保护和安全

保护

保护域

概念

保护域是操作系统中用于控制进程可以访问哪些对象的安全机制。它提供了一种分离和管理资源访问的方式，以避免不必要的资源暴露或错误操作。

对象 (Object)：

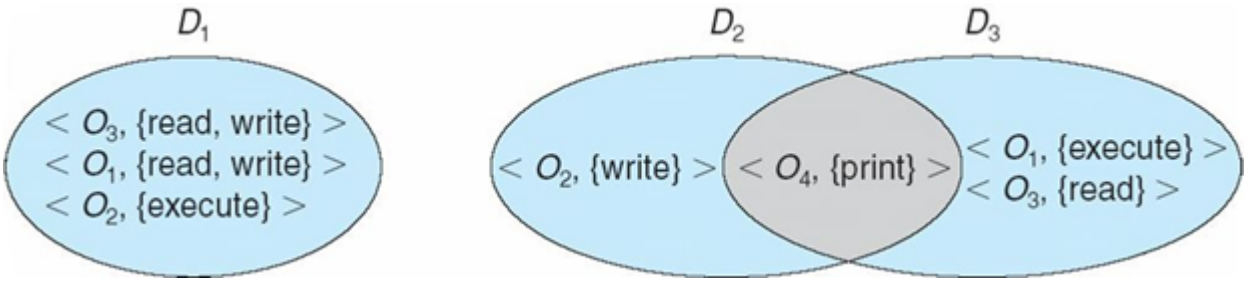
- **硬件对象**： CPU、内存段、打印机、磁盘等。
- **软件对象**： 文件、程序、信号量等。

每个对象都有唯一标识符，且访问方式受到严格约束，例如，文件只能通过读、写等操作访问。

需要知道原则 (Need-to-Know Principle)： 一个进程只能访问当前执行任务所需的对象，而非全部对象。（最小权限原则）

Domain Structure 域结构

- **一个域是访问权限的集合**
  - 访问权限 (Access-right) 定义为：`<object-name, rights-set>`
  - 其中，`rights-set` 是可以对对象执行的所有有效操作的一个子集。
- **域可以是用户 (user)、进程 (process)、过程 (procedure)**：
  - 用户/进程/过程与域之间的关联可以是静态 (static) 或动态 (dynamic) 的。
  - **动态关联**： 允许域切换 (domain switching)。



o->对象 D->Domain

🚀 访问矩阵

《操作系统笔记》--访问矩阵_访问控制矩阵-CSDN博客

概念

一种通用保护模型

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

将保护机制视为矩阵 (access matrix)：

- 行 (rows) 表示域 (domains)。
- 列 (columns) 表示对象 (objects)。
- 矩阵中的元素 **Access(i, j)** 表示在域 **D**i 中执行的进程可以对对象 **O**j 执行的一组操作。

访问矩阵的使用

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

如果一个进程在域 D_i 中尝试对对象 O_j 执行操作 op，则 op 必须在访问矩阵中定义。

创建对象的用户可以为该对象定义访问矩阵的列。

可以扩展为动态保护机制：

- 操作包括添加和删除访问权限。
- 特殊访问权限：
  - Owner（所有者）：O_i 的所有者。

- Copy（拷贝）： 将操作 op 从对象 O_i 拷贝到对象 O_j（用 * 表示）。
- Control（控制）： 域 D_i 可以修改域 D_j 的访问权限。
- Transfer（切换）： 从域 D_i 切换到域 D_j。

Copy 和 Owner 权限适用于具体对象。 Control 权限适用于域对象。

访问矩阵条目内容的受控更改需要三个附加操作： **复制(copy)**， **所有者(owner)**，**控制(control)** 。

Copy权限

复制访问矩阵的一个域(或行)的访问权限到另外一个的能力。通过访问权限后面附加的星号 “*” 来标记。

分清楚是与文件F关联的任何条目，而不是与域D关联的任何条目

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

如图，权限 R* 被从访问矩阵 access(i,j) 拷贝到 access(k,j)，只有权限 R（而不是 R*）被创建

右侧表格中，域 D1 已将 read 权限复制给 D3，但复制后的权限不包含 "*"（不可再被进一步复制）

Owner权限

所有者权限控制增加新的权限和取消某些权限这些操作。

如果access(i,j)包括所有者权限，则执行在域D~i~中的进程可以增加和删除列j的任何条目的任何权限。

注意是列j的任何条目任何权限

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

域D~1~为F~1~的所有者，并且可以增加和删除列F~1~的任何有效权限。 同样，域D~2~为F~2~与F~3~的所有者，因此可以增加和删除这两个列的任何有效权限。

Control权限

Control 控制权限允许域 Di 修改其他域 Dj 的访问权限

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

New:

Old:

D_4	read write		read write		switch			

假设access(D ~2~ , D ~4~)包含控制权限，那么，执行在域D~2~内的进程可以修改域D~4~

🚀 访问矩阵的实现

访问矩阵通常是稀疏矩阵

全局表

```
<D1, F1, {read, write}>
<D2, F1, {read}>
<D3, F2, {execute}>
```

当某个域（比如 D1）对某个对象（比如 F1）尝试操作（比如 write），操作系统会直接在全局表中查找。

如果找到匹配的记录 `<D1, F1, {read, write}>`，就允许操作；如果找不到，就拒绝访问。

ppt:

- 全局表是访问矩阵的最简单实现，它包括一组有序三元组<domain, object, rights-set>。
- 当在域Di内对对象Oj进行操作M时，就在全局表中查找三元组<Di, Oj, Rk>，其中操作M属于Rk。
- 如果遇到这个三元组，则操作允许继续，否则，就会引起异常或者错误。
- 缺点：表通常很大，所以不能存放在内存中，所以需要额外的I/O操作。

对象访问列表

```
F1:
- D1: {read, write}
- D2: {read}
- D3: {execute}
```

当域 D1 想要操作 F1 时，系统只需要查找 F1 的访问列表即可，而不是遍历所有记录。

ppt:

每个列实现为一个对象的访问列表

Domain1-Read, Write

Domain2-Read

Domain3-Read

- 访问矩阵的每个列，可以实现为一个对象的访问列表。
- 每个对象的访问列表包括一组有序对<domain, rights-set>，以定义具有非空访问权限集合的那些域。

- 当在域Di中对对象Oj尝试操作M时，搜索对象Oj的访问列表，查找条目<Di, Rk>，其中M属于Rk。如果找到条目，则允许操作，如果没有找到，则检查默认集合。如果默认集合有M， 则允许访问，否则，拒绝访问，并引起异常。

域的能力列表

```
D1:
- F1: {read, write}
- F4: {read, write, execute}
- F5: {read, write, delete, copy}
```

当域 D1 想要访问 F4 时，系统检查 D1 的能力列表即可。

ppt:

不基于对象，而是基于域的列表。

- **能力列表（Capability list）** 是一个域相关的对象列表，以及允许对这些对象执行的操作。
  - 对象 F1 – 读取（Read）
  - 对象 F4 – 读取、写入、执行（Read, Write, Execute）
  - 对象 F5 – 读取、写入、删除、拷贝（Read, Write, Delete, Copy）
- 对象由其物理名称或地址表示，称为 **能力（Capability）** 。
- 执行对对象 Oj 的操作 M 时，进程请求操作，并将能力（或指针）作为参数传递。
  - 拥有能力表示允许访问。
- 能力列表与域相关联，但不能直接被域访问。
  - 而是作为受保护的对象，由操作系统维护，并通过间接方式访问。

三者的主要区别：

方法	存储方式	查找范围	适用场景	优缺点总结
全局表	全部权限存储在一张表中	查找全局表	权限关系少，数据量小的系统	简单但效率低，表太大不适用
对象访问列表	每个对象有独立的访问列表	查找指定对象的列表	文件系统，集中管理对象的权限	对象多时存储空间开销较大
能力列表	每个域有独立的能力列表	查找指定域的能力列表	用户权限管理，域权限固定的场景	保护能力列表防止篡改是关键

安全

1. 安全问题的核心

- 定义安全性： 如果资源在各种环境下都能按照预期方式被访问，则系统是安全的。
- 常见安全威胁：
  - 入侵者（Intruders）尝试破坏系统安全。
  - 威胁（Threat）： 潜在的安全违规行为。
  - 攻击（Attack）： 意图破坏安全的行为。

2. 安全违规类型

- 机密性破坏（Breach of Confidentiality）： 未经授权的数据读取。
- 完整性破坏（Breach of Integrity）： 未经授权的数据修改。
- 可用性破坏（Breach of Availability）： 未经授权的数据销毁。
- 服务窃取（Theft of Service）： 未经授权使用资源。
- 拒绝服务（Denial of Service, DoS）： 阻止合法用户使用资源。

3. 安全违规的方法

- 伪装（Masquerading）： 冒充合法用户以获取更高权限。
- 重播攻击（Replay Attack）： 以原样或修改后的形式重放数据。
- 消息篡改（Message Modification）： 在通信过程中修改数据。
- 中间人攻击（Man-in-the-Middle Attack）： 攻击者伪装成发送方或接收方窃取数据。
- 会话劫持（Session Hijacking）： 截获已建立的会话，绕过身份验证。

4. 程序威胁

- 恶意软件（Malware）：
  - 特洛伊木马（Trojan Horse）
  - 间谍软件（Spyware）
  - 勒索软件（Ransomware）
  - 后门（Trap Door）
  - 逻辑炸弹（Logic Bomb）

- 代码注入攻击（Code-Injection Attack）：
  - 缓冲区溢出（Buffer Overflow）
- 病毒和蠕虫：
  - 文件病毒
  - 宏病毒
  - 源代码病毒
  - 加密病毒

5. 安全防御

- 安全模型层次：
  - 应用层：沙盒、软件限制。
  - 操作系统层：补丁、配置加固。
  - 网络层：加密、身份验证。
  - 物理层：安全设备、防护措施。
- 防御方法：
  - 安全策略：定义被保护的内容。
  - 漏洞评估：评估实际状态与安全策略之间的差距。
  - 入侵检测：检测成功或未遂的攻击。
  - 病毒防护：防止恶意软件感染。
  - 审计与日志：跟踪系统行为。
  - 防火墙：控制网络流量。

6. 防火墙及系统威胁

- 防火墙：位于可信与不可信网络之间，限制访问。
  - 个人防火墙：监控流量。
  - 应用代理防火墙：控制特定协议。
- 网络攻击：
  - 网络流量攻击（例如中间人攻击）。
  - 拒绝服务攻击（DoS）。
  - 端口扫描：探测系统漏洞。