OPERATING SYSTEMS

Security

Operating Systems

郑贵锋 博士
中山大学计算机学院
zhenggf@mail.sysu.edu.cn
https://gitee.com/code_sysu

中山大學 SUN YAT-SEN UNIVERSITY

Overview

- **目录**
  - 安全问题
  - 程序威胁
  - 系统和网络威胁
  - 密码术作为一种安全工具
  - 用户认证
  - 实现安全防御
  - 保护系统和网络的防火墙
  - 计算机安全分类
  - 例如：Windows7

Overview

- **安全问题**
  - 如果在所有情况下都按预期使用和访问资源，则系统安全
    - 无法实现
  - 入侵者（破解者/cracker）试图破坏安全
  - 威胁是潜在的安全违规行为
  - 攻击是企图破坏安全的行为
  - 攻击可以是意外的，也可以是恶意的
  - 比恶意误用更容易防止意外误用

Overview

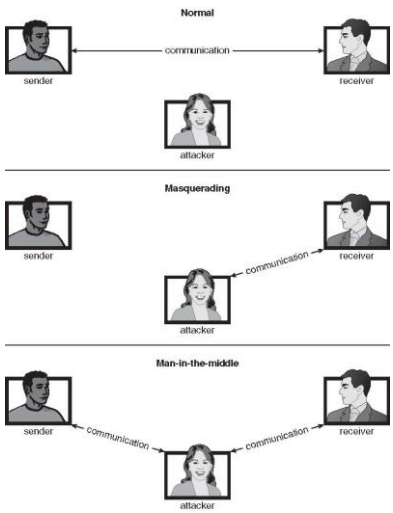
- **安全违规类别**
  - 违反保密
    - 未经授权读取数据
  - 违反诚信
    - 未经授权修改数据
  - 破坏可用性
    - 未经授权销毁数据
  - 盗窃服务
    - 未经授权使用资源
  - 拒绝服务(DoS)
    - 防止合法使用

Overview

- **安全违规方法**
  - 伪装（违反身份验证/认证）
    - 假装是授权用户以升级权限
  - 重播攻击
    - 按原样或通过消息篡改
  - 中间人攻击(man-in-middle attack)
    - 入侵者截断数据流中，伪装成发送者到接收者，反之亦然
  - 会话劫持(session hijacking)
    - 拦截已建立的会话以绕过认证

Overview

- **标准安全攻击**



Overview

- **安全措施级别**
  - 不可能有绝对的安全，但使作恶者的成本高到足以阻止大部分的入侵者
  - 安全必须在四个级别进行，才能有效：
    - 物理
      - 数据中心、服务器、连接的终端
    - 人员
      - 避免社交工程、网络钓鱼、垃圾箱潜水
    - 操作系统
      - 保护机制、调试
    - 网络
      - 截获的通信、中断、拒绝服务
  - 安全就像链条中最薄弱的一环一样脆弱
  - 但是太多的安全性会成为一个问题吗？

Program Threat

- **程序威胁**
  - 许多变体，许多名称
  - 特洛伊木马(Trojan Horse)
    - 误用其环境的代码段
    - 利用允许其他用户执行用户编写的程序的机制
    - 间谍软件、弹出式浏览器窗口、隐蔽通道
    - 高达80%的垃圾邮件由受间谍软件感染的系统发送
  - 后门(Trap Door)
    - 绕过正常安全程序的特定用户标识符或密码
    - 可以包含在编译器中
    - 如何检测它们？
      - 必须分析所有系统组件的所有源代码！

Program Threat

- **程序威胁**
  - 逻辑炸弹(logic bomb)
    - 在某些情况下引发安全事件的程序
    - 正常操作时没有安全漏洞，因此很难检测
  - 堆栈和缓冲区溢出
    - 利用程序中的错误（堆栈或内存缓冲区溢出）
    - 未能检查输入、参数的界限
    - 将堆栈上传递的参数写入堆栈上的返回地址
    - 当例程从调用返回时，返回到被攻击的地址
      - 指向加载到堆栈上的代码，以执行恶意程序
    - 未经授权的用户或权限升级

Program Threat

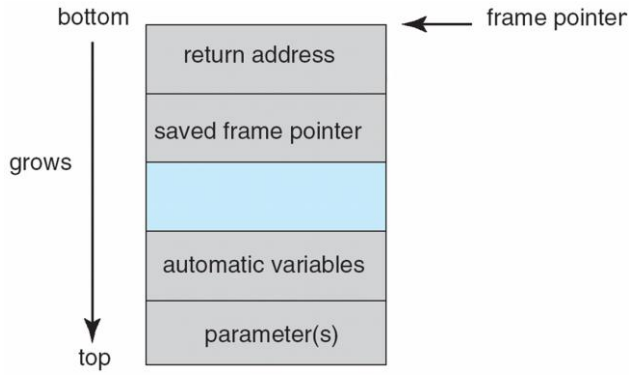
- **具有缓冲区溢出条件的C程序**

```
#include <stdio.h>
#define BUFFER SIZE 256

int main(int argc, char *argv[])
{
    char buffer[BUFFER SIZE];
    if (argc < 2)
        return -1;
    else {
        strcpy(buffer, argv[1]);
        return 0;
    }
}
/* 正确代码
 strncpy(buffer, argv[1], sizeof(buffer)-1);
*/
```

Program Threat

- **典型堆栈页帧的布局**



Program Threat

- **修改shell的代码**

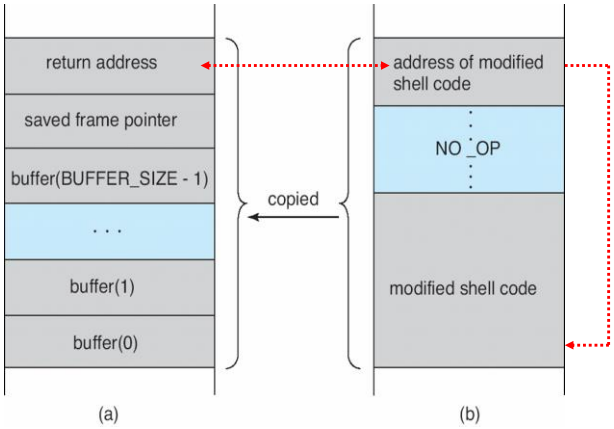
```
#include <stdio.h>

int main(int argc, char *argv[])
{
    execvp("\bin\sh","\bin \sh", NULL);
    return 0;
}

/* 该代码创建了一个shell进程，获得系统完全访问权 */
```

Program Threat

- 假设堆栈页帧



攻击前　　攻击后

Program Threat

- 需要很强的编程能力吗？
  - 对于第一步确定bug，和第二步编写漏洞代码，是的
  - 脚本骇客(*Script kiddies*)可以运行预先编写的攻击代码来攻击给定的系统
  - 攻击代码可以获得具有进程所有者权限的shell，或打开网络端口、删除文件、下载程序等。
  - 根据bug的不同，可以使用允许的连接绕过防火墙在网络上执行攻击
  - 通过禁用堆栈执行或向页表添加位以指示“不可执行”状态，可以禁用缓冲区溢出
    - 在SPARC和x86中提供
    - 但仍然有安全漏洞

Program Threat

- **程序威胁**
  - 病毒Viruses
    - 嵌入合法程序中的代码片段
    - 自我复制，设计用于感染其他计算机
    - 非常特定于CPU体系结构、操作系统和应用程序
    - 通常通过电子邮件或宏进行传输
    - 用于重新格式化硬盘驱动器的Visual Basic宏

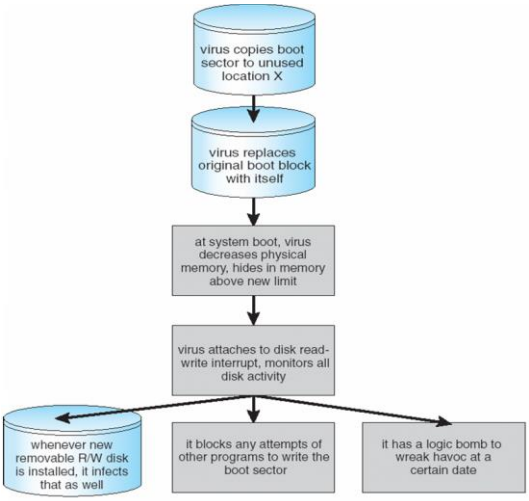
```
Sub AutoOpen()
Dim oFS
    Set oFS
    CreateObject("Scripting.FileSystemObject")
    vs = Shell("c:command.com /k format c:",vbHide)
End Sub
```

Program Threat

- **程序威胁**
  - 病毒
    - 病毒滴管将病毒插入系统
    - 许多种类的病毒，实际上是成千上万的病毒
      - 文件/寄生
      - 引导/内存
      - 宏
      - 源代码
      - 多态性以避免具有病毒特征
      - 加密的
      - 隐身
      - 隧道
      - 复合（感染多个部分）
      - 装甲（逃避检测与杀毒）

Program Threat

- **程序威胁**
  - 病毒
    - 引导扇区计算机病毒



Program Threat

- **威胁还在继续**
  - 攻击仍然很常见，仍然在发生
  - 随着时间的推移，袭击从科学实验转移到有组织犯罪的工具
    - 针对特定公司
    - 创建僵尸网络(botnet)用作垃圾邮件和DDoS递送的工具
    - 击键记录器获取密码、信用卡号码
  - 为什么Windows是大多数攻击的目标？
    - 最常见
    - 每个人都是管理员
    - 被认为有害的单一化(*Monoculture*)，同样的硬件、OS、应用软件，增加了由病毒和其他入侵带来的威胁和破坏

系统和网络威胁

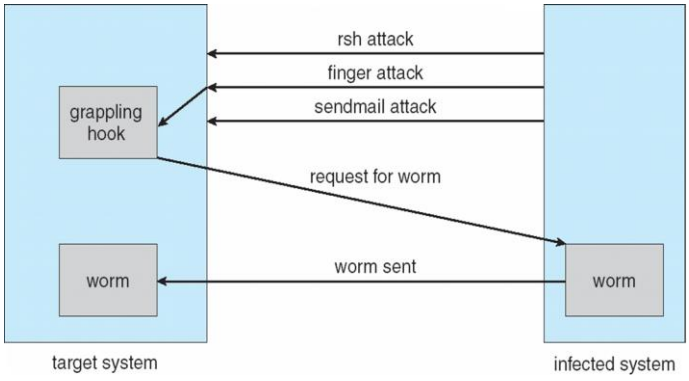
- 默认情况下，某些系统是“开放”的，而不是安全的
  - 减少攻击面
  - 但更难使用，需要更多的知识才能管理
- 网络威胁更难检测、预防
  - 保护系统较弱
  - 更难基于一个共享密码来访问
  - 系统连接到internet后没有物理限制
    - 或者在系统连接到internet的网络上
  - 即使确定连接系统的位置也很困难
    - IP地址是唯一的知识

蠕虫worm

- 利用繁殖(spawn)机制复制自身；独立程序
- 莫里斯(Morris)网络蠕虫
  - 利用了finger和sendmail程序中的UNIX网络功能（远程访问）和错误
  - 利用rsh使用的信任关系机制，无需使用密码即可访问友好系统
  - 抓钩程序上传主蠕虫程序
    - 99行C代码
    - 钩住系统然后上传主代码，试图攻击连接的系统
    - 还试图通过猜密码进入本地系统上的其他用户帐户
    - 如果目标系统已感染，则中止，但每第7次除外

蠕虫

- 莫里斯网络蠕虫



端口扫描

- 端口扫描是检测漏洞的方法
- 自动尝试连接到一个或多个IP地址上的一系列端口
- 检测应答服务协议
- 检测系统上运行的操作系统和版本
- nmap扫描给定IP范围内的所有端口以获得响应
- nessus有一个针对系统应用的协议和bug（以及漏洞）数据库
- 经常从僵尸系统中启动
  - 降低跟踪能力，因为端口扫描是可被检测的

System and Network Threats

- **拒绝服务**
  - 使目标计算机过载，使其无法执行任何有用的工作
  - 分布式拒绝服务(DDoS)同时来自多个站点（通过僵尸）
  - 考虑开始IP连接的握手(SYN)
    - 操作系统可以处理多少已启动的连接？
  - 考虑到网站的流量
    - 你如何区分成为目标和真正受欢迎之间的区别？
  - 意外 – CS学生编写错误的fork()代码
  - 有目的的 — 勒索、惩罚

Overview

- **Sobig.F蠕虫**
  - 更现代的例子
  - 伪装成照片，通过用被盗信用卡创建的帐户，上传到成人新闻组
  - 目标是Windows系统
  - 拥有自己的SMTP引擎，将自己作为附件发送给感染系统的地址簿中的每个人
  - 伪装成无伤大雅的主题，看起来像是来自某个已知的人
  - 附件是可执行程序，在默认Windows系统目录中创建WINPPR23.EXE
  - 同时修改Windows注册表

```
[HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
  "TrayX" = %windir%\winppr32.exe /sinc
[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
  "TrayX" = %windir%\winppr32.exe /sinc
```

- **密码术作为一种安全工具**
  - 可用的最广泛的安全工具
    - 在给定的计算机内部，可以知道消息的源和目的地并加以保护
      - 操作系统创建、管理、保护进程ID、通信端口等。
    - 如果没有加密，则无法信任网络上消息的源和目标
      - 本地网络 – 根据IP地址？
        - 考虑添加未授权主机
      - 广域网/互联网–如何建立认证
        - 不能根据IP地址（容易被伪造）

- **密码术**
  - 限制消息的潜在发送者（源）和/或接收者（目的地）的方法
    - 基于密码(密钥)
    - 使能够
      - 来源确认
      - 仅在特定目的地接收
      - 发送方和接收方之间的信任关系

加密

- 约束消息的可能接收者集
- 加密(Encryption)算法包括
  - 一个密钥集合 K
  - 一个消息集合 M
  - 一个密文集合 C
  - 一个加密函数 E : K → (M → C). 即，每个 E_k, $k \in K$, 是从信息生成密文的函数
    - E 和任何 k 的 E_k 都应是可有效计算的函数
  - 一个解密函数 D : K → (C → M). 即，每个D_k, $k \in K$, 是从密文生成信息的函数
    - D 和任何 k 的 D_k 都应是可有效计算的函数

加密

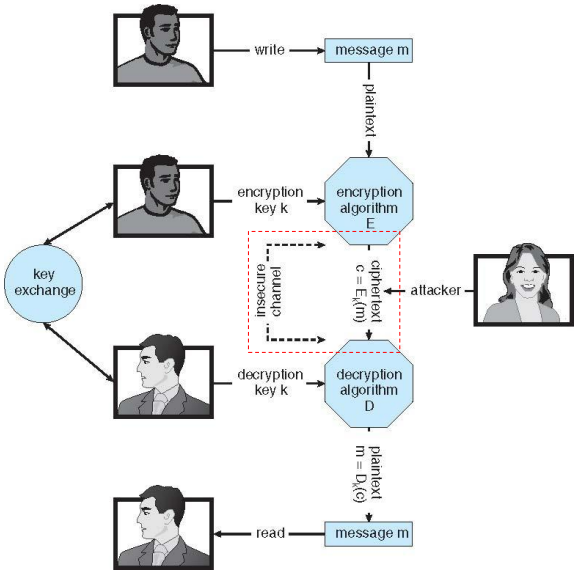
- 加密算法必须提供以下基本属性：给定一个密文 $c \in C$, 计算机只有r持有 k 才能算出 m, 使得 $E_k(m) = C$.
  - 因此，持有 k 的计算机可以将密文解密为用于产生密文的明文，但不持有 k 的计算机无法解密密文
  - 由于密文通常是公开的（例如，在网络上发送），因此重要的是不能从密文推测出 k.

Cryptography

- **对称加密**
  - 用于加密和解密的相同密钥
    - 因此k必须保密
  - 数据加密标准(DES)是最常用的对称块加密算法（由美国政府创建）
    - 一次加密一块数据
    - 密钥太短，因此现在被认为不安全
  - 三重DES被认为更安全
    - 算法使用2或3个密钥，重复加密3次
  - 2001年NIST采用了新的块密码 — 高级加密标准(AES)
    - 128、192或256位的密钥，处理128位的块
  - RC4是最常见的对称流加密，但已知存在漏洞
    - 加密/解密字节流（即无线传输）
    - 密钥是伪随机位生成器的输入
      - 生成一个无限的密钥流

Cryptography

- **不安全媒介上的安全通信**



非对称加密

- 基于每个用户有两个密钥的公钥加密：
  - 公钥 – 用于加密数据的已发布密钥
  - 私钥 – 仅用于解密数据的单个用户知道的密钥
- 必须是一个可以公开的加密方案，而不容易找出解密方案
  - 最常见的是RSA块加密
  - 测试一个数是否为素数的有效算法
  - 目前还没有一种有效的算法来求一个数的素因子

非对称加密

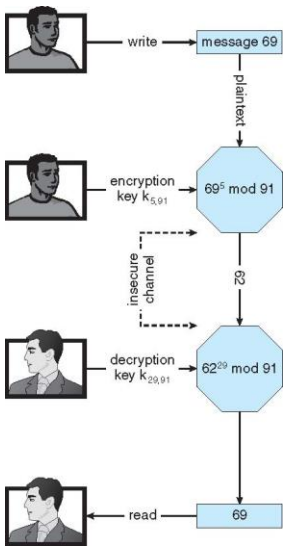
- 形式上，从 k_e,N 中推测出 k_d,N 在计算上是不可行的，因此 k_e 不需要保密，可以广泛传播
  - k_e 是公钥
  - k_d 是私钥
  - N 是两个随机选择的大素数 p 和 q 的乘积（例如，p 和 q 都有 512 bits）
  - 加密算法是 $E_{k_e,N}(m) = m^{k_e} \bmod N$, 其中 k_e 满足

    $k_e k_d \bmod (p-1)(q-1) = 1$
  - 然后，解密算法为 $D_{k_d,N}(c) = c^{k_d} \bmod N$

非对称加密

- 实例
  - 设 p = 7, q = 13
  - 然后我们计算 N = 7∗13 = 91 和 (p−1)(q−1) = 72
  - 接下来，我们选择与72互为素数且小于72的 k_e, 得到 5
  - 最后，我们计算k_d, 使得 $k_e k_d$ mod 72 = 1, 得到29
  - 我们现在有了一对密钥
    - 公钥，k_e,N = 5, 91
    - 私钥，k_d,N = 29, 91
  - 使用公钥对消息 69 进行加密将得到密文 62
  - 密文可以用私钥解密
    - 公钥可以明文形式分发给任何要与公钥持有者通信的人
  - 为什么安全？（实际使用的N是很大的数）
    - 即使已知N = 91，但没有人能够在一定的时间内确定这两个随机生成的素数p = 7, q = 13, 使得 pq = N.

使用RSA非对称加密的加密



Cryptography 35 / 54

密码术

- 注意：基于变换的对称密码，基于数学函数的非对称密码
  - 非对称的计算量非常昂贵
  - 通常不用于大量数据加密

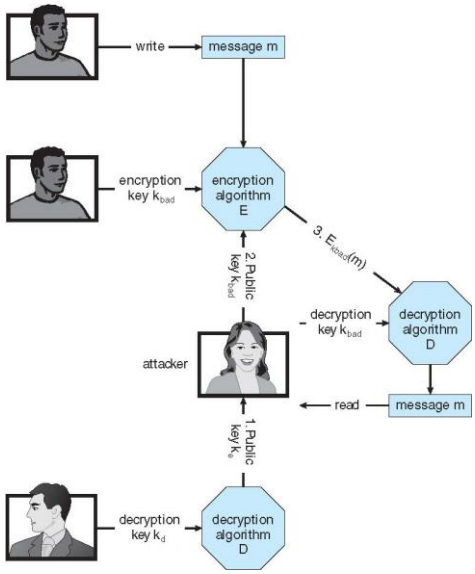
Cryptography 36 / 54

密钥分配

- 对称密钥的传递是一个巨大的挑战
  - 有时是带外(out-of-band)完成的
- 非对称公钥可以扩散 – 存储在密钥圈(key ring)上
  - 即使是非对称公钥分配也需要小心 — 中间人攻击

Cryptography

- 非对称密码系统中的中间人攻击



Cryptography

- 数字证书
  - 数字证书证明谁拥有一个公钥
  - 公钥由受信任方进行了数字签名
  - 受信任方从实体接收身份证明，并证明公钥属于该实体
  - 证书颁发机构是受信任的一方（可信中心） – 其公钥包含在Web浏览器发行版中
    - 他们通过数字签名等方式为其他机构提供担保

Cryptography

- **密码术的实现**
  - 可以在ISO参考模型的各个层上完成
    - 传输层的SSL
    - 网络层通常是IPSec
      - 密钥交换的IKE
      - 虚拟专用网络(VPN)的基础
  - 为什么不只是在最底层实现？
    - 有时需要高层的更多信息
      - 例如，用户认证
      - 例如，电子邮件传递

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication
	Segments	4. Transport	End-to-end connections and reliability, flow control
Media layers	Packet/Datagram	3. Network	Path determination and logical addressing
	Frame	2. Data Link	Physical addressing
	Bit	1. Physical	Media, signal and binary transmission

OSI model

7. Application Layer
NNTP · SIP · SSI · DNS · FTP · Gopher · HTTP · NFS · NTP · SMPP · SMTP · SNMP · Telnet · Netconf · (more)

6. Presentation Layer
MIME · XDR · TLS · SSL

5. Session Layer
Named Pipes · NetBIOS · SAP · L2TP · PPTP · SPDY

4. Transport Layer
TCP · UDP · SCTP · DCCP · SPX

3. Network Layer
IP (IPv4, IPv6) · ICMP · IPsec · IGMP · IPX · AppleTalk

2. Data Link Layer
ATM · SDLC · HDLC · ARP · CSLIP · SLIP · GFP · PLIP · IEEE 802.3 · Frame Relay · ITU-T G.hn DLL · PPP · X.25 · Network Switch · DHCP

1. Physical Layer
EIA/TIA-232 · EIA/TIA-449 · ITU-T V-Series · I.430 · I.431 · POTS · PDH · SONET/SDH · PON · OTN · DSL · IEEE 802.3 · IEEE 802.11 · IEEE 802.15 · IEEE 802.16 · IEEE 1394 · ITU-T G.hn PHY · USB · Bluetooth · Hubs

This box: view · talk · edit

Cryptography

- **加密示例–SSL**
  - 在ISO网络模型的一层（传输层）插入加密
  - SSL – 安全套接字层（也称为TLS）
  - 限制两台计算机只与对方交换消息的加密协议
    - 非常复杂，有很多变种
  - 用于Web服务器和浏览器之间的安全通信（信用卡号）
  - 服务器通过证书(*certificate*)进行验证，确保客户端与正确的服务器通信
  - 非对称加密用于为会话期间的大部分通信建立安全会话密钥（对称加密）
  - 然后，每台计算机之间的通信使用对称密钥加密
  - 参阅课本中的更多细节

Authentication

- 认证
  - 限制消息的潜在发送者集合
    - 加密的补充（加密限制消息的可能接收者集合）
    - 也可以证明消息未被修改
  - 算法组件
    - 一个密钥集合 K
    - 一个消息集合 M
    - 一个认证者集合 A
    - 一个函数 S: K → (M → A)
      - 也就是说，对于每个 $k \in K$, S_k 是一个用于从消息生成认证者的函数
      - S 和任何 k 的 S_k 都应该是可有效计算的函数
    - 一个函数 V: K→ (M × A → {true, false}). 也就是说，对于每个 $k \in K$, V_k 是用于验证消息上的身份认证者的函数
      - V 和任何 k 的 V_k 都应该是可有效计算的函数

Authentication

- 认证
  - 对于消息 m, 计算机可以生成认证者 $a \in A$ 使得只有当它持有 k 时，$V_k(m, a) = true$
  - 因此，持有 k 的计算机可以在消息上生成认证者，以便持有 k 的任何其他计算机都可以验证它们
  - 未持有 k 的计算机无法对可以使用 V_k 验证的消息生成认证者
  - 由于认证者通常是公开的（例如，它们与消息本身一起在网络上发送），因此从认证者推测出 k 应是不可行的
  - 实际上，如果 $V_k(m, a) = true$, 那么我们知道 m 没有被修改，消息的发送者持有 k.
    - 如果我们只与一个实体共享 k, 则知道消息的来源

认证 – 哈希函数

- 认证的基础
- 从消息 m 创建固定大小的数据块，称为报文摘要或哈希值
- 哈希函数 H 在 m 上必须是抵抗碰撞的
  - 一定不可能找到一个 $m' \neq m$ 使得 $H(m) = H(m')$
- 如果 $H(m) = H(m')$, 那么 $m = m'$
  - 消息未被修改
- 常见的报文摘要函数包括 MD5（生成128位哈希）和SHA-1（输出160位哈希）
- 报文摘要可用于检测修改，但不能作为认证者
  - 例如，H(m)可以与消息一起发送
    - 但如果已知 H，则有人可以将 m 修改为 m' 并重新计算 $H(m')$，这样就不能检测出消息修改
    - 因此必须认证 H(m)

认证 – MAC

- 消息认证码(MAC)认证算法中使用对称加密
- 使用密钥从消息生成加密校验和
  - 可以安全认证短消息
- 首先将长消息 m 通过抵抗碰撞的哈希函数 H 获得哈希值 H(m)，再对短的 H(m) 认证，据此获得安全认证长消息的方法
- 注意，计算生成认证者函数 S_k 和验证认证者函数 V_k 都需要 k，因此任何人能够计算其中一个都可以计算另一个
  - 对称加密的问题

Authentication

- **认证 – 数字签名**
  - 基于非对称密钥和数字签名算法
  - 产生的认证者是数字签名
  - 非常有用 – 任何人都可以验证消息的真实性
  - 在数字签名算法中，从 k_v 推测出 k_s 在计算上是不可行的
    - V 是一个单向函数
    - 因此，k_v 是公钥，k_s 是私钥
  - 考虑RSA数字签名算法
    - 与RSA加密算法类似，但密钥用途是相反的
    - 消息的数字签名 $S_{k_s}(m) = H(m)^{k_s} \bmod N$ — 用私钥
    - 密钥 k_s 也是一对(d, N)，其中 N 是两个随机选择的大素数 p 和 q 的乘积
    - 验证算法为 $V_{k_v}(m, a)$

      $$a^{k_v} \bmod N = H(m)$$

      式中，k_v 满足 $k_v k_s \bmod (p-1)(q-1) = 1$

Authentication

- **认证**
  - 为什么认证是加密的一个子集？
    - 计算量更少（RSA数字签名除外）
    - 认证者通常比消息短
    - 有时需要认证，但不需要保密
      - 例如，对软件补丁进行签名，证明其来自公司且未被修改
    - 可作为不可否认(*non-repudiation*)的基础

Authentication

- **用户认证**
  - 正确识别用户至关重要，因为保护系统取决于用户ID
  - 用户身份通常使用密码建立，可以被视为密钥或能力的特例
  - 密码必须保密
    - 频繁更改密码
    - 避免使用历史上用过的密码
    - 使用“不可猜测”密码
    - 记录所有无效访问尝试（但不记录密码本身）
    - 未经授权的转移
  - 密码也可以加密或只允许使用一次
    - 加密密码能解决暴露问题吗？
      - 可能解决嗅探(sniffing)问题
      - 考虑肩窥(shoulder surfing)
      - 考虑特洛伊木马击键记录器
      - 密码是如何存储在认证站点上的？

Authentication

- **密码**
  - 加密以避免必须保密
    - 但无论如何都要保密（例如，Unix使用仅超级用户可读的文件/etc/shadow）
    - 使用易于计算但难以反演的算法
    - 只存储加密密码，从不保存解密的密码
    - 添加“盐(salt)”以避免将相同的密码加密为相同的值
  - 一次性密码
    - 使用基于种子的函数计算用户和计算机的密码
    - 使用硬件设备/计算器/密钥卡生成密码
      - 变化非常频繁
  - 生物特征
    - 某些物理属性（指纹、掌纹扫描）
  - 多因素认证
    - 认证需要两个或多个因素
      - USB“加密狗”、生物特征测量，和密码

Implementing Security Defenses

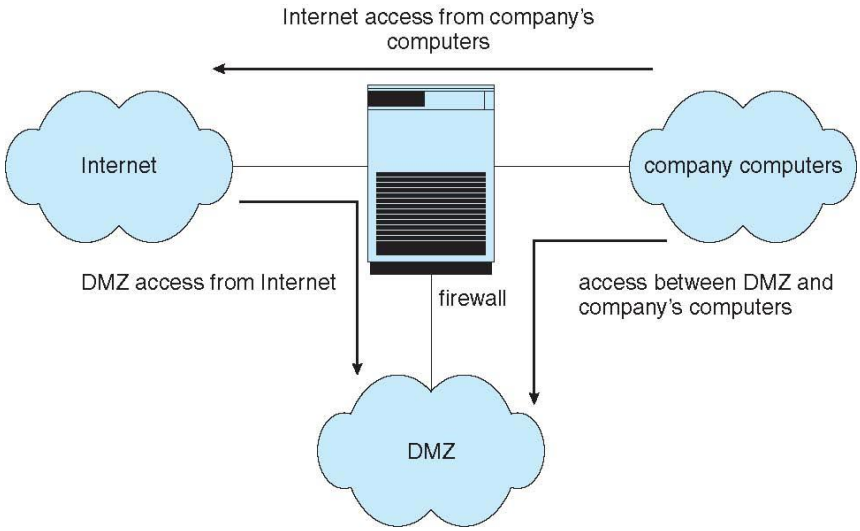
- **实现安全防御**
  - 深度防御是最常见的安全理论——多层安全
  - 安全策略描述所保护的内容
  - 漏洞评估将系统/网络的真实状态与安全策略进行比较
  - 入侵检测致力于检测尝试的或成功的入侵
    - 基于签名的检测发现已知的坏模式
    - 异常检测发现与正常行为的差异
      - 可以检测零日(zero-day)攻击
    - 假阳性（假报警）和假阴性（错过入侵）都是问题
  - 病毒防护搜索所有程序或正在执行的程序，以查找已知的病毒模式
    - 或者在沙箱中运行，以免损坏系统
  - 对所有或特定系统或网络活动进行审计、记帐和记录日志
  - 实践安全计算——避免感染源，只从"好"网站下载，等等。

Firewall

- **保护系统和网络的防火墙**
  - 网络防火墙位于受信任主机和不受信任主机之间
    - 防火墙限制这两个安全域之间的网络访问
  - 可以是隧道或欺骗（防火墙的漏洞）
    - 隧道允许不允许的协议在允许的协议内传输（例如，HTTP内部的telnet）
    - 防火墙规则通常基于可伪造的主机名或IP地址
  - 个人防火墙是给定主机上的软件层
    - 可以监视/限制进出主机的通信
  - 应用代理防火墙理解并控制应用程序的通信协议（例如SMTP）
  - 系统调用防火墙监视所有重要的系统调用并对其应用规则（即，此程序可以执行该系统调用）

Firewall

- 通过防火墙实现域分离的网络安全



Computer Security Classifications

- 计算机安全等级
  - 美国国防部(DoD)规定了计算机安全的四种等级：A、B、C和D
    - D – 最低安全性
    - C – 采用审计，为用户及其行为提供定制的保护
      - 分为 C1 和 C2
        - C1 标识具有相同保护级别的合作用户
        - C2 增加允许用户级访问控制
    - B – 具有C2级系统的所有属性，且为每个对象贴上唯一的敏感标签
      - 分为B1、B2和B3
    - A – 具有B3级，采用正式的设计规范和验证技术来确保安全性

Example: Windows 7

- **示例：Windows7**
  - 安全性基于用户帐户
    - 每个用户都有唯一的安全ID
    - 登录到ID创建安全访问令牌
      - 包括用户、用户组和特权的安全ID
      - 每个进程都获得令牌的副本
      - 系统检查令牌以确定是否允许或拒绝访问
  - 使用主题模型来确保访问安全
    - 主题跟踪并管理用户运行的每个程序的权限
    - 主题包括用户的访问令牌和代表用户运行的程序
  - Windows中的每个对象都有一个由安全描述符定义的安全属性
    - 例如，文件有一个安全描述符，指示所有用户的访问权限

Overview

- **示例：Windows7**
  - 添加了强制完整性控制 – 为每个安全对象和主题指定完整性标签
    - 给定主题必须具有定制访问控制列表中要求的权限，才能获得访问对象的权限（主题的标签不低于对象的标签）
  - 由安全描述符描述的安全属性
    - 所有者ID、组安全ID、定制访问控制列表、系统访问控制列表