



操作系统 Operation System

第八章 安全

沃天宇

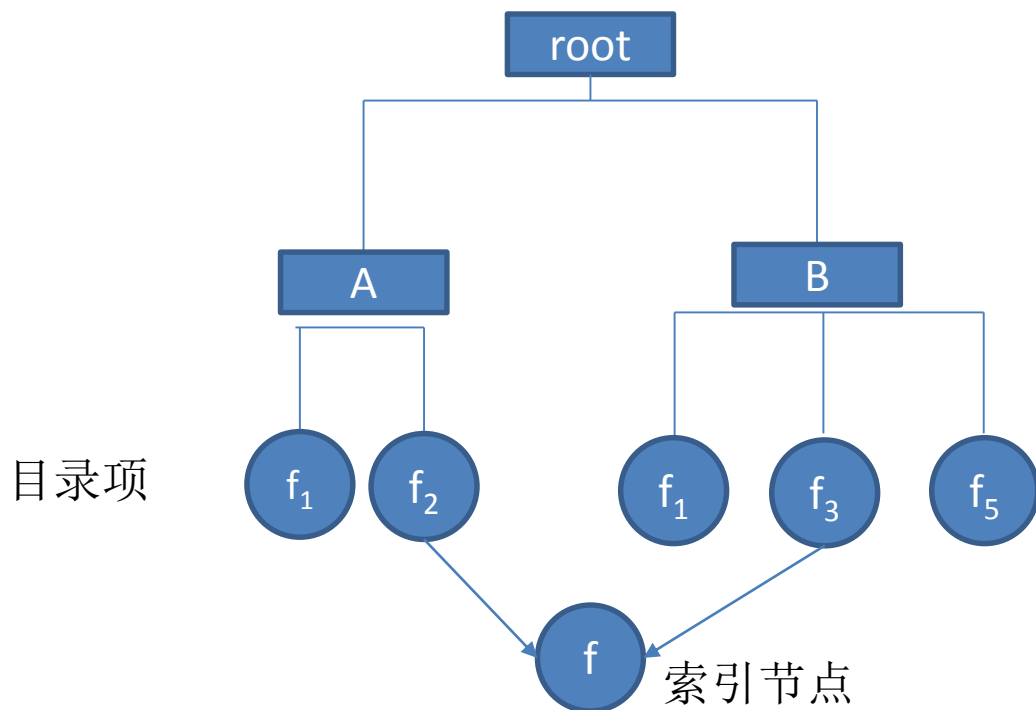
woty@buaa.edu.cn



练习1

- 用户A有两个文件 f_1 和 f_2 ，用户B有3个文件 f_1 、 f_3 和 f_5 。其中用户A的文件 f_1 与用户B的文件 f_1 不是同一个文件，用户B的文件 f_3 和用户A的文件 f_2 是同一个文件。请针对以上需求，给出一种目录组织方案。可画图辅助说明。

- 答案：

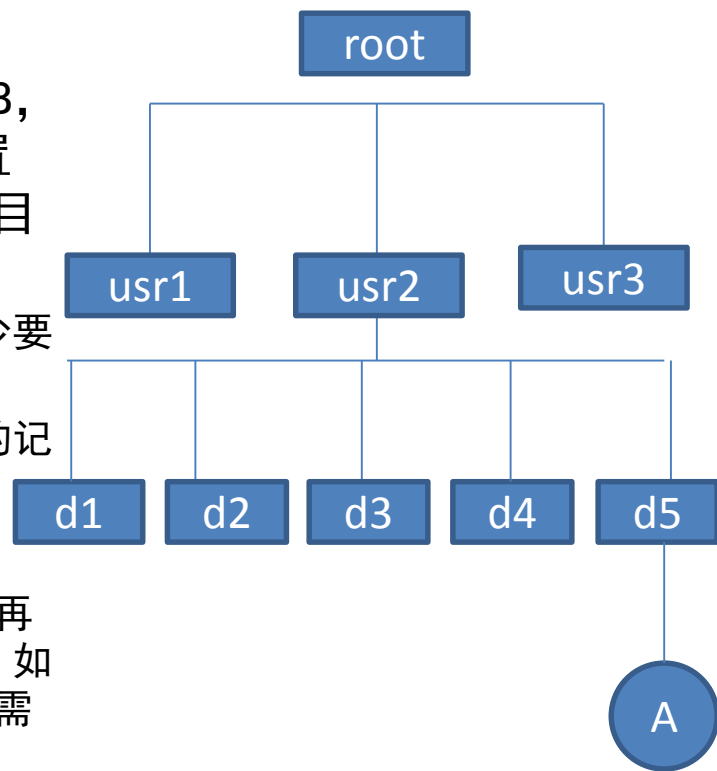


练习2

- 有一个磁盘文件系统，文件由512B大小的块组成。每个文件有一个目录项，该目录项包含文件名、文件长度、第一块和最后一个块的位置（如果是索引结构的文件，该目录项指明第一个索引块，一个索引块指向511个文件块，其中包含下一个索引块的指针）。针对连续、串联结构和索引结构的文件，分别说明在系统中如何实现逻辑地址到物理地址的映射。如果当前位于逻辑块10，且下次希望访问逻辑块4，针对三种类型的文件分别说明需从磁盘上读取多少个物理块。
- 答案：
 - 连续文件。第 m 逻辑块物理块号为：起始物理块+ $m-1$ 。因此需要从磁盘读取1块即可。
 - 串联文件。从第一个物理块开始沿着链表进行查找，直至找到目标物理块。因此需要从磁盘读取4块。
 - 索引文件。先读第一个索引块，如果逻辑块位于前510个块，则直接从索引项中找到物理块号；否则，读取下一级索引块，进行查找。由于已经读取了逻辑块10，因此第一个索引块已经在内存，逻辑块4所对应的物理块可以直接在索引项中找到，只需从磁盘上读取1块即可。

练习3

- 某文件系统以硬盘作为文件存储器，物理块大小为512B。有文件A包括490个逻辑记录，每个记录占255B，每个物理块存2个记录。文件A在该文件目录中的位置如图所示。每个目录项占127B，每个物理块存放4个目录项。根目录内容常驻内存。
 - 若文件采用链接分配方式，如果要将文件A读入内存，至少要存取几次硬盘？
 - 若文件采用连续分配方式，如果要读取逻辑记录号为480的记录，至少要存取几次硬盘？



- 答案：**
 - 采用链接分配：先读取第二级目录，再读取第三级目录，再读取 $490/2=245$ 个物理块。因此最少需要 $1+1+245=247$ 次。如果读取第三级目录时，第一次未读到包括d5的索引块，就需要读两个物理块。因此最多需要 $1+2+245=248$ 次。
 - 采用连续分配方式：知道第一块的物理块号就可直接计算出480记录所在的块。一个块可放两个记录，因此480号记录放在第240磁盘块。最少需要 $1+1+1=3$ 次；最多需要 $1+2+1=4$ 次。

内容提要

- 操作系统的安全问题
- 安全标准及安全等级
- 常见问题、方法与机制

安全需求

- **计算机信息系统安全性**

- 机密性。指防止信息的非授权修改，这也是信息安全最重要的要求。
- 完整性。要求信息在存储或传输过程中保持不被修改、破坏和丢失。
- 可用性。指当需要时是否能存取所需信息，保护信息的可用性的任务就是防止信息失效或变得不可存取。
- 可靠性。指系统提供信息的可信赖程度。

- **操作系统安全的目标**

- 为用户信息处理提供安全的软件环境，为应用程序运行提供安全可靠的运行环境。

操作系统安全的概念

- **操作系统是信息系统安全的基础之一**
 - 操作系统是软件系统的底层
 - 操作系统是系统资源的管理者
 - 操作系统是软硬件的接口
- **操作系统安全的困难性**
 - 操作系统的规模庞大，以至于不能保证完全正确
 - 理论上一般操作系统的安全是不可判定问题
- **必须拥有自己的操作系统**
 - 操作系统受治于人，不能从根本上确保信息安全
 - 立足国内，发展自己的操作系统

实例

- 美国英特尔公司：在其生产的奔腾III处理器中加入了一种唯一序列码，可通过计算机的“唯一识别”功能用来识别并自动跟踪用户，读取用户信息；
 - 微软公司：“Windows 98”系统软件在编制时疑似留有一个具有超级特权的用户账号，微软总部通过这个账号可以随时畅通无阻地进入用户操作系统。
-

操作系统的安全需求

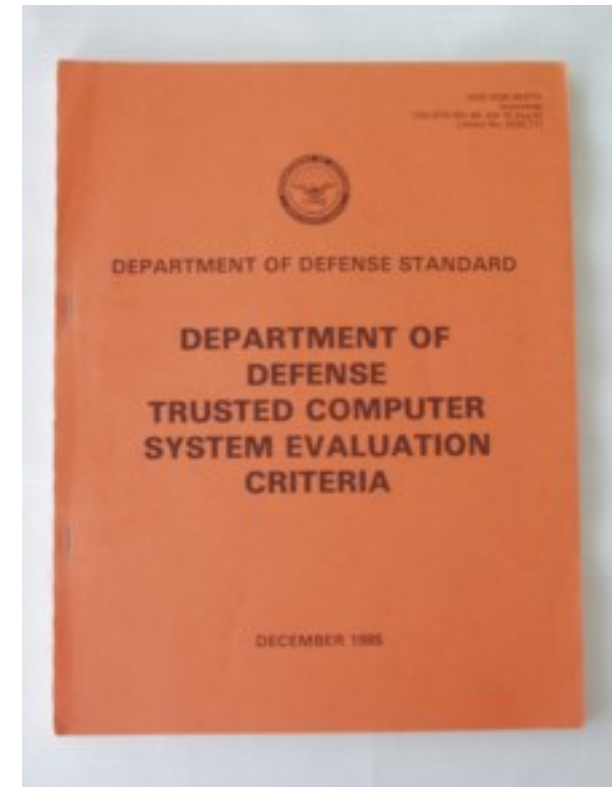
- **系统边界安全**
 - 认证和鉴别禁止非法用户进入系统；
- **系统使用权限管理机制**
 - 不同用户配置不同的权限，每个用户只拥有他能够工作的最小权利；
- **应用和数据的访问控制机制**
 - 用户只能按照指定的访问控制安全策略访问数据；
- **为系统用户提供可信通路**
 - 保证系统登录和应用层提供的安全机制不被旁路；
- **系统操作的安全审计和管理**
 - 检查错误发生的原因，或者受到攻击时攻击者留下的痕迹；

内容提要

- 操作系统的安全问题
- 安全标准及安全等级
- 常见问题、方法与机制

美国TCSEC标准

- 美国国防部于1983年提出“可信计算机系统评估准则”TCSEC，评估计算机安全最有名的标准之一。
- TCSEC将安全保护分成四等、八个级别，分别为D，C1，C2，B1，B2，B3，A1，超A1，安全级渐次增强。
- 不满足任何较高级别安全可信条件的系统划入D类。




- C 类为自主型保护，由两个级别组成。
 - C1级：具有一定的自主型存取控制机制。象UNIX的OWNER/GROUP/OTHER 存取控制。
 - C2级：具有更细粒度（到每一个单独用户）的自主型存取机制，而且引入审计机制。
-

- B类为强制型保护，由三个级别组成：
 - B1级：满足C2级所有的要求，而且需要有所用安全策略模型的非形式化描述，实施强制型存取控制。
 - B2级：系统的可信计算基(TCB)是基于明确定义的形式化模型，并对系统中所有的主体和客体实施了自主型存取控制和强制型存取控制。另外，具有可信通路机制、系统结构化设计、最小特权管理以及对隐通道的分析和处理等。
 - B3级：系统的TCB设计要满足能对系统中所有的主体对客体的访问进行控制，TCB不会被非法篡改，且TCB设计要非常的小巧和结构化以便于分析和测试其正确性。支持安全管理者的实现，审计机制能实时报告系统的安全性事件，支持系统恢复。
-

- A类为验证型保护，由两个级别组成：
 - A1级：从实现的功能上看，它等同于B3。它的特色在于形式化的顶层设计规格（FTDS）、形式化验证FTDS与形式化模型的一致性和由此带来的更高的可信度；
 - A1以上级：比A1级可信度更高的系统归入该级。
-

TCSEC标准分级



A 级	▶	形式化校验级保护，波音SNS
B3 级	▶	安全内核，高抗渗透能力，Trusted Mach
B2 级	▶	结构化内容保护，形式化安全策略模型，MULTICS
B1 级	▶	标记安全保护，如System V等
C2 级	▶	C1+访问控制，广泛审核，Linux/WINDOWS NT
C1 级	▶	主体自主决定的安全保护，UNIX/WINDOWS
D 级	▶	不分等级，无口令和权限控制，MS DOS

中国计算机安全标准

- 对应于TCSEC，我国于九十年代提出自己的计算机安全等级国家标准，称为《计算机信息系统安全保护等级划分准则》（GB 17859-1999），将计算机信息系统的安全等级划分为五级：
 - 第一级：用户自主保护级-C1
 - 第二级：系统审计保护级-C2
 - 第三级：安全标记保护级-B1
 - 第四级：结构化保护级-B2
 - 第五级：访问验证保护级-B3
-

其他相关标准

1993年6月，美国政府同加拿大及欧共体共同起草单一的通用准则（CC标准）并将其推到国际标准。制定CC标准的目的是建立一个各国都能接受的通用的信息安全产品和系统的安全性评估准则。在美国的TCSEC、欧洲的ITSEC、加拿大的CTCPEC、美国的FC等信息安全准则的基础上，由6个国家7方（美国国家安全局和国家技术标准研究所、加、英、法、德、荷）共同提出了“信息技术安全评价通用准则（The Common Criteria for Information Technology security Evaluation, CC）”，简称CC标准，它综合了已有的信息安全的准则和标准，形成了一个更全面的框架。

GB/T 18336《信息技术安全性评估准则》等同采用 国际标准 15408（CC）

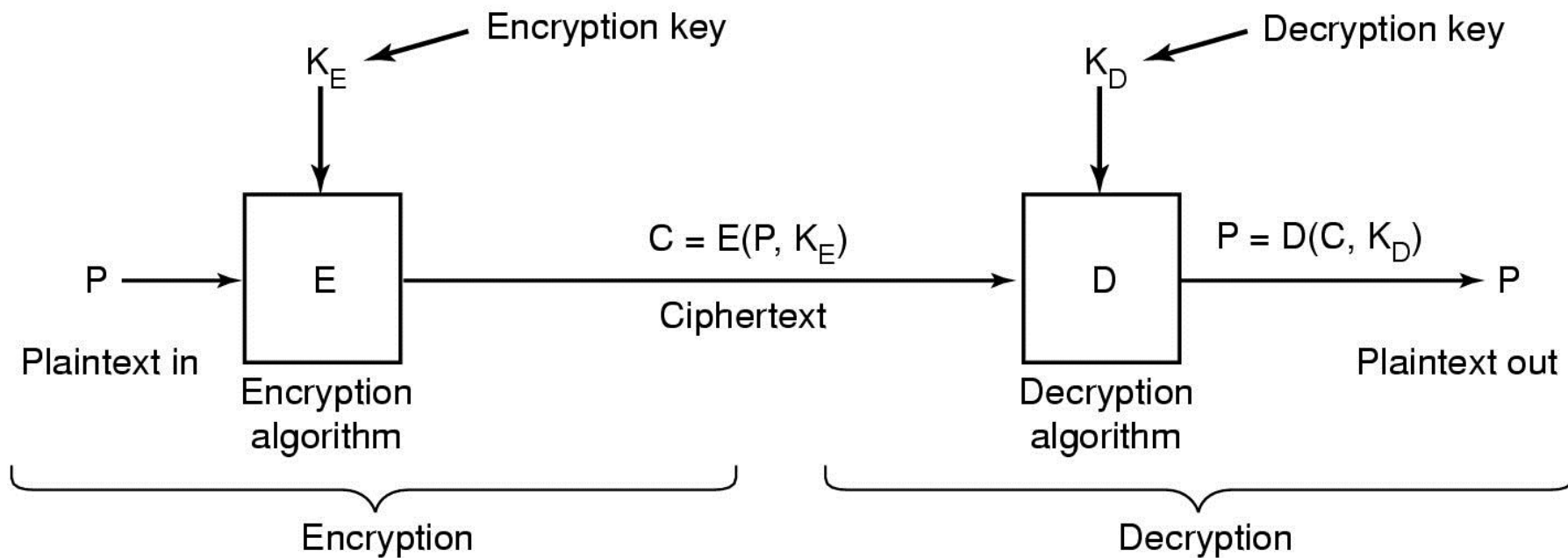
在GB/T 18336 国际标准 15408（CC）中定义了以下7个评估保证级：

- (1) 评估保证级1(EAL1)——功能测试；
 - (2) 评估保证级2(EAL2)——结构测试；
 - (3) 评估保证级3(EAL3)——系统地测试和检查；
 - (4) 评估保证级4(EAL4)——系统地设计、测试和复查；
 - (5) 评估保证级5(EAL5)——半形式化设计和测试；
 - (6) 评估保证级6(EAL6)——半形式化验证的设计和测试；
 - (7) 评估保证级7(EAL7)——形式化验证的设计和测试。
-

内容提要

- 操作系统的安全问题
- 安全标准及安全等级
- 常见问题、方法与机制
 - 密码学原理
 - 常见保护机制
 - 用户身份认证
 - 内部攻击方法
 - 代码漏洞攻击
 - 恶意软件简介
 - 常见防御方法

密码学基础



数据加密

对称密钥： Secret-Key Cryptography

基于字母替代的对称密钥算法:

明文Plaintext: ABCDEFGHIJKLMNOPQRSTUVWXYZ

密文Ciphertext: QWERTYUIOPASDFGHJKLZXCVBNM

ATTACK
QZZQEA

主要处理方法： 易位、置换

代表性算法： 美国NIST的DES算法、AES算法

THIS IS A SECRET

ZIOL OL Q LTEKTZ

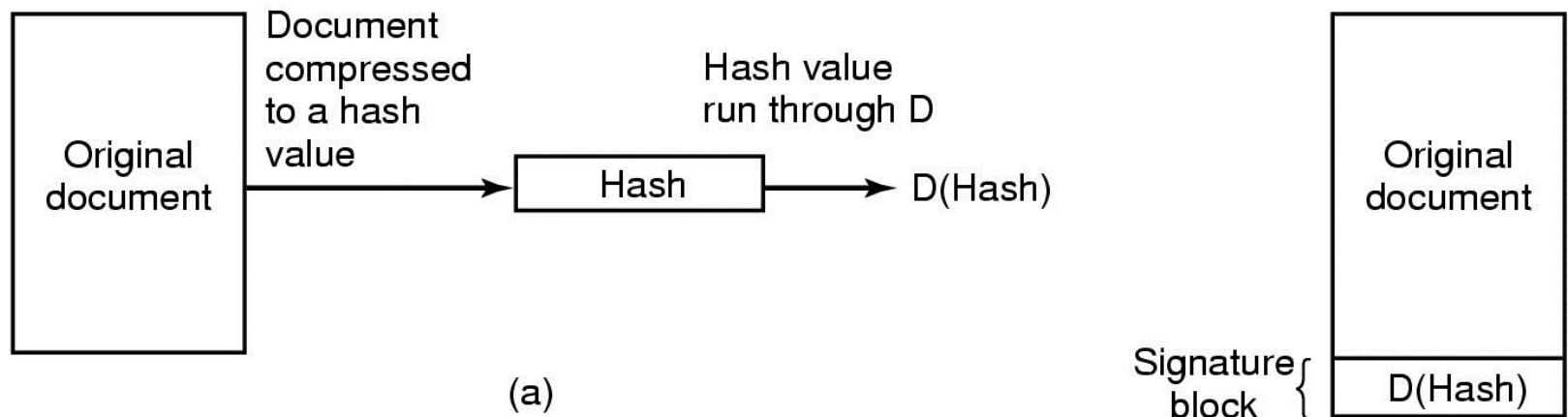
(字频攻击)

非对称密钥：Public-Key Cryptography

- Encryption makes use of an "easy" operation, such as how much is $314159265358979 \times 314159265358979$?
- Decryption without the key requires you to perform a hard operation, such as what is the square root of $3912571506419387090594828508241$?
- Function such that given formula for $f(x)$
 - easy to evaluate $y = f(x)$
- But given y ,
 - computationally infeasible to find x

代表性算法：RSA算法

数字签名： Digital Signatures



散列函数：MD5、SHA
公钥、私钥如何使用？

用户身份认证

登录系统之前，系统需要了解的用户信息：

- 用户知道的信息
 - 用户拥有的信息
 - 用户是谁？
-

基于密码的身份认证

LOGIN: mitch
PASSWORD: FooBar!-7
SUCCESSFUL LOGIN

(a)

LOGIN: carol
INVALID LOGIN NAME
LOGIN:

(b)

LOGIN: carol
PASSWORD: Idunno
INVALID LOGIN
LOGIN:

(c)

Figure 9-17. (a) A successful login.

(b) Login rejected after name is entered.

(c) Login rejected after name and password are typed.

口令简单：黑客侵入系统

```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

口令字典

Figure 9-18. How a cracker broke into a U.S. Department of Energy computer at LBL.

UNIX Password Security

Bobbie, 4238, e(Dog, 4238)
Tony, 2918, e(6%%TaeFF, 2918)
Laura, 6902, e(Shakespeare, 6902)
Mark, 1694, e(XaB#Bwcz, 1694)
Deborah, 1092, e(LordByron,1092)

n位的随机数

Figure 9-19. The use of salt to defeat precomputation of encrypted passwords.

一次性口令

- 单向函数 $y=f(x)$
 - given x it is easy to find y , but given y it is computational infeasible to find x .
 - The input and output should be the same length, i.e. 128 bits
 - $p_{i-1} = f(p_i)$
 - $p_0 = f(p_1), p_1 = f(p_2), p_2 = f(p_3), p_3 = f(p_4)$
 - $p_1 = f(f(f(f(s))))$, $p_2 = f(f(f(s)))$, $p_3 = f(f(s))$, $p_4 = f(s)$

挑战-响应认证

The questions should be chosen so that the user does not need to write them down.

Examples:

- Who is Marjolein's sister?
- On what street was your elementary school?
- What did Mrs. Woroboff teach?
- 更加复杂的请求、响应认证
 - client get r , send $f(r,k)$

使用实际物体进行认证

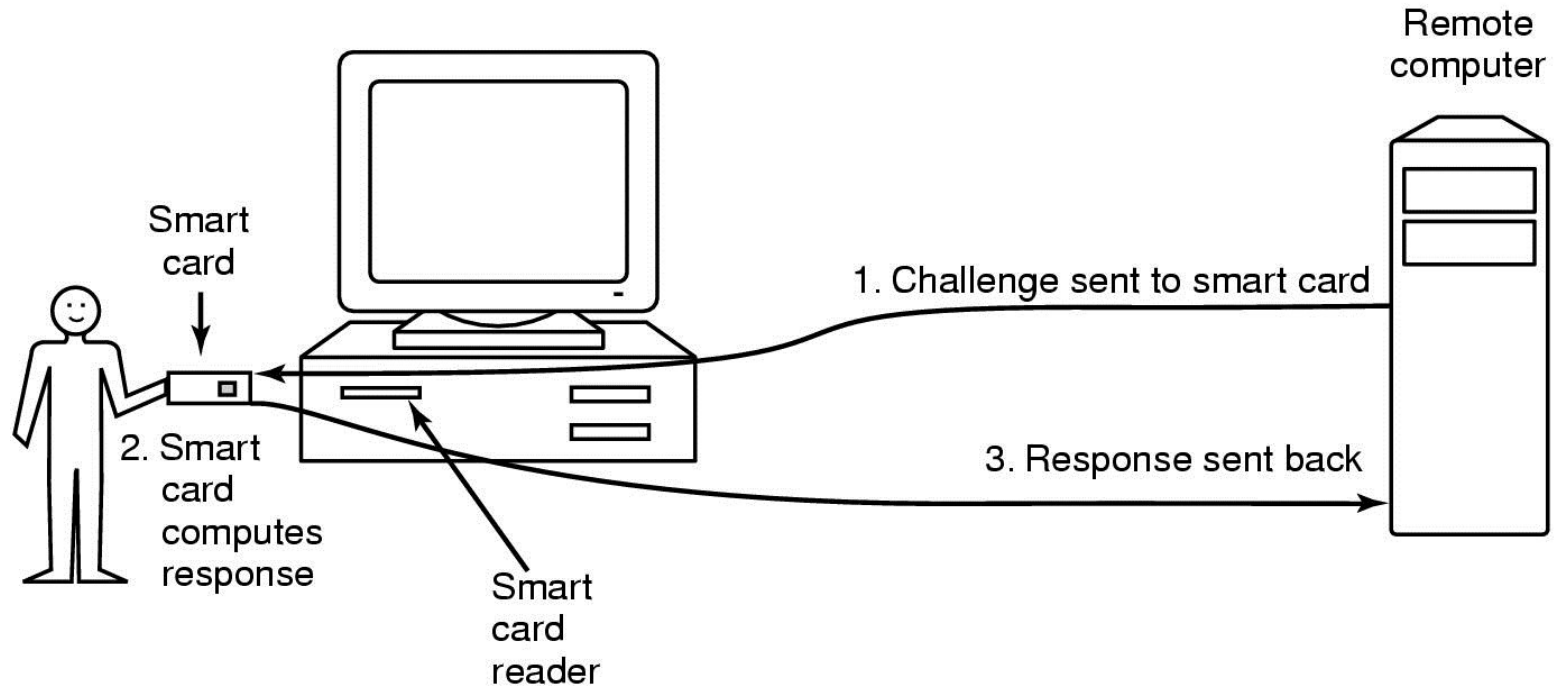


Figure 9-20. Use of a smart card for authentication.

基于生物特征识别的认证

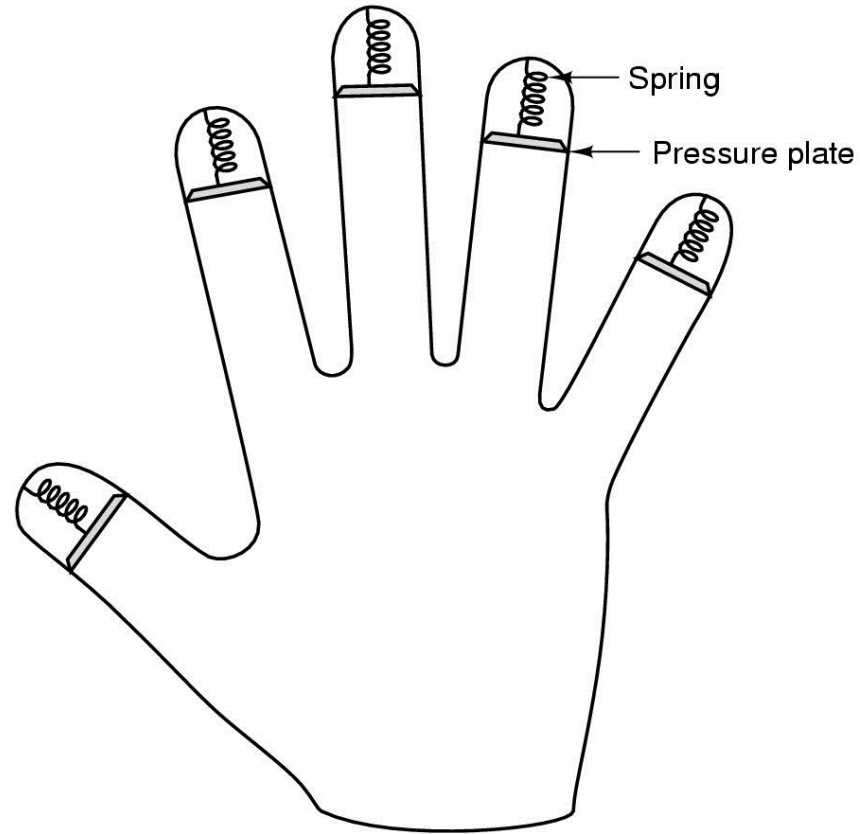


Figure 9-21. A device for measuring finger length.

内部攻击

- 逻辑炸弹: logic bomb
 - 后门陷阱: trap doors
 - 登录欺骗: login spoofing
-

Trap Doors

```
while (TRUE) {  
    printf("login: ");  
    get_string(name);  
    disable_echoing( );  
    printf("password: ");  
    get_string(password);  
    enable_echoing( );  
    v = check_validity(name, password);  
    if (v) break;  
}  
execute_shell(name);
```

(a)

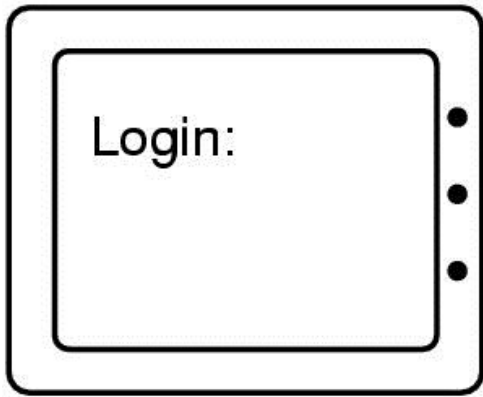
```
while (TRUE) {  
    printf("login: ");  
    get_string(name);  
    disable_echoing( );  
    printf("password: ");  
    get_string(password);  
    enable_echoing( );  
    v = check_validity(name, password);  
    if (v || strcmp(name, "zzzzz") == 0) break;  
}  
execute_shell(name);
```

(b)

(a) 正常代码.

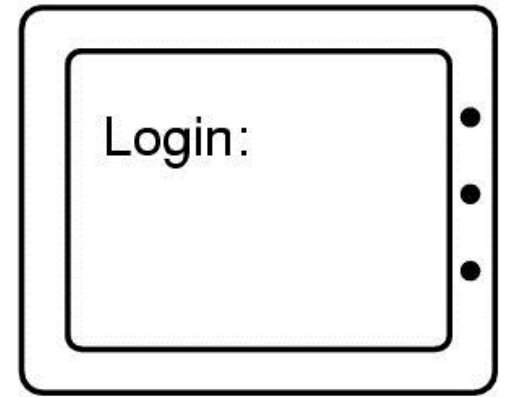
(b) 插入后门的代码示例

Login Spoofing



(a)

(a) 正常的登录界面.



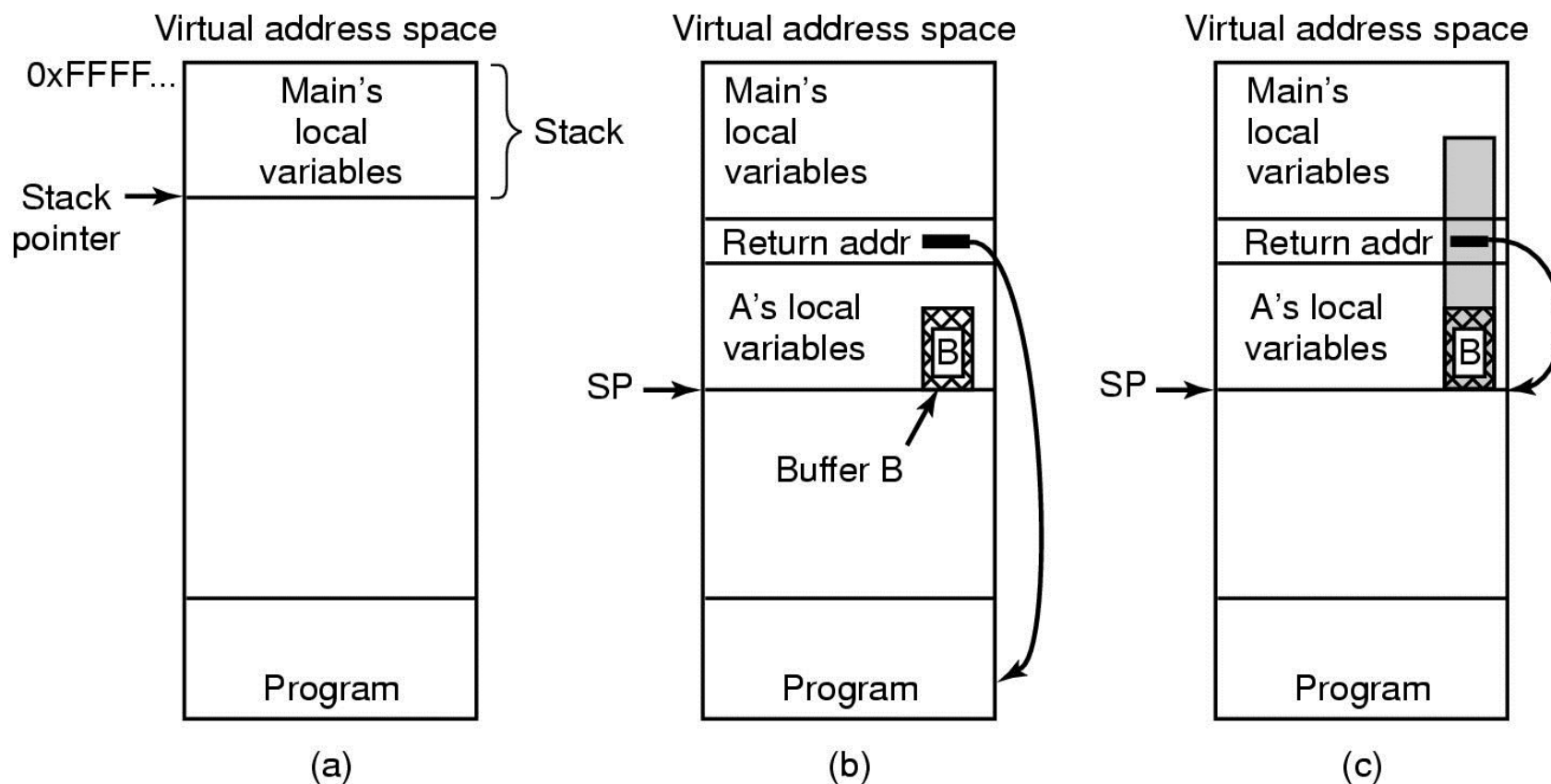
(b)

(b) 仿冒的登录界面

外部攻击：利用代码漏洞

- 缓冲区溢出攻击
 - 格式化字符串攻击
 - 返回libc攻击
 - 整数溢出攻击
 - 代码注入攻击
 - 权限提升攻击
-

缓冲区溢出攻击: Buffer Overflow Attacks

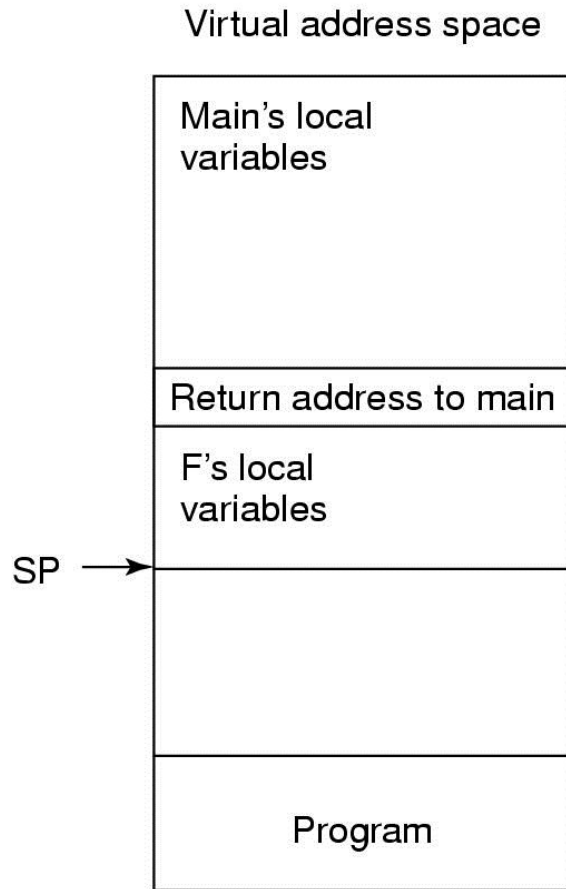


(a) 主程序运行的情况.

(b) 过程A调用之后

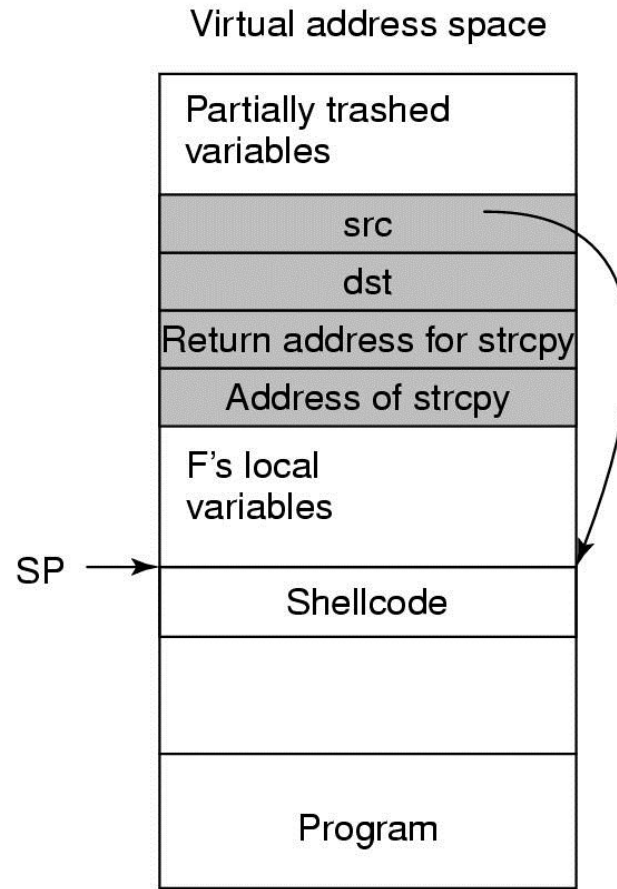
(c) 灰色部分表示缓冲区溢出

返回到libc的攻击: Return to libc Attacks



(a)

(a) 攻击之前的栈情况



(b)

(b) 攻击之后栈被重写

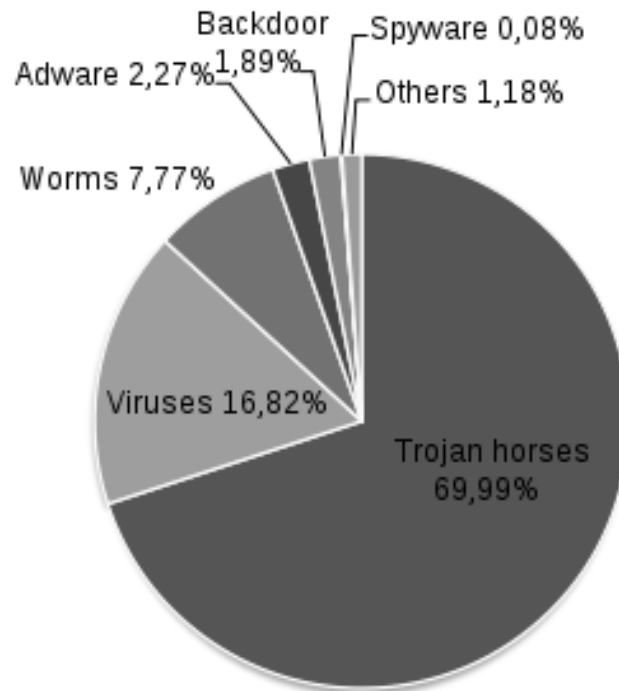
代码注入攻击： Code Injection Attacks

```
int main(int argc, char *argv[])
{
    char src[100], dst[100], cmd[205] = "cp ";
    printf("Please enter name of source file: ");
    gets(src);
    strcat(cmd, src);
    strcat(cmd, " ");
    printf("Please enter name of destination file: ");
    gets(dst);
    strcat(cmd, dst);
    system(cmd);
}
```

/* declare 3 strings */
/* ask for source file */
/* get input from the keyboard */
/* concatenate src after cp */
/* add a space to the end of cmd */
/* ask for output file name */
/* get input from the keyboard */
/* complete the commands string */
/* execute the cp command */

可能导致代码注入攻击的代码片段

恶意软件：Malware



Malware by categories

March 16, 2011



Greetings from General Encryption

To purchase a decryption key for your hard disk, please send \$100 in small unmarked bills to Box 2154, Panama City, Panama.
Thank you. We appreciate your business.

病毒的种类

- 共事者病毒: Companion virus
- 可执行程序病毒: Executable program virus
- 寄生病毒: Parasitic virus
- 内存驻留病毒: Memory-resident virus
- 引导扇区病毒: Boot sector virus
- 设备驱动程序病毒: Device driver virus
- 宏病毒: Macro virus
- 源代码病毒: Source code virus

可执行程序病毒(1)

```
#include <sys/types.h> /* standard POSIX headers */
#include <sys/stat.h>
#include <dirent.h>
#include <fcntl.h>
#include <unistd.h>
struct stat sbuf; /* for lstat call to see if file is sym link */

search(char *dir_name)
{
    DIR *dirp; /* recursively search for executables */
    struct dirent *dp; /* pointer to an open directory stream */
    /* pointer to a directory entry */

    dirp = opendir(dir_name); /* open this directory */
    if (dirp == NULL) return; /* dir could not be opened; forget it */
    ...
}
```

在Unix系统中查找可执行文件的递归过程

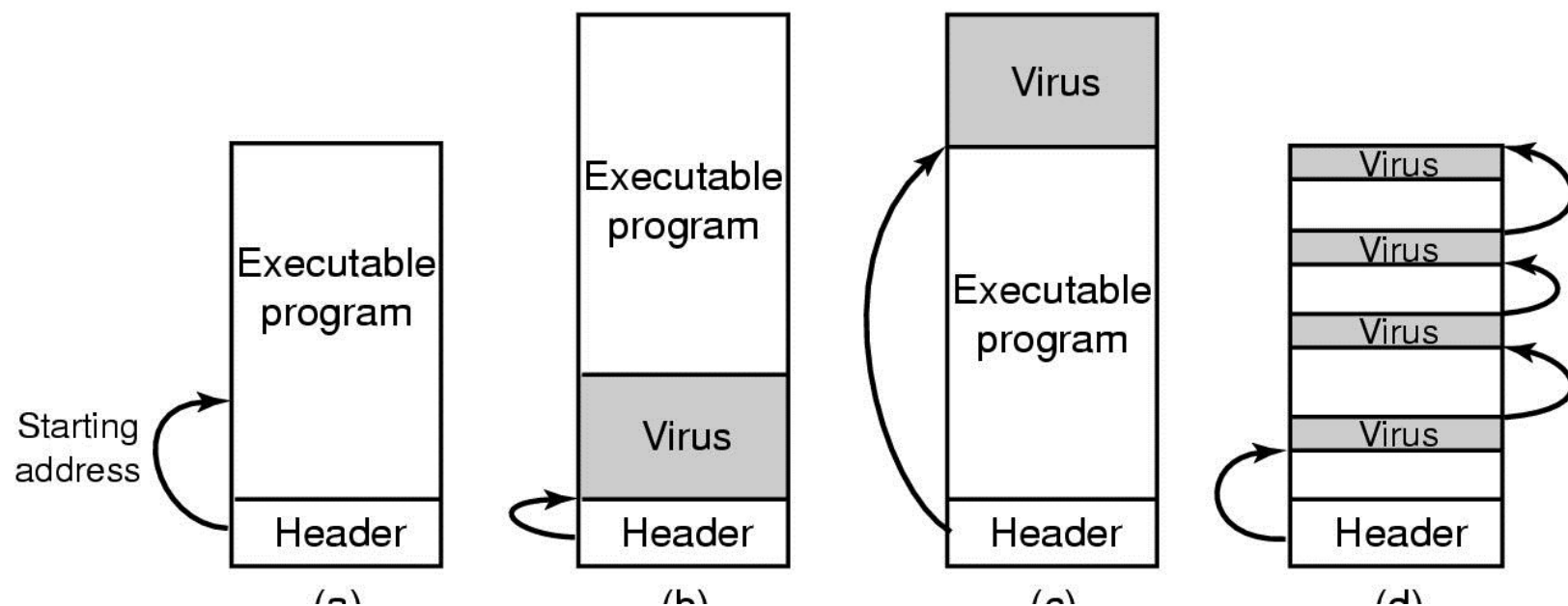
Executable Program Viruses (2)

...

```
while (TRUE) {
    dp = readdir(dirp);
    if (dp == NULL) {
        chdir ("..");
        break;
    }
    if (dp->d_name[0] == '.') continue;
    lstat(dp->d_name, &sbuf);
    if (S_ISLNK(sbuf.st_mode)) continue;
    if (chdir(dp->d_name) == 0) {
        search(".");
    } else {
        if (access(dp->d_name,X_OK) == 0) /* if executable, infect it */
            infect(dp->d_name);
    }
    closedir(dirp);
}
```

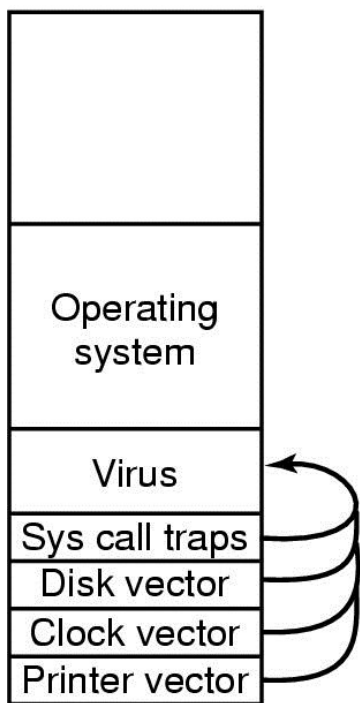
在Unix系统中查找可执行文件的递归过程

寄生病毒：Parasitic Viruses

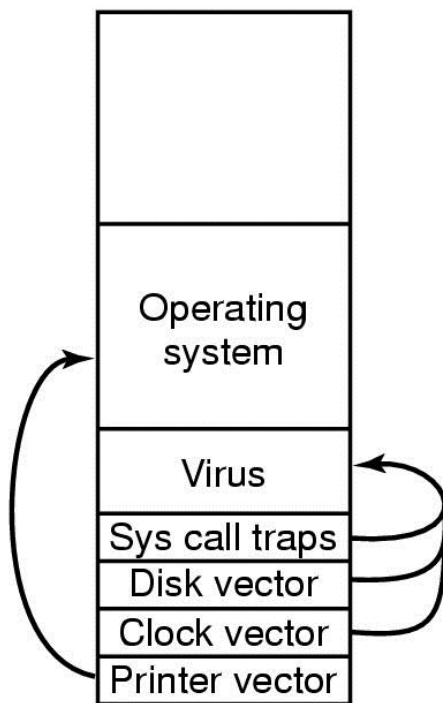


(a) 可执行程序. (b) 病毒位于前端. (c) 病毒位于末端. (d) 病毒分布于空闲空间

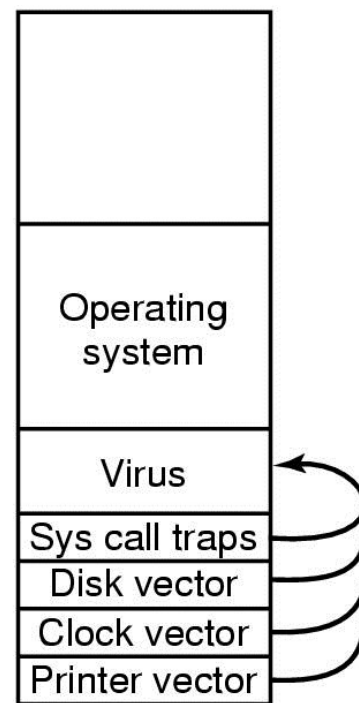
引导扇区病毒: Boot Sector Viruses



(a)



(b)



(c)

(a) 病毒捕获了中断向量表. (b) 操作系统重新获得中断向量表的指针. (c) 病毒发现后, 重新获得中断向量表.

2017 WannaCry勒索病毒

- 受害者只有按要求支付等额价值300美元的比特币才能解密——为什么比特币？
- 利用永恒之蓝（Eternal Blue）“网络武器”，“永恒之蓝”针对CVE-2017-（0143~0148）这几个漏洞开发的漏洞利用工具，通过利用Windows SMB协议的漏洞来远程执行代码，并提升自身至系统权限。



间谍软件： Spyware

- 间谍软件：秘密加载到计算机，在后台进行超出用户意愿的处理。
- 主要特征
 - 自身隐藏性
 - 收集用户数据
 - 将收集的信息传给远程的监控程序
 - 用户难以卸载
- 传播方式：下载软件、浏览器工具条、ActiveX控件

Spyware的破坏行为

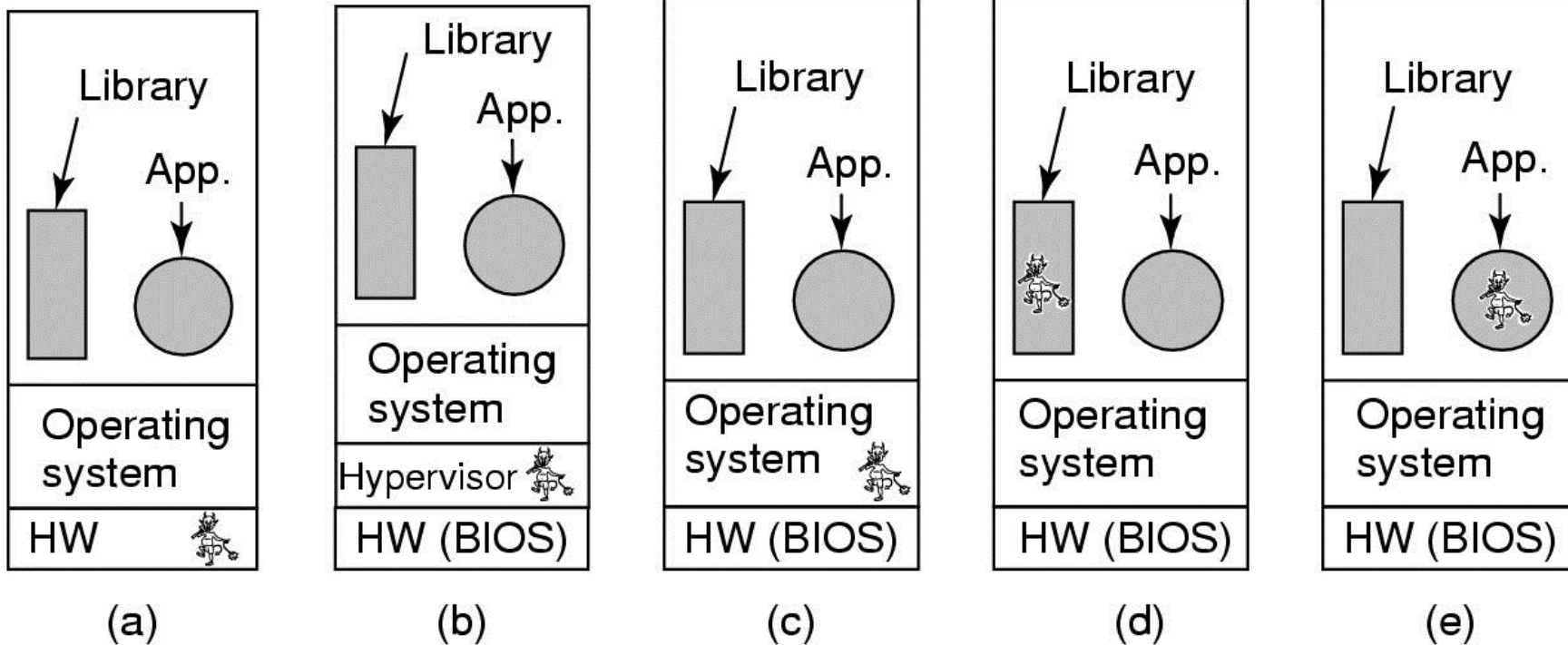
- 修改浏览器主页.
- 修改浏览器的收藏夹.
- 在浏览器中添加新的工具栏.
- 修改缺省的媒体播放器.
- 修改用户缺省的搜索引擎.
- 在Windows桌面上添加新图标.
- 替换网页顶端的广告.
- 在标准的Windows对话框中放置广告
- 连续弹出广告窗口.

Rootkit的类型 (1)

- 固件: Firmware rootkits
- 管理程序: Hypervisor rootkits
- 内核: Kernel rootkits
- 库文件: Library rootkits
- 应用程序: Application rootkits

主要功能: 隐藏其他程序进程的软件
How?

Rootkit的类型 (2)

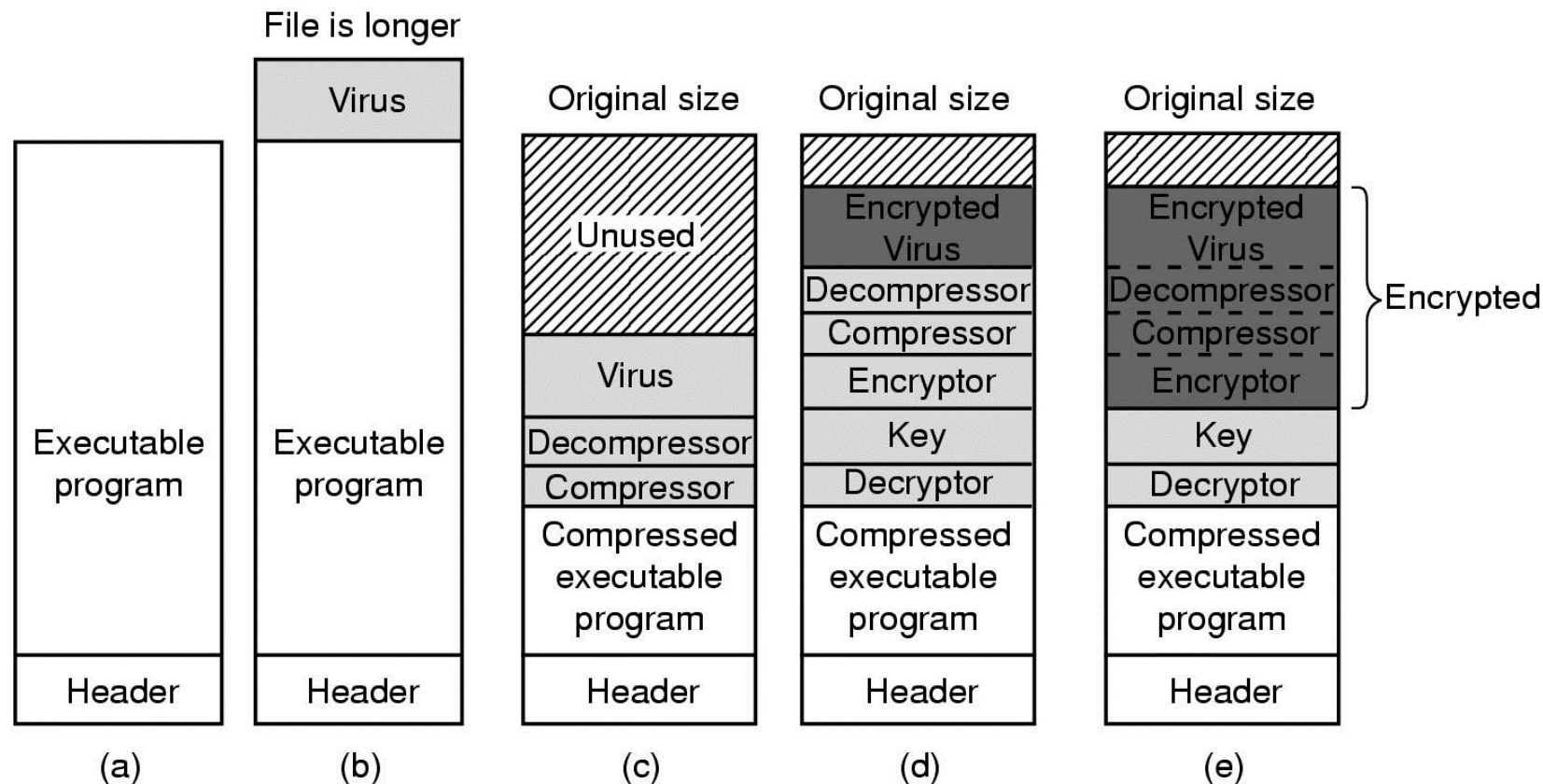


Rootkit 的五个藏身之处.

反病毒技术

- 病毒扫描: Virus scanners
- 完整性检查: Integrity checkers
- 行为检查: Behavioral checkers
- 病毒避免: Virus avoidance

病毒扫描: (1)



(a) 可执行程序. (b) 已经被感染的程序.

(c) 经过压缩的被感染程序. (d) 加密的病毒. (e) 压缩的病毒, 并对压缩码进行了加密.

病毒扫描(2)

```
MOV A,R1
ADD B,R1
ADD C,R1
SUB #4,R1
MOV R1,X
```

(a)

```
MOV A,R1
NOP
ADD B,R1
NOP
ADD C,R1
NOP
SUB #4,R1
NOP
MOV R1,X
```

(b)

```
MOV A,R1
ADD #0,R1
ADD B,R1
OR R1,R1
ADD C,R1
SHL #0,R1
SUB #4,R1
JMP .+1
MOV R1,X
```

(c)

```
MOV A,R1
OR R1,R1
ADD B,R1
MOV R1,R5
ADD C,R1
SHL R1,0
SUB #4,R1
ADD R5,R5
MOV R1,X
MOV R5,Y
```

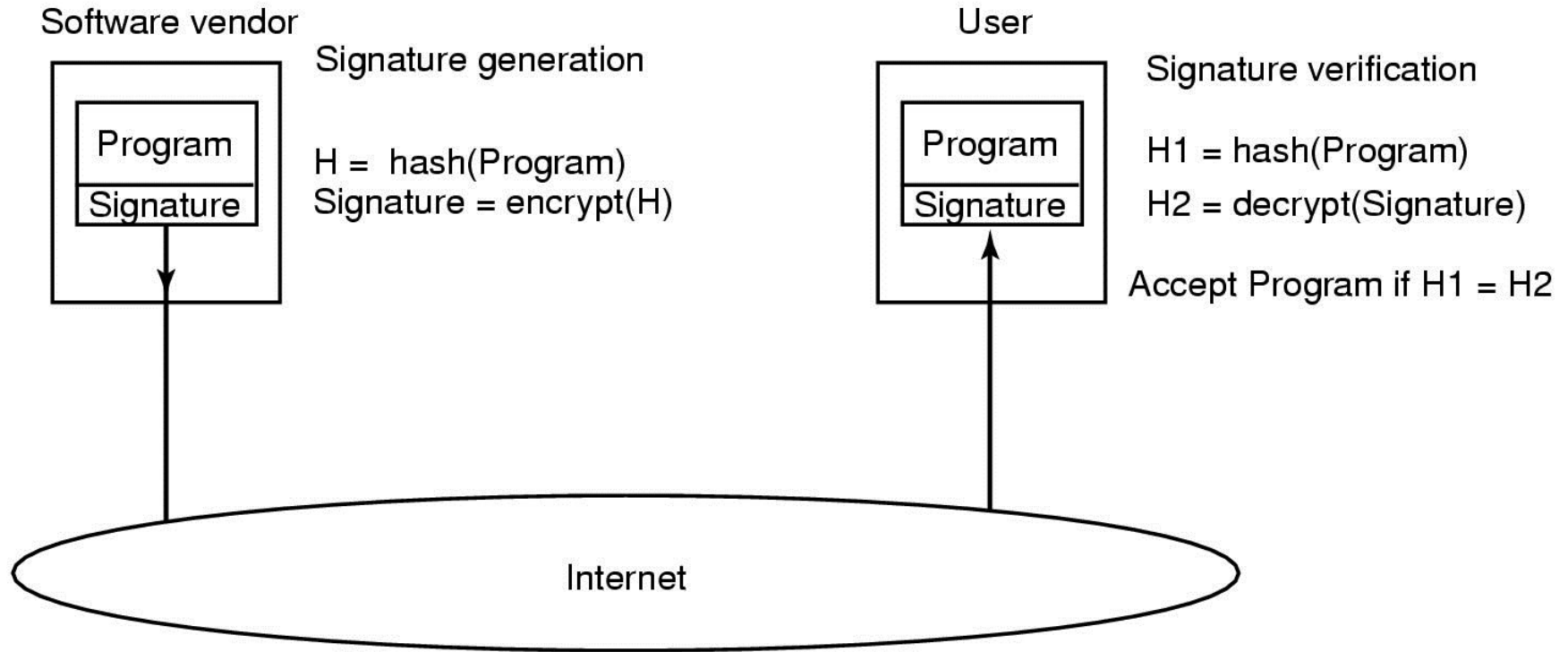
(d)

```
MOV A,R1
TST R1
ADD C,R1
MOV R1,R5
ADD B,R1
CMP R2,R5
SUB #4,R1
JMP .+1
MOV R1,X
MOV R5,Y
```

(e)

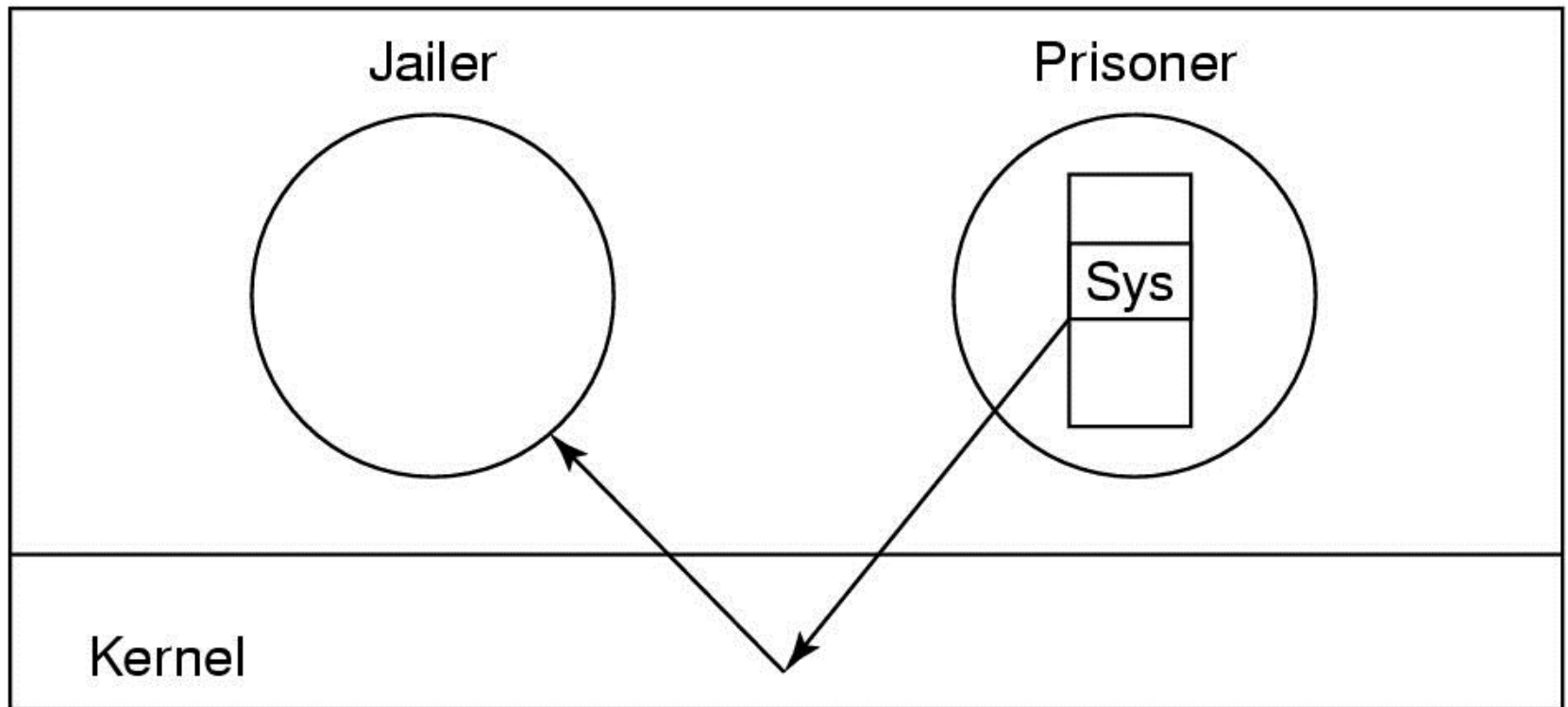
病毒的多态性.

代码签名: Code Signing



代码签名的处理流程.

囚禁: Jailing

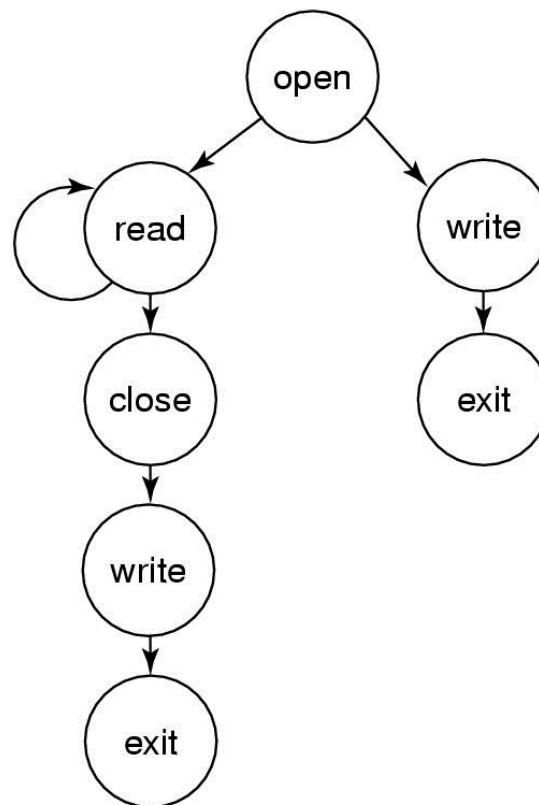


基于模型的入侵检测

```
int main(int argc *char argv[])
{
    int fd, n = 0;
    char buf[1];

    fd = open("data", 0);
    if (fd < 0) {
        printf("Bad data file\n");
        exit(1);
    } else {
        while (1) {
            read(fd, buf, 1);
            if (buf[0] == 0) {
                close(fd);
                printf("n = %d\n", n);
                exit(0);
            }
            n = n + 1;
        }
    }
}
```

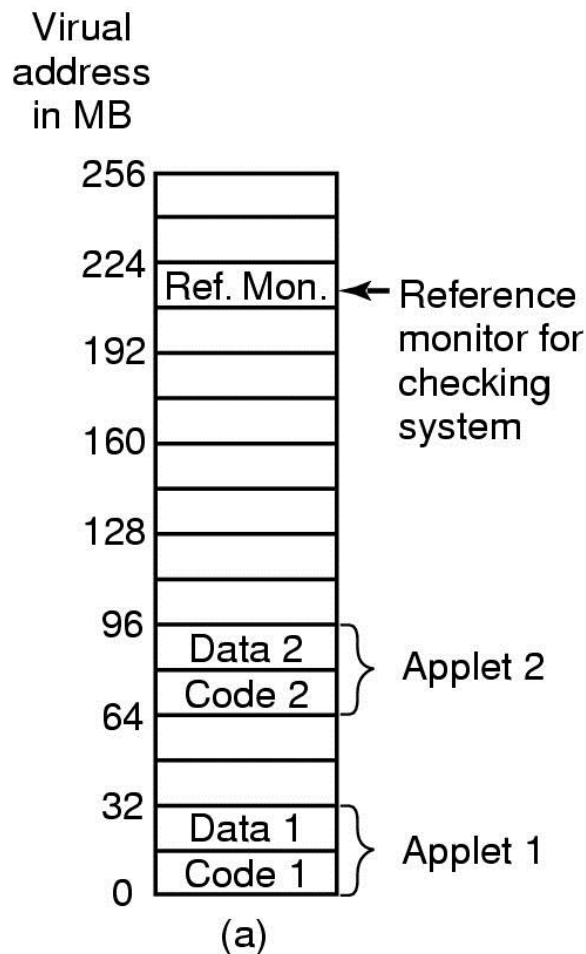
(a)



(b)

(a) 程序. (b) 程序的系统调用图.

沙箱机制: Sandboxing

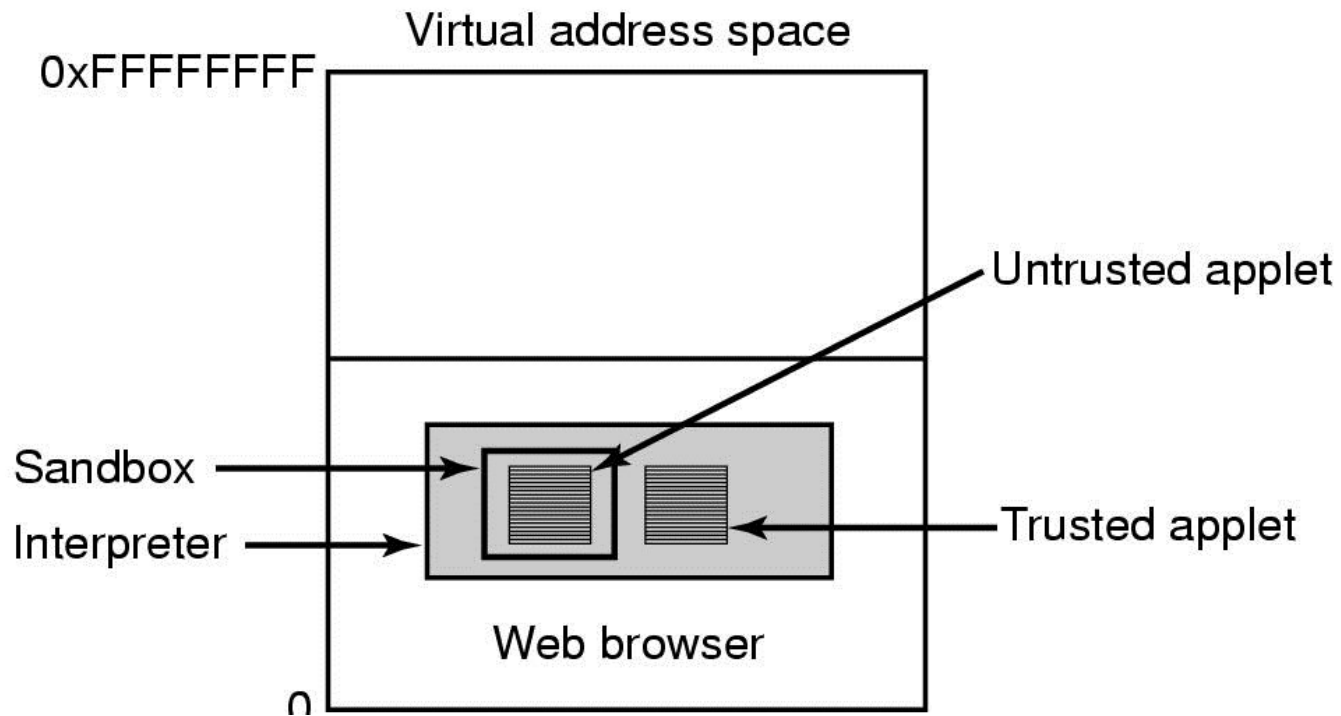


```
MOV R1, S1
SHR #24, S1
CMP S1, S2
TRAPNE
JMP (R1)
```

(b)

- (a) 内存划分为16-MB 大小的沙箱
- (b) 一种检查指令有效性的方法.

解释执行: Interpretation

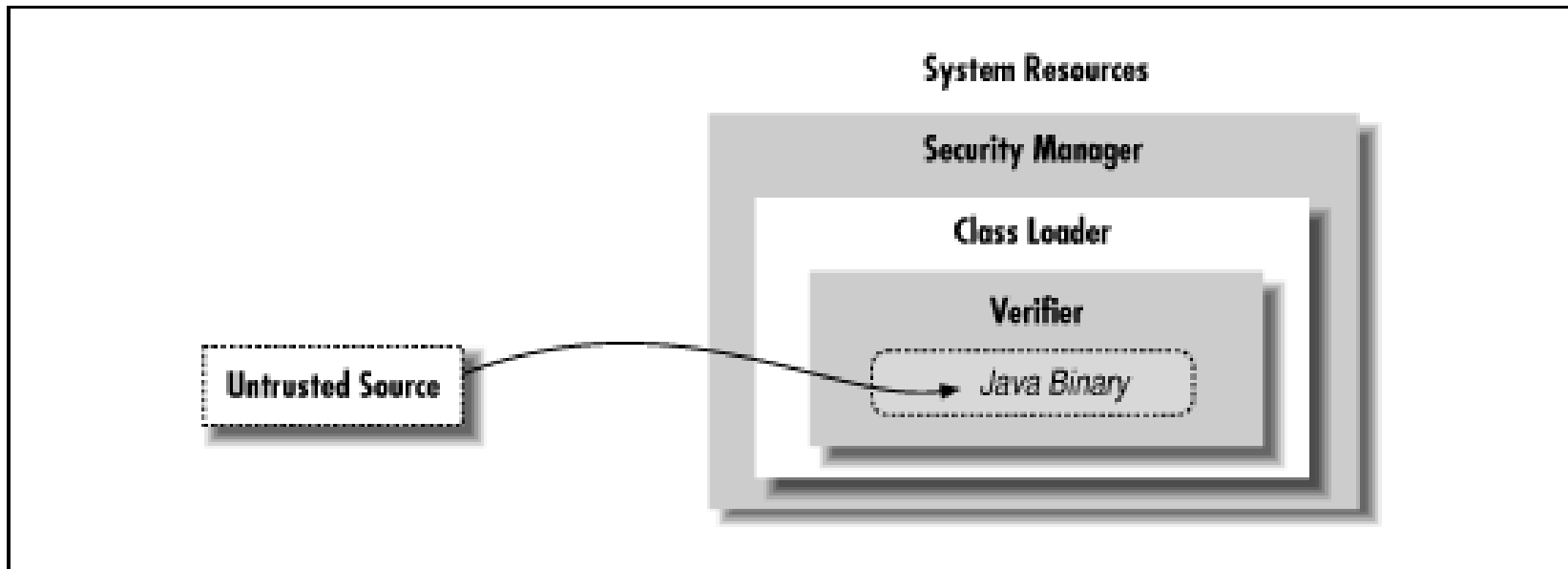


Web浏览器解释Applet执行

Java安全：(1)

JVM 字节码验证器检查Applet是否满足特定规则:

- 是否试图构造指针？（恶意编译器）
- 是否违反访问权限？
- 是否在把某类型的变量用于其他类型的变量？
- 是否会产生堆栈的上溢或者下溢？
- 是否非法进行类型的转换？



Java安全： (2)

URL	Signer	Object	Action
www.taxprep.com	TaxPrep	/usr/susan/1040.xls	Read
*		/usr/tmp/*	Read, Write
www.microsoft.com	Microsoft	/usr/susan/Office/—	Read, Write, Delete

JDK 1.2的安全配置示例

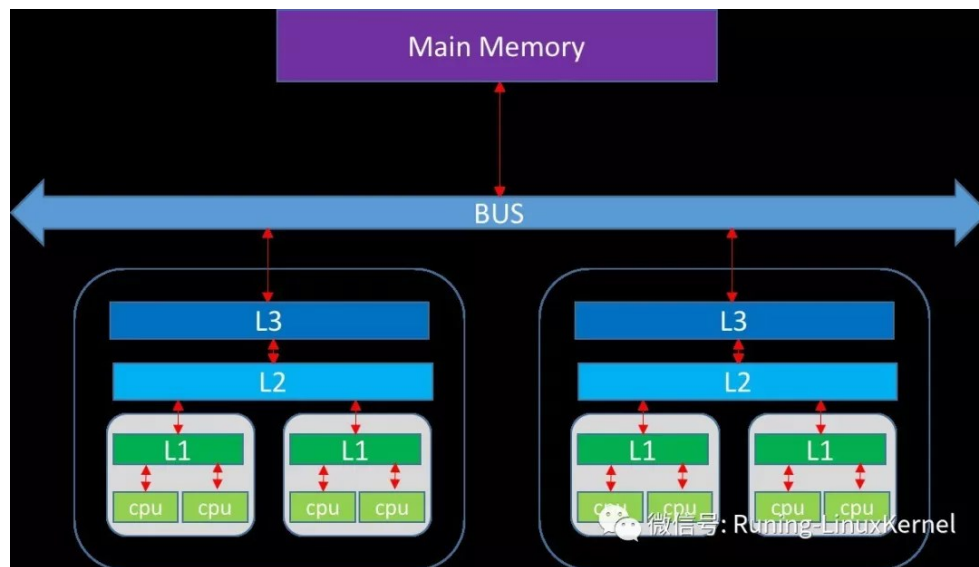
Meltdown & Spectre

- 2018年初爆发
 - 利用CPU乱序执行 + 侧信道攻击
 - 难以修复，修复代价大
- 漏洞背景
- 原理
- 利用方法
- POC
- 防范与修复



Meltdown & Spectre

- CPU内存体系
 - 访存性能差异明显



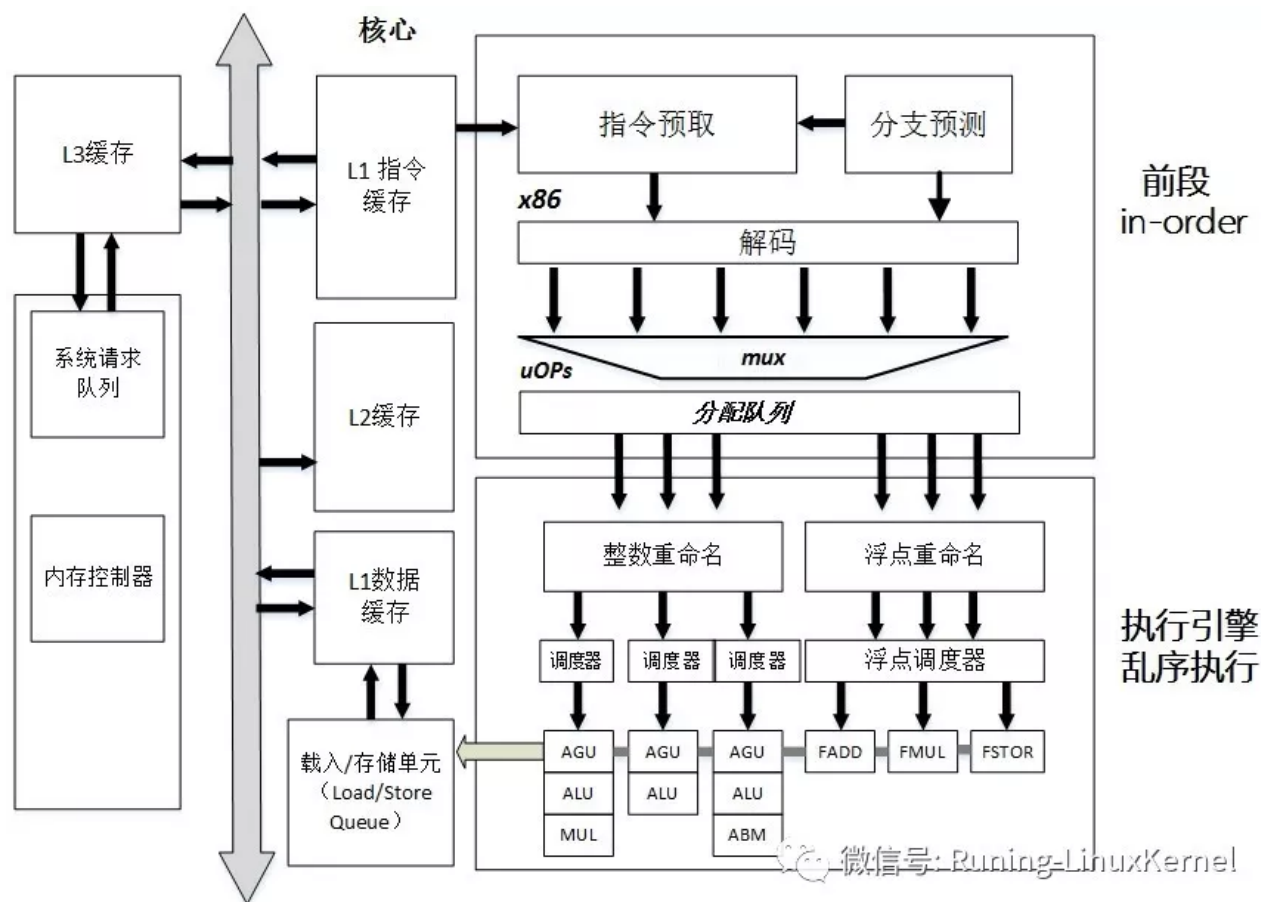
各级存储结构的访问延迟

访问类型	延迟
L1 cache命中	约4个时钟周期
L2 cache 命中	约10个时钟周期
L3 cache命中	约40个时钟周期
访问本地DDR	约60 纳秒
访问远端内存节点DDR	约100纳秒

Meltdown & Spectre

- CPU乱序执行 (Out-of-Order)

- 寄存器重命名
- 正确性
- 异常处理
- Cache污染



Meltdown & Spectre

- 漏洞原理

1: `clflush for user_probe[];` // 把user_probe_addr对应的cache全部都flush掉

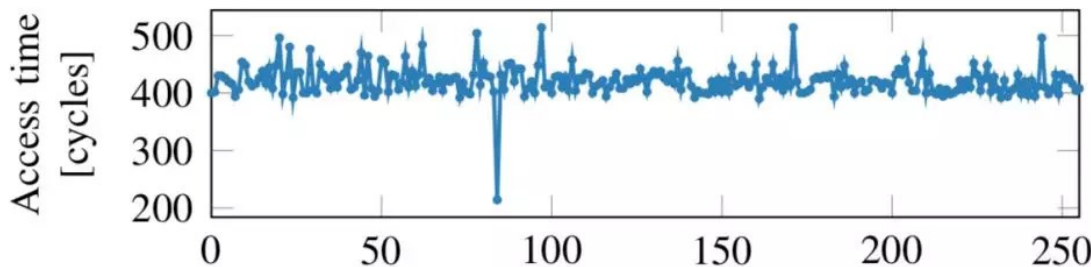
2: `u8 index = *(u8 *) attacked_mem_addr;` // attacked_mem_addr存放被攻击的地址

3: `data = user_probe_addr[index * 4096];` // user_probe_addr存放攻击者可以放访问的基地址

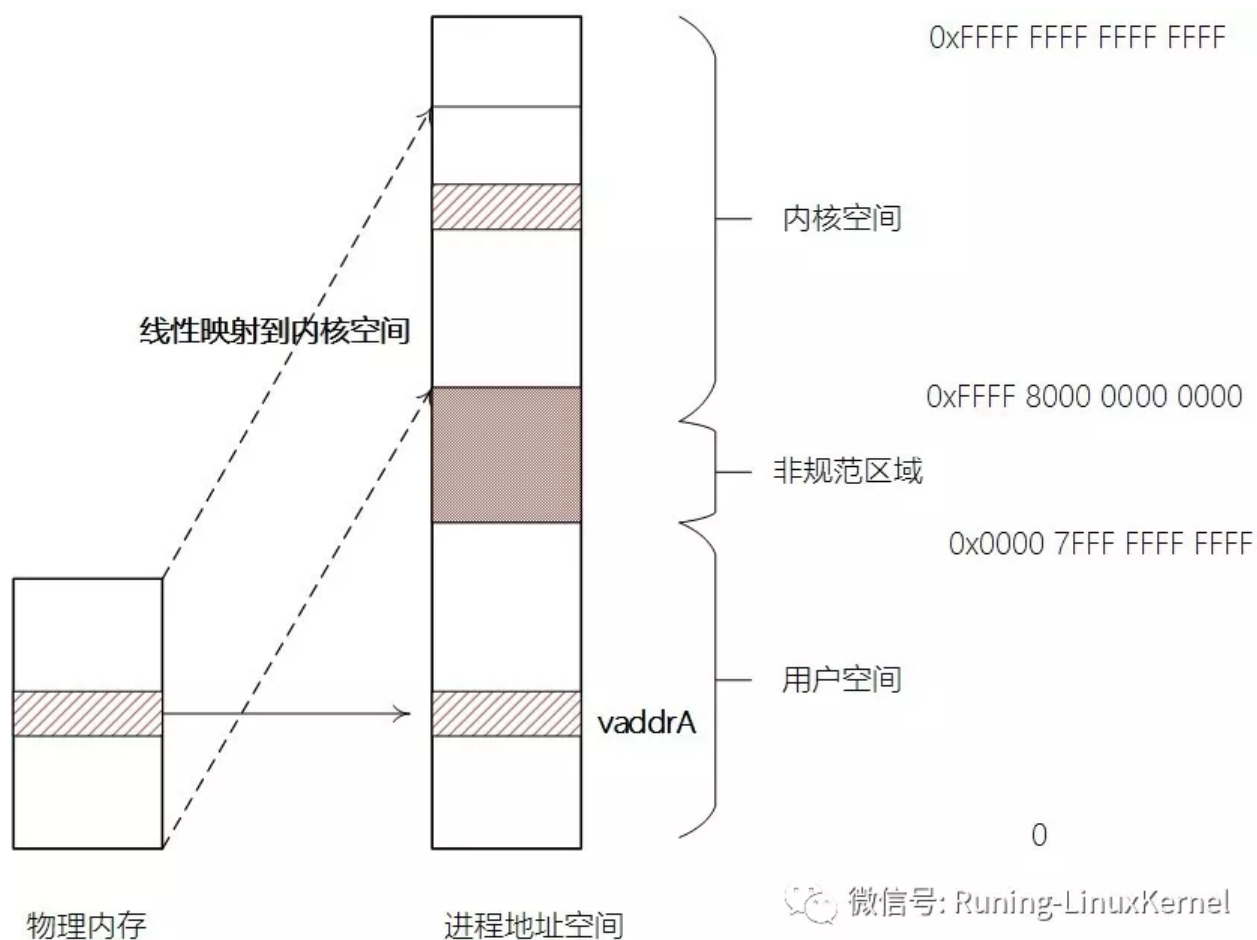
- 测量访问时间，进行侧信道攻击

- 问题

– 为什么要*4096?



如何访问敏感数据？



修复与防范

- Linux Kernel采用的是Kernel page-table isolation (KPTI)
 - The current state of kernel page-table isolation
<https://lwn.net/Articles/741878/>
 - Kernel page-table isolation
https://en.wikipedia.org/wiki/Kernel_page-table_isolation
-

不作恶 (Don't be evil)

- 能力越大责任越大 (With great power comes great responsibility)
 - 根据《中华人民共和国刑法》第二百八十六条规定，破坏计算机信息系统罪是指违反国家规定，对计算机信息系统功能或计算机信息系统中存储、处理或者传输的数据和应用程序进行破坏，或者故意制作、传播计算机病毒等破坏性程序，影响计算机系统正常运行，后果严重的行为。
 - 犯本罪的，处五年以下有期徒刑或者拘役；后果特别严重的，处五年以上有期徒刑。
-

问题？