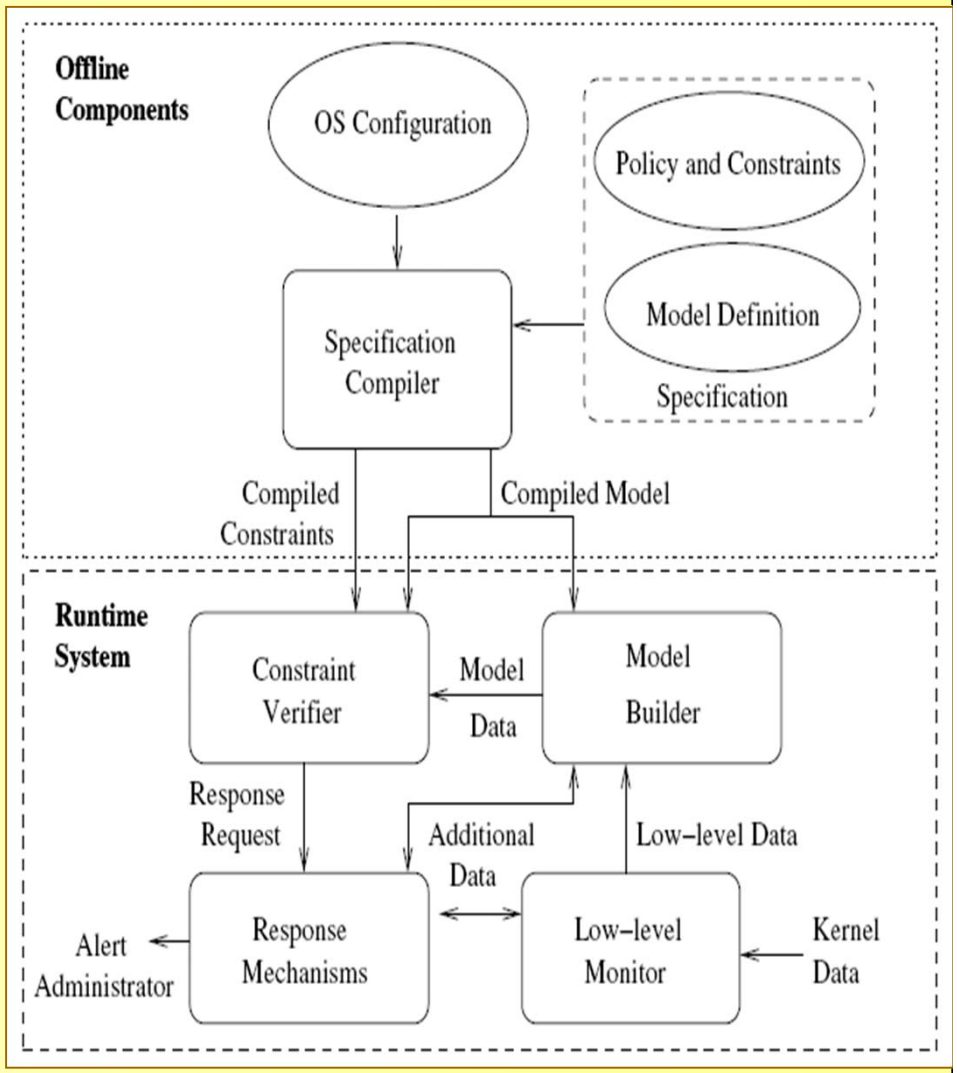
- **Nick L. Petroni, Jr., Timothy Fraser, AAron Walters, William A. Arbaugh, An Architecture for Specification-Based Detection of Semantic Integrity Violations in Kernel Dynamic Data, Security ’ 06: 15th USENIX Security Symposium.**

基于语义的 操作系统内核完整性监控



基于语义的操作系统内核完整性监控

- 底层监视器
  - 能够读到内核所有虚拟存储空间，而不必依赖被监控的内核的完整性。
  - 虚拟机监控器，协处理器等。
- 模型生成器
  - 将底层监视器获取的内存数据转化成高层内核数据对象。
- 一致性验证器
  - 验证生成的高层内核数据对象是否满足完整性条件。
- 响应机制
  - 当内核数据对象不满足完整性条件时给出响应。
- 规约生成器
  - 定义高层数据对象、完整性条件、响应机制



持久性内核控制流攻击的自动化检测

- **S Bhatkar, A Chaturvedi, R Sekar, Dataflow anomaly detection, Proc. IEEE Symposium on Security and Privacy, 2006.**

  Department of Computer Science University of Maryland.
- 文章提出了基于状态的控制流完整性（state-based control-flow integrity）的动态操作系统内核完整性监控。
- 在虚拟机监控器Xen和VMware中实现。
- 实验中能够检测出25个linux rootkit中的24个。

持久性内核控制流攻击的自动化检测

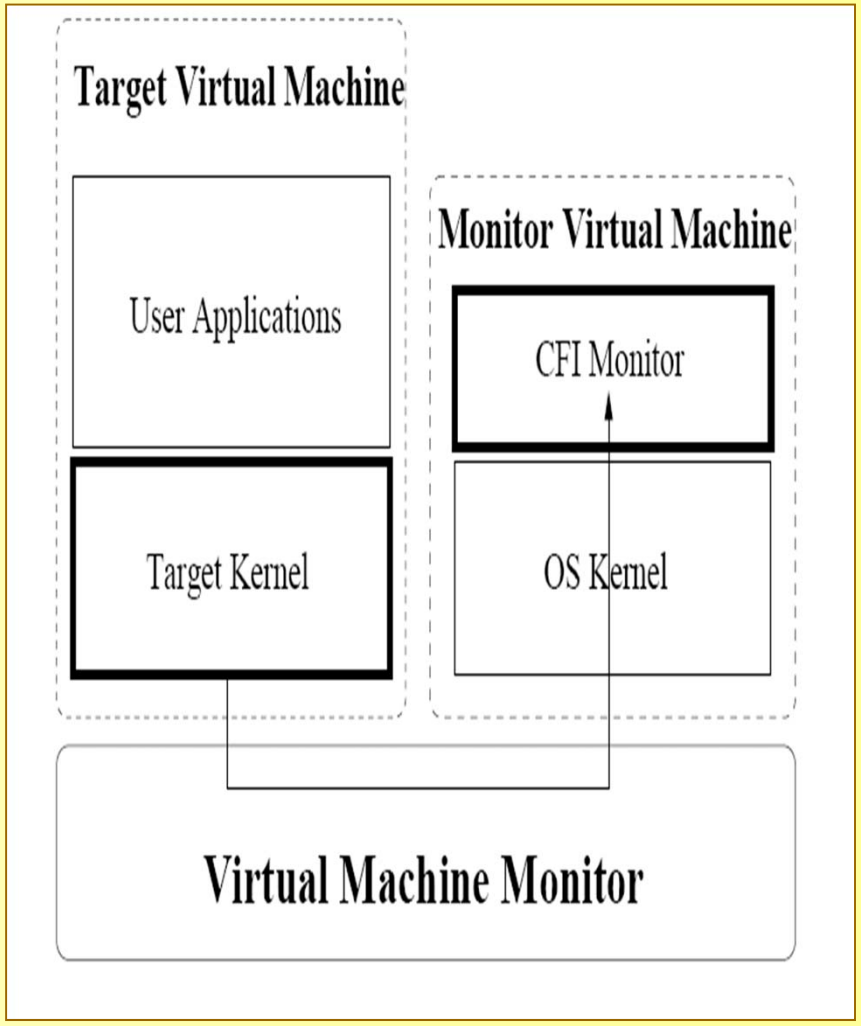
- 控制流完整性
  - 如果一个程序P到检测时刻为止都是沿着设计的控制流图（control-flow graph (CFG)）执行，则称程序P满足控制流完整性（control-flow integrity (CFI)）。

持久性内核控制流攻击的自动化检测

- **基于状态的控制流监控**
  - 每隔一段时间检查系统状态而不是随程序执行每步检查。
  - 通过监控器保存内核代码的备份确保内核文本，包括静态控制流传递的正确性；
  - 通过类似垃圾收集机制的扫描栈、寄存器、堆、全局变量，根据指针定位函数验证正确性确保动态控制流传递的正确性。

持久性内核控制流攻击的自动化检测

- 实现
  - 在Xen的实现中，目标内核和监控器分属不同的虚拟机。在Vmware的实现中，目标内核在一个虚拟机中，监控器在宿主机中作为一个程序运行
- M(n, p)
  - 根据性能与安全的平衡调整检测间隔的步数n。
  - 根据研究rootkit得到的特性p确定是否满足CFI。



基于数据流的完整性保护

- **Miguel Castro, Manuel Costa, Tim Harris. Securing software by enforcing data-ow integrity. Proceedings of the 7th Symposium on Operating Systems Design and Implementation, 2006, 146-160.**
- 静态分析:
  - 分析被监控程序的关键数据的写于读。
  - 分析每一个“读”的可达集。
- 动态监控:
  - 在每一个“读”与“写”插入监控代码。
  - 在“写”记录指令的地址。
  - 在“读”处检查，记录的“写”地址是否在“读”的可达集中。

基于数据流的完整性保护

```
1: int authenticated = 0;
2: char packet[1000];
3:
4: while (!authenticated) {
5: PacketRead(packet);
6:
7: if (Authenticate(packet))
8:         authenticated = 1;
9: }
10:
11: if (authenticated)
12:         ProcessPacket(packet);
```

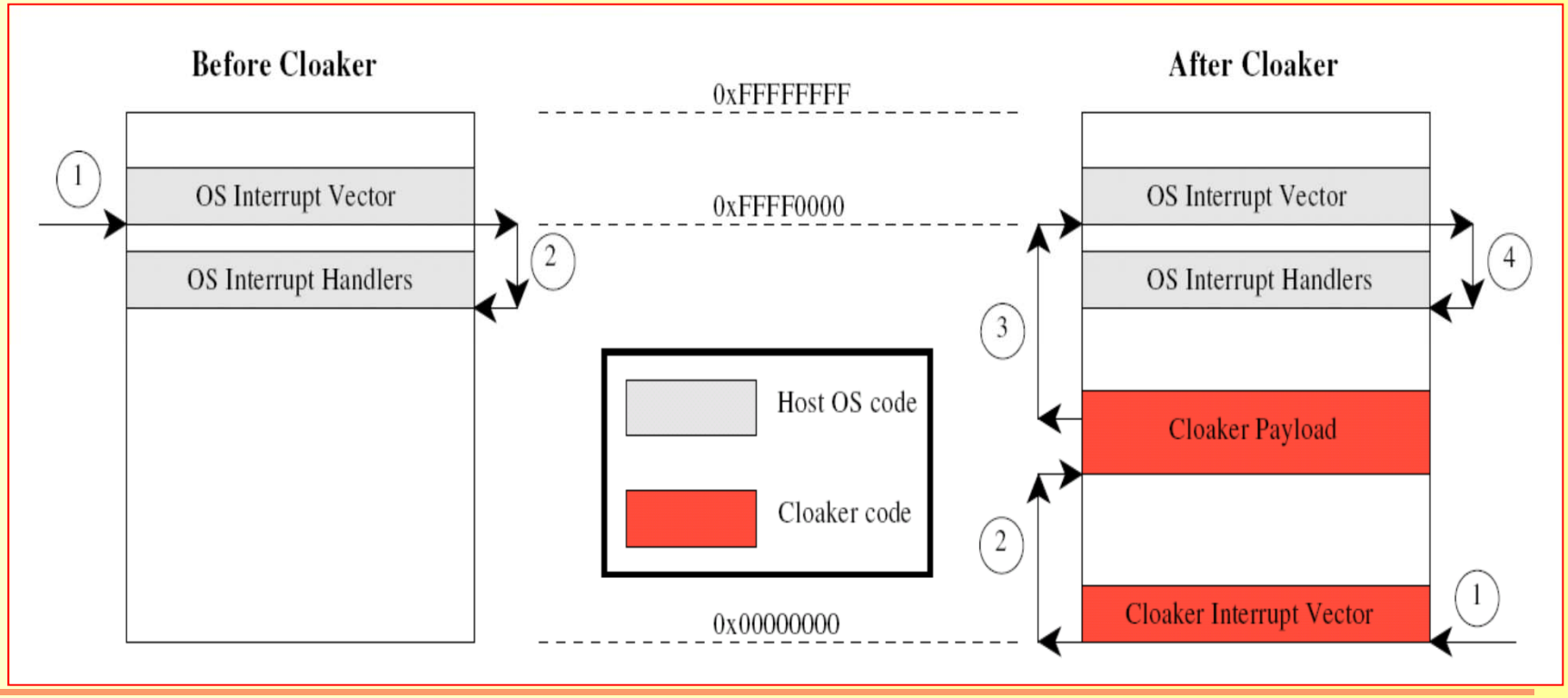
- 对象authenticated有两个“写”，分别在1，8。
- 对象authenticated有两个“读”，分别在4，11。
- 在第11行的“读”的可达集为{1,8}。
- 如果在执行第11行的“读”时可达集超过了{1,8}则报告错误。

基于硬件属性的攻击 — Cloaker

- Cloaker目标:
  - 隐藏自己，避免检测工具检测到。
  - 不修改现有的操作系统的任何代码和数据。
  - 不利用现有操作系统的任何服务。

基于硬件属性的攻击 — Cloaker

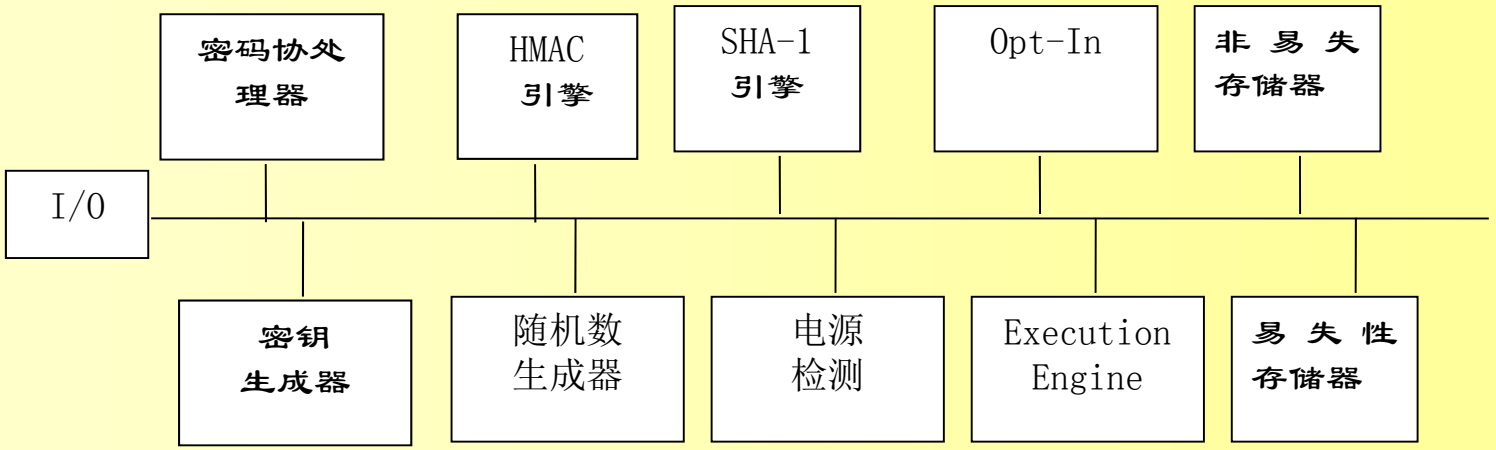
- ARM926EJ-S，协处理器CP15寄存器C1位13
  - 1：中断向量在高端：0XFFFF0000
  - 0：中断向量在低端：0X00000000



Rootkit检测— 完整性检查

- 平台完整性检测
  - TPM可以用来检测BIOS、固件、OS加载代码等的完整性。

TPM的功能特性



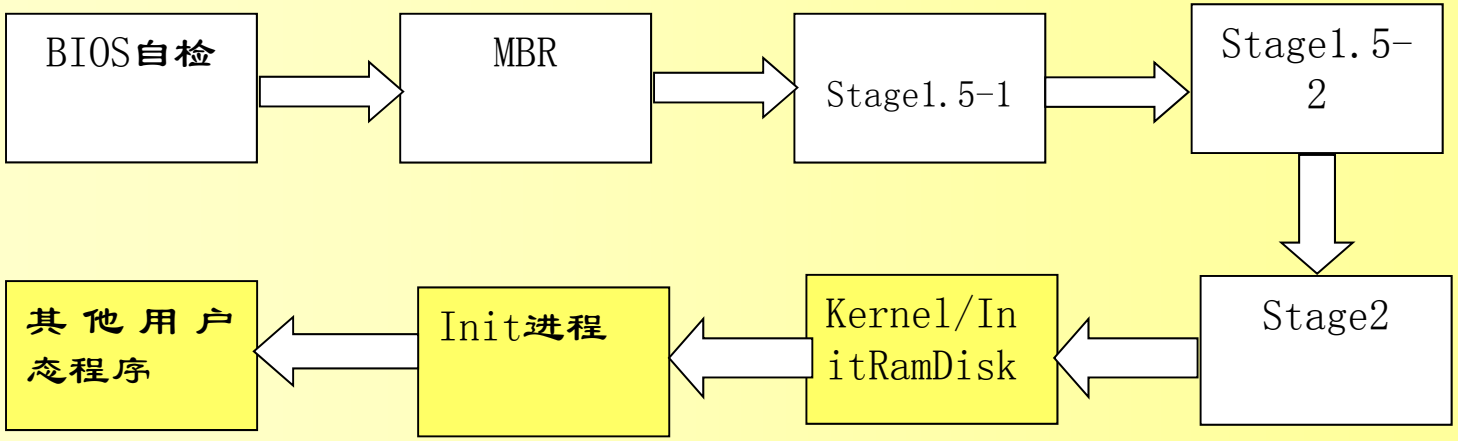
TPM体系结构

TPM的功能特性

- 与主板相连，在系统启动过程中始终存在
- PCR寄存器，只能够通过扩展操作修改
- 随机数生成器，与OS随机池相分离
- 密钥生成器，能够在芯片内部安全的生成密钥
- 签名加密功能
- 远程平台证明，能够向远程平台证明自己
- 数据与PCR寄存器绑定（TPM_SealData、TPM_UnSealData）

子系统结构

传统操作系统的启动流程

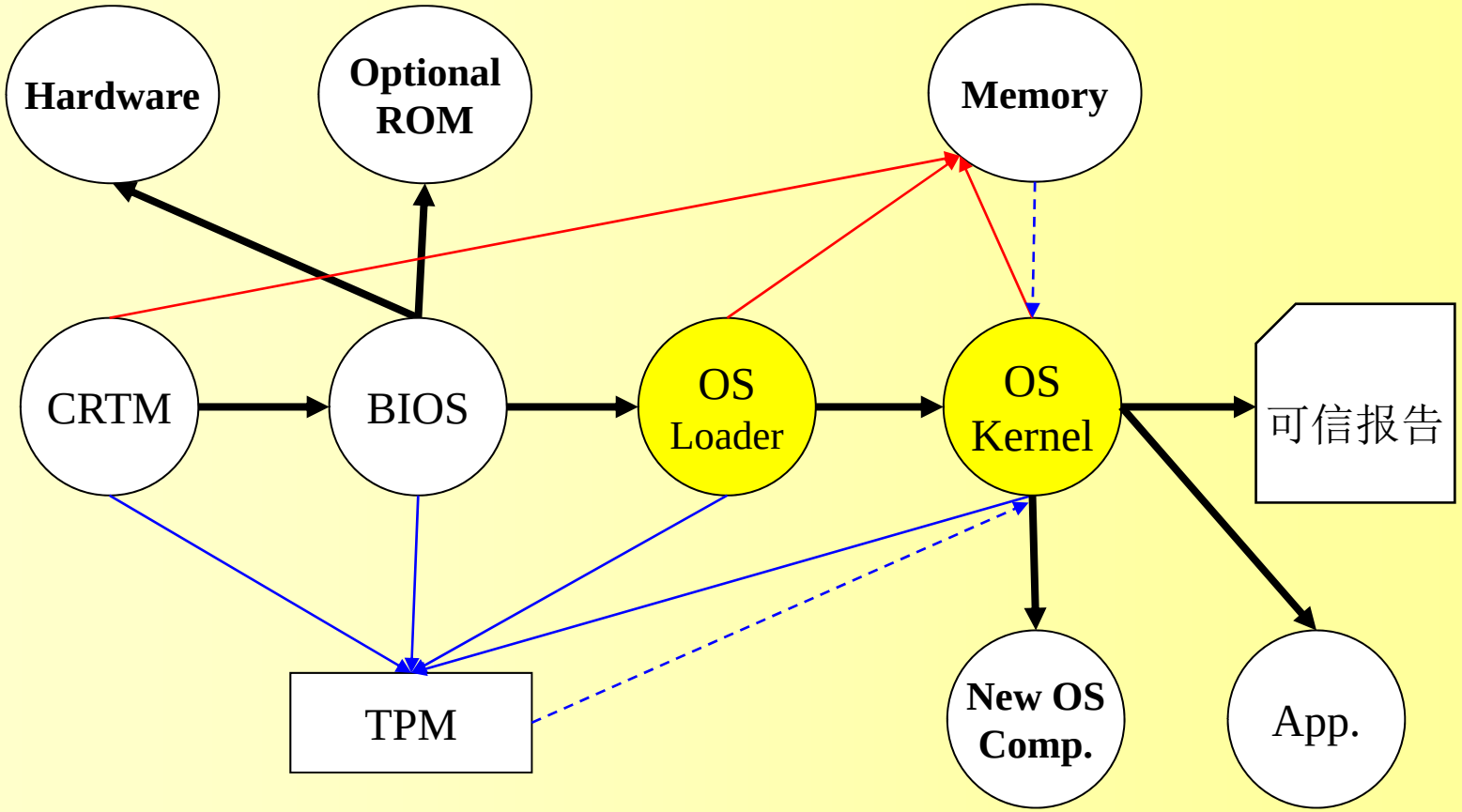


Grub启动链

子系统结构

- 系统的启动过程实际上就是执行代码间控制权的转移。
- 这些代码块的完整性决定了我们系统运行平台是否可信。
- 对启动过程中启动链上的所有元素进行度量，在启动完毕后度量，根据结果能够确定系统的完整性与否。

子系统结构


Hardware
Optional ROM
Memory
CRTM
BIOS
OS Loader
OS Kernel
可信报告
TPM
New OS Comp.
App.

（5） 安全审计

(Common **Criteria**)

- Security audit automatic response
- Security audit data generation
- Security audit analysis
- Security audit event selection
- Security audit event storage

5.1 安全审计的自动响应

- This family defines the response to be taken
  - in case of detected events indicative of a potential security violation.
- The following actions could be considered for the management functions
  - (addition, removal, or modification) of actions
- The following actions should be auditable
  - Actions taken due to imminent security violations.

5.2 安全审计的日志生成 (1)

- 在记录安全相关事件的审计日志方面
  - 考虑所有可审计的事件类型
  - 考虑所有各种审计日志必须记录的信息内容。

5.2 安全审计的日志生成 (2)

Audit data generation

- **The TSF shall be able to generate an audit record of the following auditable events:**
  - **Start-up and shutdown of the audit functions;**
  - **All auditable events for the *minimum or basic or detailed* level of audit;**
- **The TSF shall record within each audit record at least the following information:**
  - **Date and time of the event**
  - **type of event**
  - **subject identity**
  - **and the outcome (success or failure) of the event**
  - ***Other audit relevant information***

5.2 安全审计的日志生成 (3)

- **User identity association**
  - **The TSF shall be able to associate each auditable event with the identity of the user that caused the event.**

5.3 安全审计分析

- 安全审计必须考虑对可能的安全入侵事件相关的系统行为和审计信息的自动分析。
  - 采用入侵检测手段;
  - 对可能要发生的入侵行为的响应。

5.3 Security audit analysis

- **Potential violation analysis**, basic threshold detection on the basis of a fixed rule set is required.
- **Profile based anomaly detection**, the TSF maintains individual *profiles* of system usage, where a profile represents the **historical patterns** of usage performed by members of the *profile target group*.
- **Simple attack heuristics**, the TSF shall be able to detect the occurrence of **signature events** that represent a significant threat to TSP enforcement.
- **Complex attack heuristics**, the TSF shall be able to represent and detect **multi-step** intrusion scenarios.

Security audit event selection

- This family defines the requirements for audit tools that should be available to authorised users to assist in the review of audit data.
  - **The TSF shall provide [assignment: *authorised users*] with the capability to read [assignment: *list of audit information*] from the audit records.**
  - **The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.**
  - **The TSF shall provide the ability to perform [selection: *searches, sorting, ordering*] of audit data based on [assignment: *criteria with logical relations*].**

Security audit event storage

- This family defines the requirements for the TSF to be able to create and maintain a secure audit trail.
  - **The TSF shall protect the stored audit records from unauthorised deletion.**
  - **The TSF shall protect the stored audit records from unauthorised deletion or modifications**
  - **The TSF shall take [assignment: *actions to be taken in case of possible audit storage failure*] if the audit trail exceeds [assignment: *pre-defined limit*].**
  - **Prevention of audit data loss.**

（6） 对象重用保护

- 计算机保持效率的一种方法就是对象重用。
- 通常，文件占用的空间来自于磁盘上先前被用过、但现在已被释放的空间，或其他存储设备上的空间。被释放的空间是“脏”的，也就是说，它仍然包含先前用户的数据。
- 恶意的用户会申请大量磁盘空间，然后从中获取敏感信息。这种攻击被称为对象重用。
- 这个问题包括磁盘、主存、处理器的寄存器、其他磁介质(例如磁带)或者其他可重用的存储媒体。
- 磁介质对于此类攻击尤其脆弱。非常精密和昂贵的仪器有时候能够将最近的数据和它先前记录的数据分开，然后再将后者与后者之前的数据分开，依次类推。这种威胁，称为磁记忆(magetic relnanence) 。
- 在任何情况下，操作系统在允许对资源的访问之前必须负责清除资源上的信息。

(7) 可信信道

- 恶意用户获得不合适访问的一种途径就是“欺骗”用户，使他们认为自己正和一个合法的安全系统在通信，而实际上这时候他们键入的内容以及命令已经被截获且分析了。
- 因此，对于关键的操作，如设置密码或者更改访问许可，用户希望能进行无误的通信(称为可信路径)，以确保他们只向合法的接收者提供这些重要的、受保护的信息。
- 在一些操作系统中，用户通过输入一个惟一的键序列来请求一条可信路径。这个惟一键序列在设计上直接被安全实施软件截获（在Windows登陆时，Ctrl+Alt+<Del>键）。在其他可信系统中，与安全相关的改变只能在系统启动的时候进行，也就是说，改变只能在除安全实施代码外的其他任何进程运行之前进行。